Communities in Annotated, Multilayer, and Correlated Networks



Adina Roxana Pamfil
University College
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy

Trinity 2018

Acknowledgements

First and foremost, I would like to thank my supervisors, Mason Porter and Sam Howison, whose ongoing support throughout my DPhil has been essential for my success. In particular, I am grateful to Mason for his impressive knowledge of the network-science literature, for his lightning-quick feedback on all of my written work, and for encouraging me to own my project from the very beginning. Sam joined as a supervisor partway through my DPhil, but he quickly caught up with my work and made a world of difference to my research. I am grateful to him for asking seemingly simple questions that led me to a better understanding of what I was studying, and also for using our meetings to show me his new additions to his extensive collection of mathematical toys.

This doctoral research is based on work supported by the EPSRC Centre for Doctoral Training in Industrially Focused Mathematical Modelling (EP/L015803/1) in collaboration with dunnhumby. I am grateful to my industrial supervisors from dunnhumby, Rosie Prior and Shreena Patel, who trusted me fully with this project and who championed my work within the company. Other dunnhumbians past and present have also helped me in various ways during my project; in particular, I would like to thank Giles Pavey, Lorna Barclay, and Piers Stobbs, who were involved in earlier stages of my DPhil.

Among network scientists, I am thankful to Renaud Lambiotte, Marya Bazzi, Lucas Jeub, and Florian Klimm for many fruitful discussions that have made my work better. I am also grateful to my officemate, Fabian Ying, for our weekly whiteboard conversations that helped both of us stay focused at the beginning of each week. I appreciate the friendship and moral support of other network scientists in the department: particularly Andy Mellor, Alice Schwarze, Barbara Mahler, and Se-Wook Oh.

Working in the Mathematical Institute for the past four years has been a privilege, and I am very grateful for the welcoming community of people

that I found here. I would especially like to acknowledge everyone in InFoMM, including Chris, Colin, Laura, Amanda, and my peers in cohort 1. I enjoyed the good times in C6, the numerous skills-training courses, the study groups, the table football games, Monday cake, and everything else that we went through together. Applying to a new CDT felt like a bit of a gamble at the time, but I am now glad to have been a part of this fantastic programme.

This PhD provided many opportunities for reflection on my long-term mathematical training, to which many people contributed. I am especially grateful to Gheorghe Sfara, my teacher during middle school and high school, who taught me from a very young age how to write mathematics; that still serves me well today. I am also thankful to Michael Brenner, my undergraduate thesis advisor, for showing me that research can be fun and for putting the idea of a PhD on my radar for the first time.

I appreciate my family for their constant support, and for putting up with only having vague ideas of what I have been doing for the past four years. I would especially like to thank my mother, who started me on this journey of becoming a mathematician. I would not be where I am today without her sacrifices and support. And finally, I am immensely grateful to Rareș, my husband, who has helped me stay balanced and happy during the last four years, and who was a voice of reason and optimism during the twists and turns of the PhD. I owe a big part of my success to him.

Abstract

Networks are abstract representations of systems in which objects called "nodes" interact with each other via "edges", typically in a pairwise fashion. Examples of networks include neurons interacting via synapses in the brain, people connected on an online social network, and cities linked by roads. Most real-world networks have a complex structure that is neither fully random nor fully regular, and characterising their "mesoscale" structures is an important topic in network science that is traditionally known as "community detection". Two of the most popular approaches for community detection are maximising an objective function called "modularity" and performing statistical inference using stochastic block models, and these both play a prominent role in our work.

For many applications, an ordinary graph is an inadequate representation of the underlying data. In this thesis, we study community detection in networks that incorporate additional features beyond a set of nodes linked together by edges. First, we study annotated networks, which include additional data as node labels that help describe the network architecture. We then consider multilayer networks, which combine a collection of related networks into a single object. We uncover a deep connection between multilayer modularity maximisation and some multilayer stochastic block models, and we use this result to guide the selection of tuning parameters in modularity maximisation. Finally, we study correlated multilayer networks, in which we sample edges that connect a given pair of nodes across multiple layers from multivariate distributions, rather than as independent events.

Throughout this thesis, we are guided by consumer-behaviour applications. Specifically, we study "bipartite" networks in which customers are linked to products that they previously purchased. Detecting communities in these networks is valuable for designing personalised recommendation systems and for other business insights. We illustrate our main methodological contributions on consumer–product networks that we construct using "pseudonymised" transaction data from several stores of a large UK retailer.

Papers and Other Contributions

Papers

At the time of submission of this thesis, one paper (based on the work from Chapter 5) has been submitted for publication and one paper (based on the work from Chapter 6) is in preparation:

- A. R. Pamfil, S. D. Howison, R. Lambiotte, and M. A. Porter. Relating modularity maximization and stochastic block models in multilayer networks. *arXiv preprint arXiv:1804.01964*, 2018. (Submitted to SIAM Journal on Mathematics of Data Science)

- A. R. Pamfil, S. D. Howison, and M. A. Porter. Edge correlations in multilayer networks. *In preparation*, 2019. (To be submitted to Physical Review E)

A. R. Pamfil carried out the mathematical derivations, numerical simulations, and data analysis in these papers, with guidance from S. D. Howison and M. A. Porter (supervisors), as well as from R. Lambiotte for the first paper.

Software

- A. R. Pamfil. Iterative modularity maximisation algorithm for selecting resolution and interlayer coupling parameters. MATLAB implementation available at `https://github.com/roxpamfil/IterModMax`.

Company Contributions

This work involved collaboration with dunnhumby, a global customer data science company. After submitting the thesis, A. R. Pamfil continued in a 6-month knowledge transfer scheme funded jointly by the University of Oxford and dunnhumby. Contributions to the company (both during and after the DPhil) include the following: internal presentations; technical and non-technical reports; external-facing blog posts; a new way of targeting relevant customers for promotional campaigns that is currently being tested; and a hands-on workshop on network science for dunnhumby's data scientists (to be delivered in March/April 2019).

Contents

1 Introduction **1**

1.1 Mesoscale Structures in Networks 1

1.2 Consumer Behaviour 2

1.3 Organisation of the Thesis 4

2 Preliminaries **7**

2.1 Basic Definitions 7

2.2 Bipartite Networks 9

2.3 Multilayer Networks 10

2.4 Community Detection 13

2.4.1 Modularity Maximisation 14

2.4.2 Stochastic Block Models 24

2.4.3 Modularity Versus SBMs for Community Detection 29

2.5 Validation 30

2.5.1 Measures of Partition Similarity 30

2.5.2 Edge Prediction 32

3 Product-Purchase Networks **35**

3.1 Data 35

3.1.1 Transaction Data 35

3.1.2 Customer and Product Metadata 36

3.2 Network Construction 37

3.2.1 Time Windows 37

3.2.2 Edge Weights 38

3.2.3 Data Sets 39

3.3 Exploratory Community Structure 40

3.3.1 Customer and Product Communities in a Small Store 40

3.3.2 Validation Using Promotional Campaign Data 43

4 Community Detection in Annotated Bipartite Networks 49
4.1 Motivation . . . 49
4.2 Model and Statistical Inference . . . 52
4.2.1 Unweighted Networks . . . 52
4.2.2 Weighted Networks . . . 61
4.3 Number of Blocks . . . 67
4.3.1 Networks Without Annotations . . . 68
4.3.2 Networks With Annotations . . . 72
4.4 Applications to Shopping Data . . . 72
4.4.1 Network Construction . . . 72
4.4.2 Significance of Annotations . . . 73
4.4.3 Results . . . 74
4.4.4 Impact of Annotations . . . 83
4.5 Summary and Discussion . . . 84

5 Community Detection in Multilayer Networks 87
5.1 Motivation . . . 87
5.2 Existing Approaches . . . 89
5.2.1 Modularity Maximisation . . . 89
5.2.2 Statistical Models . . . 92
5.3 Relating Modularity and SBMs . . . 93
5.3.1 Monolayer Networks . . . 93
5.3.2 Multilayer Modularity and Multilayer SBMs . . . 96
5.3.3 Temporal Networks . . . 98
5.3.4 Multilevel Networks . . . 104
5.3.5 Multiplex Networks . . . 108
5.3.6 Potential Extensions to Weighted Networks . . . 112
5.4 Parameter Estimation . . . 114
5.4.1 Iterative Algorithm . . . 114
5.4.2 Estimating SBM Parameters . . . 116
5.4.3 Tests on Synthetic Networks . . . 120
5.4.4 Application to a Multiplex Social Network . . . 130
5.5 Applications to Shopping Data . . . 134
5.5.1 Temporal Networks . . . 134
5.5.2 Multilevel Networks . . . 142
5.6 Summary and Discussion . . . 150

6 Introducing Edge Correlations into Multilayer Networks 155
6.1 Motivation . . . 155
6.2 Correlated Erdős–Rényi Layers . . . 159
6.2.1 Bernoulli Model . . . 159
6.2.2 Poisson Model . . . 164
6.3 Correlated SBMs With Fixed Blocks . . . 168
6.3.1 Bernoulli Model Without Degree Correction . . . 169
6.3.2 Poisson Model Without Degree Correction . . . 172
6.3.3 Bernoulli Model With Degree Correction . . . 175
6.4 Correlated SBMs with Heterogeneous Blocks . . . 181
6.4.1 Bernoulli Model Without Degree Correction . . . 183
6.5 Edge Prediction . . . 187
6.5.1 Edge Prediction Using Correlated SBMs . . . 187
6.5.2 Tests on Synthetic Networks . . . 189
6.6 Applications to Shopping Data . . . 195
6.6.1 Correlation Calculations . . . 195
6.6.2 Edge Prediction . . . 200
6.7 Summary and Discussion . . . 201

7 Conclusions 205

Bibliography 209

Appendix 223

A Additional Results for Annotated Networks 225
A.1 Number of Blocks . . . 225
A.2 Partitions with Six Customer Blocks and Six Product Blocks . . . 225
A.3 Partitions with a Larger Number of Blocks . . . 230

B Additional Results for Multilayer Networks 233
B.1 Temporal Networks . . . 233
B.2 Multilevel Networks . . . 238

Chapter 1

Introduction

1.1 Mesoscale Structures in Networks

Characterising mesoscale organisation in networks is one of the most prominent topics in network science and is important for many applications [107]. There are numerous methods available (and more are proposed on a regular basis) for the extraction of such mesoscopic structures from networks; they include greedy techniques, statistical inference, methods based on dynamics or information theory, and machine-learning algorithms. Like network science as a whole, the problem of finding mesoscale structures in networks is an interdisciplinary endeavour, attracting expertise from fields such as mathematics, statistical physics, computer science, statistics, and sociology.

One type of mesoscale feature in networks is community structure, in which (possibly overlapping) sets of nodes are densely connected internally but sparsely connected to other dense sets of nodes [46, 48, 124]. Examples of communities in empirical networks include left-leaning and right-leaning political blogs in the US, scientists who frequently co-author papers, and groups of friends in offline and online social networks [107, 124]. Other types of mesoscale structures in networks include core–periphery structure [33, 130], hierarchical structure [28, 115, 134], and structures based on role similarity [131]. A framework that generalises many of these types of structure is that of stochastic block models (SBMs) [166], in which nodes assigned to the same block are considered "stochastically equivalent". SBMs are becoming increasingly popular for inferring the mesoscale structure of networks. In this thesis, when it is clear, we sometimes use the term "community structure" even when one can consider more general block structures.

For many applications, it is important to move beyond ordinary graphs (i.e., "monolayer networks") and consider more complicated network structures, such as by incorporating metadata or by studying a collection of interrelated networks. A

recurring theme throughout this thesis is that by analysing mesoscale structure of networks with various complications, one can obtain more meaningful results and extract insights from data that would not be accessible with more basic graph representations.

1.2 Consumer Behaviour

Our study of mesoscale structures in networks is motivated by consumer-behaviour applications. With the introduction of store loyalty cards in recent decades, there has been an ever-growing body of data on shopping habits and preferences. Despite such rich data, understanding and predicting consumer behaviour remains a difficult problem. This is due to the myriad of factors that affect consumers' decisions (anything from mood to current weather), the inherent incompleteness of the data (as customers may frequent more than one retailer), and the ever-changing needs and habits of customers (such as increased preference for healthier foods).

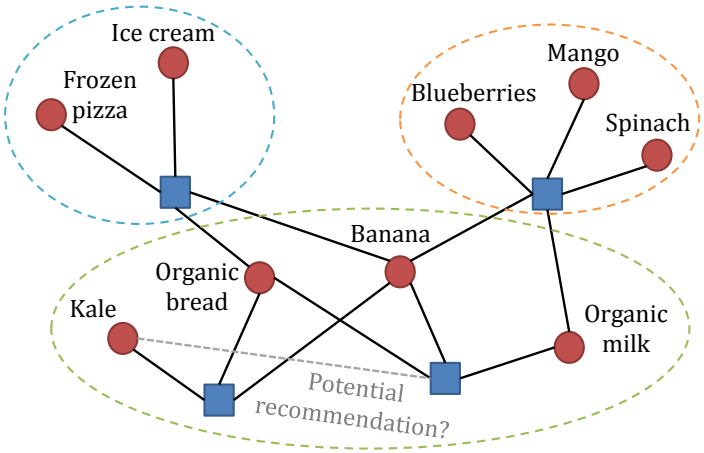


Figure 1.1: Example of a product-purchase network with three communities.

A simple network model of shopping activity consists of customers connected to products that they purchased. The resulting structure, in which there are two types of nodes and edges occur only between nodes of different types, is an example of a *bipartite network* (see Figure 1.1). It is common practice to add weights to edges to indicate lower or higher preference for different products. In this thesis, we study mesoscale structures in such networks. For example, communities of densely connected nodes in these networks reveal customers with similar purchasing practices and products that they buy more than customers in other communities. These communities also suggest indirect connections between customers and products in their communities

that they have not yet purchased, as illustrated in Figure 1.1. Such "missing links" can be incorporated into a recommendation system that gives personalised suggestions to each customer. Looking at the communities of customers and products separately can also suggest new segmentations (i.e., groupings) that stores are able to use for marketing (in the case of customers) or shelving and stocking decisions (in the case of products). Network approaches for studying consumer behaviour are especially beneficial when data is sparse (e.g., for customers with a small number of purchases), where supervised techniques often fail for lack of sufficient training data.

Above, we have touched on two major research areas that arise in the context of consumer behaviour. One involves the design of recommender systems, which have become ubiquitous in the Internet age; online users are presented with tailored choices not just for physical products, but also for movies, songs, news articles, hotels, and more. A second area is that of segmenting customers or products that have similar features. These two tasks are generally considered separately in the existing literature; by analysing the structure of bipartite consumer–product networks, it is possible to simultaneously study both problems.

The literature on recommender systems is vast [91, 128, 136, 147], and we do not attempt to discuss it in detail. We briefly mention the key concept of *collaborative filtering* [128], an idea that appears in some form in most modern recommender systems. Suppose that one wishes to provide personalised recommendations to one particular customer or user. The first step is to identify a set of customers that are similar, in some pre-defined way, to the target customer. Given this set of "collaborating" users, one then "filters" their preferences to come up with a set of relevant items for the target user. Each of the two steps can be performed in myriad ways. Popular techniques include matrix factorisation [80] and so-called "neighbourhood methods" [60], as well as hybrid approaches that combine them [79].

Previous techniques that have been used for customer segmentation[1] include K-means clustering [21], sometimes complemented by other machine-learning techniques [77, 81, 155]. In these methods, each data point is specified as a combination of features that are defined in advance by a user or are learned in some other way from the raw data (e.g., by performing factor analysis). Such features may relate to customer brand loyalty, store loyalty, size of purchase, price sensitivity, and other factors [21]. One drawback of these clustering methods is that features ought to be recalibrated over time as consumer behaviours change. Furthermore, stores in different geographical

[1]Studies that segment products, rather than customers, are considerably rarer in the literature. Given that products are often organised into well-defined hierarchies (see also Section 3.1.2), there is perhaps less of a need for additional, algorithmic classifications of these products.

areas, which are likely to have qualitatively different customer bases, may require separate training of feature sets. By contrast, by detecting communities and other mesoscale structures in networks constructed directly from transaction data, we avoid having to recalibrate features and build separate models for different stores.

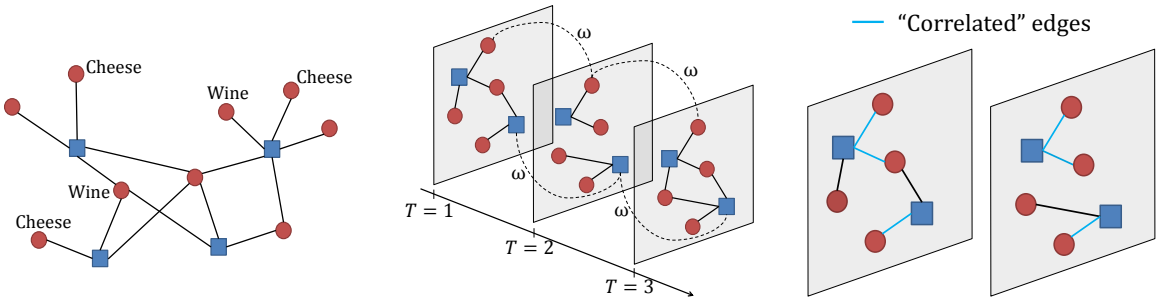
In the network-science literature, studies related to consumer behaviour are still relatively rare. Several papers [27, 29, 93] have analysed the structure of product co-purchase networks, in which edges connect products that are purchased frequently by the same customers. (One reason for the focus on products may be data availability, as data sets that include customer information raise privacy concerns.) Studies that analyse the structure of bipartite consumer–product networks for the purpose of recommendations include a paper that proposed extracting dense bipartite subgraphs as the first step in a collaborative-filtering system [126] and a paper that proposed a mixed-membership SBM for predicting item ratings [52].

The data that we consider in this thesis differs in two key ways from data sets that were analysed previously in related studies. One difference is that customers often buy the same grocery items repeatedly, whereas that is not usually the case for other settings in which consumers typically interact with items on a one-time basis (e.g., products on Amazon, movies on Netflix, and reviews on Yelp). Secondly, in the grocery-shopping setting, consumers do not usually express direct preferences through reviews (although that is changing in the context of online grocery shopping); instead, their preferences are revealed through repeated purchases of some products and not others.

1.3 Organisation of the Thesis

The rest of this thesis is organised as follows. We introduce some basic terminology related to networks and to mesoscale structures in Chapter 2. In Chapter 3, we describe the transaction data that has been provided to us, and we give details about the process of constructing networks from these data. Additionally, using some data from a previous promotional campaign, we show that network methods can be used to identify customers that are most likely to redeem a coupon for a particular product; this finding confirms that network community structure can be used to provide meaningful product recommendations. Chapters 4, 5, and 6 are research chapters. A common theme across these chapters is that they consider network representations that go beyond ordinary graphs; see Figure 1.2 for an illustration. In Chapter 4, we examine annotated networks, which include additional data in the form of node

labels. When this extra information correlates with the underlying network structure, one can infer mesoscale structure that is "better" in some sense than by using the network data alone. Chapter 5 deals with multilayer networks, including temporal, multiplex, and multilevel (i.e., hierarchical) networks. Our main result connects two popular approaches for community detection in multilayer networks. In Chapter 6, we propose models of multilayer networks that incorporate edge correlations. We then show how to use these models to calculate correlations between network layers and to improve edge-prediction performance over single-layer models. Finally, in Chapter 7, we summarise our work and we suggests a few topics for future research.



(a) Annotated network (b) Time-dependent network (c) Correlated network

Figure 1.2: Illustrations of three network representations that we examine in this thesis. (a) Annotated networks incorporate additional data as labels on the nodes. For example, one can use product categories as node annotations in a consumer–product network. (b) A time-dependent network, in which transactions from different time periods belong to different layers, is one example of a multilayer network. (c) In a correlated network, edges that connect the same pair of nodes in multiple layers are no longer independent random variables.

Chapter 2

Preliminaries

2.1 Basic Definitions

Networks are useful representations of many systems of interacting entities such as cities linked by railways, computers connected on the internet, and neurons interacting through synapses [107]. The simplest kind of network is a *graph* — a collection of *nodes* (i.e., *vertices*) that are linked to each other by *edges* in a pairwise fashion. Mathematically, a graph $\mathcal{G}$ is a pair $(\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$ is its set of nodes and $\mathcal{E} \subset \mathcal{N} \times \mathcal{N}$ is its set of edges. Two nodes $i, j \in \mathcal{N}$ are said to be *adjacent* if they are connected by an edge (i.e., if $(i, j) \in \mathcal{E}$). Let $N = |\mathcal{N}|$ be the number of vertices in a given graph. A fundamental object associated to a graph is its $N \times N$ *adjacency matrix* $\boldsymbol{A}$. In the simplest case, $\boldsymbol{A}$ is a binary matrix whose nonzero entries indicate the location of edges:

$$A_{ij} = \begin{cases} 1\,, & \text{if } (i,j) \in \mathcal{E}\,, \\ 0\,, & \text{if } (i,j) \notin \mathcal{E}\,. \end{cases} \tag{2.1}$$

An edge $(i, j) \in \mathcal{E}$ is said to be *incident* to nodes i and j.

It is helpful to distinguish between *undirected* and *directed* graphs. In undirected graphs, (i, j) is an edge if and only if (j, i) is also an edge. The adjacency matrix of an undirected graph is symmetric. That is not usually the case for directed graphs, in which an edge (i, j) is now an *ordered* pair of vertices that encodes an interaction from i to j. We say that (i, j) is an out-edge for i and an in-edge for j,[1] and the associated adjacency-matrix element is $A_{ij} = 1$. Directed networks are useful objects for representing asymmetric interactions between entities (e.g., users "following" other users on Twitter).

[1]Some researchers take the opposite convention of letting $A_{ij} = 1$ if and only if there is an incoming edge from j to i.

Another fundamental concept in network science (and graph theory) is the *degree* of a node, defined as the number of edges incident to that node. For undirected networks, the degree d_i of node $i \in \mathcal{N}$ is

$$d_i = \sum_{j=1}^{N} A_{ij} = \sum_{j=1}^{N} A_{ji} \,. \tag{2.2}$$

(Note, also, that the sum of all entries of $\boldsymbol{A}$ is equal to twice the number of edges in the corresponding network.) For directed networks, it is practical to define both *in-degrees* and *out-degrees*:

$$d_i^{\text{in}} = \sum_{j=1}^{N} A_{ji} \,, \quad d_i^{\text{out}} = \sum_{j=1}^{N} A_{ij} \,. \tag{2.3}$$

The *degree distribution* of a network is the probability distribution of observed node degrees over the whole network. Thus, if there are n_k nodes with degree k in a network with N nodes, the degree distribution is given by $P(k) = n_k/N$ (for $k \in \{0, 1, 2, \ldots\}$). For a directed network, one can similarly define the *in-degree distribution* and the *out-degree distribution*.

Often, it is useful to add *weights* to the edges of a network to indicate the strength of different interactions. For example, edge weights can indicate the number of interactions between people in a social network or the number of times that a customer purchased a product in a consumer–product network. Formally, this requires defining a weight function $w : \mathcal{E} \to U$ from the set of edges to some other set U; common choices for U are $\mathbb{Z}_+$ (the positive integers), $\mathbb{R}_+$ (the positive reals), $\{-1, +1\}$ (for modelling positive and negative interactions), the interval $[-1, 1]$ (for modelling correlations), and more. In cases where there is no ambiguity (i.e., because there are no edges with a weight of 0), it is common to encode the weights directly into a weighted adjacency matrix $\boldsymbol{W}$, so that $W_{ij} = w(i, j)$ if $(i, j) \in \mathcal{E}$ and $W_{ij} = 0$ if $(i, j) \notin \mathcal{E}$. The degree of a node is still defined as the number of edges incident to that node, regardless of weights. The *strength* of a node is defined as the total weight of the edges that are incident to it. For undirected networks, the strength s_i of node i is

$$s_i = \sum_{j=1}^{N} W_{ij} = \sum_{j=1}^{N} W_{ji} \,.$$

For directed networks, one can define the in-strength and out-strength of a node in a similar way.

Another pervasive concept in network science (and one that we use frequently in this thesis) is that of sparsity. Many empirical networks are said to be "sparse", a

term that loosely means that the number of edges is much smaller than the number of potential edges. More precisely, a network from a network ensemble is said to be *sparse* if its density ρ, defined as the number of edges divided by the number of potential edges, tends to 0 as the number of nodes N tends to infinity. In situations where the limit $N \to \infty$ is not defined — for example, because we are considering a single network rather than an ensemble of networks — the meaning of sparsity is typically context-dependent. For instance, in social networks, there is a limit to the number of connections an individual can realistically have, and that limit is independent of the number of people in the network. (See discussion of the now-famous Dunbar number in [41] and a verification of this concept for Twitter networks in [55].) Similarly, for shopping networks, the number of different products that a customer purchases does not increase linearly with the number of products that are available. For both types of networks (and for many others), we expect that the density ρ decreases as the number of nodes N increases, which justifies treating these networks as sparse. (Note that not all networks of practical interest are sparse, however. For example, asset-correlation networks [45] and functional brain networks [16] — both of which are constructed by computing correlations between time series — are normally dense.)

2.2 Bipartite Networks

A *bipartite* network is a network in which there are two types of nodes, and in which edges occur only between nodes of different types. (This is in contrast to *unipartite* networks, in which one distinguishes a single type of node.) Examples include actors connected to movies in which they acted, recipes connected to required ingredients, plants connected to their pollinators, people connected to committees on which they serve, and shoppers connected to products that they previously purchased. For concreteness, we keep this last application in mind when discussing bipartite networks throughout this thesis, although the notation and derivations that we present apply to any bipartite network. Let $\mathcal{N}_C$ and $\mathcal{N}_P$ be the sets of nodes corresponding to the two different node types — in our case, customers and products. Because edges can occur only between nodes of different types, $\mathcal{E} \subset \mathcal{N}_C \times \mathcal{N}_P$. (We assume implicitly that edges are undirected, as most bipartite networks studied in the past have been treated this way.) Let $N_C = |\mathcal{N}_C|$ and $N_P = |\mathcal{N}_P|$ be the number of customers and products, respectively; and let $N = N_C + N_P$ be the total number of nodes. Because there are no edges between nodes of the same type, the $N \times N$ adjacency matrix $\widetilde{\boldsymbol{A}}$

has the following structure:

$$\widetilde{\boldsymbol{A}} = \begin{bmatrix} \mathbf{0} & \boldsymbol{A} \\ \boldsymbol{A}^{\mathrm{tr}} & \mathbf{0} \end{bmatrix}.$$

The matrix $\boldsymbol{A}$ is a rectangular matrix of size $N_C \times N_P$, and $\boldsymbol{A}^{\mathrm{tr}}$ denotes its transpose; the $\mathbf{0}$ matrices along the diagonal of $\widetilde{\boldsymbol{A}}$ have dimensions $N_C \times N_C$ and $N_P \times N_P$, respectively. The matrix $\boldsymbol{A}$ is sometimes called a *bi-adjacency* matrix to emphasise that it corresponds to a bipartite network; however, we refer to $\boldsymbol{A}$ as an adjacency matrix, and we do not work with the full matrix $\widetilde{\boldsymbol{A}}$ unless explicitly indicated.

Instead of analysing a bipartite network directly, it is common to first construct a unipartite projection onto one of the two node sets of the orginal network [172]. This is called a *one-mode projection*. Two nodes of the same type (e.g., two customers or two products) are adjacent in the projection if they share at least one common neighbour in the original bipartite network. One often weights these edges by the number of neighbours in the original network, in which case the adjacency matrices of the projected networks are $\boldsymbol{A}\boldsymbol{A}^{\mathrm{tr}}$ and $\boldsymbol{A}^{\mathrm{tr}}\boldsymbol{A}$. For our application, a one-mode projection onto the set of customers has an edge between two customer nodes if they have at least one purchased product in common; a one-mode projection onto the set of products has an edge between two product nodes if they have both been purchased by the same customer. One problem with one-mode projections (both weighted and unweighted ones) is that they entail loss of information, as multiple bipartite networks can have the same unipartite projection. Another issue is that these projections are often much denser than the original networks, posing problems to various methods that work best in sparse regimes. As a result, it is desirable to work with bipartite networks whenever possible, without applying additional transformations. This is especially important for community detection. Indeed, as discussed in [11], one can easily obtain misleading results when detecting communities in one-mode projections of bipartite networks. We discuss how to perform community detection directly in a bipartite network in Section 2.4.

2.3 Multilayer Networks

For many applications of network science, it is important to move beyond ordinary graphs (i.e., "monolayer networks") and consider more complicated network structures, such as by studying a collection of interrelated networks. One can study such structures through the flexible lens of multilayer networks [23, 35, 78].

Like monolayer networks, a multilayer network can be construed as a collection of nodes connected pairwise by (possibly weighted) edges, $\mathcal{M} = (\mathcal{N}_{\mathcal{M}}, \mathcal{E}_{\mathcal{M}})$. However,

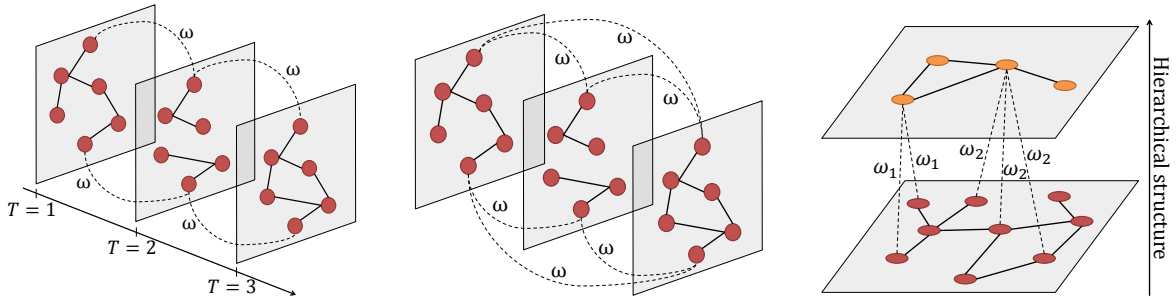
each node is now identified as belonging to a particular layer. We illustrate this general framework with an example adapted from [17]. Consider a collection of individuals who interact with each other on both Twitter and Instagram (e.g., via "retweets" or "likes"). Additionally, assume that we aggregate these interactions into time windows of fixed length (e.g., 1 day). In the language of [78], this network has two "aspects"; one identifies the social network (i.e., Twitter or Instagram) of a particular node, and one identifies a time window (e.g, the one corresponding to the day 20 August 2018). Each layer $\boldsymbol{\alpha}$ is then identified by a sequence of aspects. Edges occur between node-layer tuples of the form $(i, \boldsymbol{\alpha}) \in \mathcal{N}_{\mathcal{M}}$. For simplicity, and unless we specify otherwise, we use the term "node" to refer to such a node-layer tuple in the context of multilayer networks. It is also common to refer to such structures as "state nodes" [17, 133]. On occasion, it is useful to consider all node-layer tuples that correspond to the same entity (e.g., a single person that can appear as a Twitter or Instagram user in any of the layers). We use the term "physical node" to refer to this common entity, following terminology from [133] that was later adapted for multilayer networks [17, 36].

Edges in a multilayer network can be categorised as either intralayer edges, which are of the form $((i, \boldsymbol{\alpha}), (j, \boldsymbol{\alpha}))$, or interlayer edges, which are of the form $((i, \boldsymbol{\alpha}), (j, \boldsymbol{\beta}))$ for $\boldsymbol{\alpha} \neq \boldsymbol{\beta}$. It is also helpful to distinguish interlayer edges that connect state nodes in different layers that correspond to the same physical node. These take the form $((i, \boldsymbol{\alpha}), (i, \boldsymbol{\beta}))$ and are often called "coupling" edges [17, 78]. It is common to use coupling edges to identify state nodes that correspond to the same physical node. In the earlier example, such edges include those of the form (i, ("Twitter", "20 August 2018"), (i, ("Instagram", "20 August 2018"))) and those of the form (i, ("Instagram", "20 August 2018"), (i, ("Instagram", "21 August 2018"))).

In this general framework, the sets of state nodes in each layer need not be identical. (For example, someone can have a Twitter account but not an Instagram account.) In practice (and for simplicity, even though some generality is lost), researchers typically assume that each node is present in all layers. These were called "fully-interconnected" or "node-aligned" networks in [78]. In situations in which this is not the case, one can "pad" a network with additional nodes that are not adjacent to any other nodes in their layer. We primarily consider networks in which coupling edges are the only type of interlayer edges; one exception are multilevel networks (in which interlayer edges consist of inclusion relations and represent hierarchical relationships between layers), which we define and discuss in detail in Chapter 5. Following [78], we say that these networks have "diagonal" interlayer structure. Finally, we say that a multilayer

network is "layer-coupled" if the weight of interlayer edges depends only on the layers (and not on the nodes themselves) [78].

In this thesis, we study exclusively multilayer networks with a single aspect. We distinguish between networks with "categorical" or "ordinal" coupling. The archetypal example of the former are multiplex networks, a generalisation of edge-coloured multigraphs [78] in which intralayer edges correspond to different types of interactions between nodes. A common example of networks with ordinal coupling are temporal multilayer networks, in which layers are ordered in time. Throughout this thesis, we refer to such networks simply as "temporal networks", although we note that the term has a much broader meaning in the literature [63]. Typically, coupling edges in a multilayer representation of a temporal network have been used only to connect nodes in consecutive layers; when one makes this assumption, any dependencies between layers that are further apart flow indirectly through these edges along the arrow of time. In Figure 2.1, we illustrate different types of multilayer structure that we study in this work (primarily in Chapter 5).



(a) Temporal network (b) Multiplex network (c) Multilevel network

Figure 2.1: Networks with different kinds of multilayer structure.

We now introduce some notation that we use later in this thesis, focusing on fully-interconnected multilayer networks. Let T denote the number of layers, and let N denote the number of nodes in each layer (i.e., the number of physical nodes). We write $\boldsymbol{A}^t$ for the (possibly weighted) $N \times N$ adjacency matrix of the network in layer $t \in \{1, \dots, T\}$. In addition to these intralayer edges, for any two distinct layers $s, t \in \{1, \dots, T\}$, the $N \times N$ matrix $\boldsymbol{\Omega}^{st} = (\omega_{ij}^{st})_{1 \leq i,j \leq N}$ specifies the weights of interlayer connections between nodes in those layers. When these interlayer edges are diagonal, the matrices $\boldsymbol{\Omega}^{st}$ are themselves diagonal, and we use $\omega_i^{st} \equiv \omega_{ii}^{st}$ to denote the i^{th} diagonal entry. A convenient representation of multilayer networks is a

supra-adjacency matrix [53, 78], which has the following block structure:[2]

$$\boldsymbol{\mathcal{A}} = \begin{bmatrix} \boldsymbol{A}^1 & \boldsymbol{\Omega}^{12} & \cdots & \boldsymbol{\Omega}^{1T} \\ \boldsymbol{\Omega}^{21} & \boldsymbol{A}^2 & \cdots & \boldsymbol{\Omega}^{2T} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\Omega}^{T1} & \boldsymbol{\Omega}^{T2} & \cdots & \boldsymbol{A}^T \end{bmatrix} .$$

For multilayer bipartite networks, we extend the notation from Section 2.2 and let $\widetilde{\boldsymbol{A}}^t$ denote the adjacency matrix of layer t, which has a block structure given by

$$\widetilde{\boldsymbol{A}}^t = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{A}^t \\ (\boldsymbol{A}^t)^{\mathrm{tr}} & \boldsymbol{0} \end{bmatrix} .$$

For networks that are fully interconnected, each matrix $\boldsymbol{A}^t$ is an $N_C \times N_P$ rectangular matrix, where N_C and N_P are the numbers of nodes of each type (e.g., customers and products) in each layer.

2.4 Community Detection

An enormous amount of research on networks is dedicated to studying mesoscale structures in networks [107]. Detecting such structures amounts to a type of coarse-graining, providing representations of a network that are more compact than listing all the edges. For example, one can partition a network into sets or blocks of nodes based on their distinctive interaction patterns, or one can assign probability distributions of block assignments to a network's nodes. Types of mesoscale structures include community structure [48], core–periphery structure [33, 130], role similarity [131], and others.

In this thesis, we focus on community detection [1, 46, 48, 118, 124, 137], although some of the methods that we discuss apply more broadly (see Section 2.4.2). There is no consensus definition of what constitutes a community, and the precise definition typically depends on the methodology that one uses to detect communities [48]. In qualitative terms, communities are sets of nodes that are densely connected with each other and sparsely connected with other nodes in a network. Communities can reveal meaningful structure in networks; for example, they can uncover functional pathways in metabolic networks, related pages in the World Wide Web, groups of friends in social networks, and more [107, 124].

Many existing community-detection methods involve the optimisation of some quality function, such as modularity [104, 110], Markov stability [39, 83, 84], the

[2]We note that the supra-adjacency matrix of a multilayer network corresponds to the adjacency matrix of the monolayer network that one obtains by disregarding layer information.

"map equation" in the Infomap method [36, 132], and likelihood functions or posterior probabilities from generative network models [76, 109, 114, 119]. Among these approaches, we distinguish two types (see also [50]). The first set of methods uses a quality function based on heuristic or other arguments. For example, one may construe a partition of a network as "better" if its communities have a "surprising" number of connections compared with some null model (as in modularity, which is heuristic), if they form bottlenecks to random walks (as in Markov stability), or if they give a good compression of a network from an information-theoretic perspective (as in Infomap). The second set of methods encompasses statistical models of networks with mesoscale structures (in particular, community structure), which one then fits to data using some form of statistical inference. These models take many forms, and it is particularly popular to use various types of stochastic block models (SBMs).

We discuss modularity maximisation and SBM inference in Sections 2.4.1 and 2.4.2, respectively, and we briefly compare their strengths and weaknesses in Section 2.4.3. See the review articles [46, 48, 124] for details on other types of methods for community detection.

2.4.1 Modularity Maximisation

Modularity is an objective function that measures the "quality" of a network partition into disjoint sets of nodes known as "communities". Larger values of modularity correspond to partitions in which more edges fall within groups than would be expected by chance, as quantified by a null model. Let g_i denote the (unknown) community label of node i. One seeks to determine a vector $\boldsymbol{g}$ of community assignments that maximises modularity. Consider a monolayer network with N nodes and its (possibly weighted) $N \times N$ adjacency matrix $\boldsymbol{A}$. A general form of modularity is

$$Q(\boldsymbol{g}) = \sum_{i,j=1}^{N} (A_{ij} - \gamma P_{ij}) \delta(g_i, g_j) \,, \tag{2.4}$$

where P_{ij} gives the expected number of edges (or the expected edge weight, if considering weighted networks) between nodes i and j under some specified null model, and γ is a specified *resolution parameter* [127]. The Kronecker delta $\delta(\cdot, \cdot)$ is equal to 1 whenever its arguments are the same, and it is equal to 0 when they are different. Therefore, the only terms that contribute to the sum in equation (2.4) are ones in which nodes i and j are in the same community. The matrix $\boldsymbol{B} = \boldsymbol{A} - \gamma \boldsymbol{P}$, given $\boldsymbol{P}$ and γ, is called the *modularity matrix*. Sometimes, it can be helpful to normalise $Q(\boldsymbol{g})$ by $2m = \sum_{i,j=1}^{N} A_{ij}$, where m is the number of edges (or the total weight) in

the network associated with the adjacency matrix $\boldsymbol{A}$. When maximising normalised modularity with respect to $\boldsymbol{g}$, one construes values closer to 1 as indicating stronger community structure in a given network. This requires caution, however, as one can obtain modularity values that are arbitrarily close to 1 in partitions of trees constructed in a certain way [10], despite the fact that trees are not usually regarded as having modular structure.

We can now formulate community detection in terms of finding a vector $\boldsymbol{g}$ of community assignments that maximise $Q(\boldsymbol{g})$. In practice, because modularity maximisation is NP-hard [24], one generically finds only local optima of this objective function.

Null models

Maximising modularity as written in (2.4) requires the choice of a null model and its corresponding expected adjacency matrix $\boldsymbol{P}$ [18]. Entries of this matrix correspond to the number or the weight of edges that are expected "at random" between two nodes (i.e., if the given network had no community structure). Depending on context, different null models may be suitable, and researchers need to decide how much domain-specific knowledge to incorporate into a null model (rather than learn algorithmically). For instance, specialised null models have been developed for spatially-embedded, temporal networks [135] and for contact networks in granular media [14].

For undirected monolayer networks, the version of modularity that is used most often employs the Newman–Girvan null model [110] (which is closely related to configuration models [49]). In this case, the modularity function compares the entries of a given network's adjacency matrix $\boldsymbol{A}$ with the expectation of the entries after uniformly randomly rewiring the edges such that the expected degrees of the nodes are preserved:

$$Q(\boldsymbol{g}) = \sum_{i,j=1}^{N} \left(A_{ij} - \gamma \frac{d_i d_j}{2m} \right) \delta(g_i, g_j) \,, \tag{2.5}$$

where d_i is the degree of node i (and d_j is the degree of node j), and m is the number of edges. The formula (2.5) is also valid for weighted networks [103], where A_{ij} is now an entry of the weighted adjacency matrix $\boldsymbol{A}$, d_i and d_j are the strengths of nodes i and j, and m is the total edge weight in the network. Versions of the Newman–Girvan null model that are appropriate for networks with negative weights, with self-loops, and with directed edges are also available [54].

Null models for bipartite networks

The defining characteristic of bipartite networks is that edges occur only between nodes of different types (e.g., between a customer and a product). When maximising modularity, it is therefore appropriate to compare the observed network to a null network that also has this characteristic. (Otherwise, node pairs (i, j) bring a negative contribution to the modularity (2.5) whenever i and j are of the same type.) The usual choice is an analogue of the Newman–Girvan null model that Barber proposed for the bipartite setting [11]. In this null model, the matrix $\boldsymbol{P}$ gives the expected edge weights over all networks on the original set of nodes where edges are placed uniformly at random, subject to the constraints that the total edge weight from each vertex is the same as in the original network and the bipartite structure of the network is preserved. Thus, $P_{ij} = 0$ if i and j are nodes of the same type (in our case, two customer nodes or two product nodes). Otherwise,

$$P_{ij} = \frac{d_i d_j}{m}\,, \tag{2.6}$$

where $d_i = \sum_{j \in \mathcal{N}_P} A_{ij}$ and $d_j = \sum_{i \in \mathcal{N}_C} A_{ij}$ are the strengths from nodes $i \in \mathcal{N}_C$ and $j \in \mathcal{N}_P$, respectively; and $m = \sum_{i \in \mathcal{N}_C} \sum_{j \in \mathcal{N}_P} A_{ij}$ is the total edge weight in the network.

The Barber null model is a general-purpose null model for bipartite networks. For specific applications, it can be useful to devise specialised null models that incorporate additional domain-specific knowledge. For instance, one of the assumptions of the Barber null model is that any two nodes can form an edge, as long as they are of different types. This assumption can be inaccurate for consumer–product networks, in which, for example, customers who are vegan do not purchase meat or dairy products, customers who do not have pets do not purchase pet food, and so on. Devising a null model that respects these patterns (and other less obvious ones) requires significant effort and needs to be done with care, so as not to bias community-detection algorithms in unintended ways. While the choice of null model does impact the outputs of community-detection algorithms, we believe that other aspects of consumer–product networks are more salient (see Section 1.3). Thus, throughout this thesis, we use the Barber null model whenever we apply modularity maximisation to bipartite networks.

Resolution parameters

The resolution parameter γ, which is equal to 1 in the traditional formulation of modularity [110], was introduced in [127] and mitigates (but does not remove) resolution issues, wherein communities below a certain size are undetectable using the

original definition of modularity [47, 56]. For small resolution values, when $0 \leq \gamma \leq \min_{i \neq j, P_{ij} \neq 0} \{A_{ij}/P_{ij}\}$, modularity maximisation places all nodes in one community and we obtain the coarsest possible community structure [18]. At the other extreme, when $\gamma \geq \max_{i \neq j, P_{ij} \neq 0} \{A_{ij}/P_{ij}\}$, modularity is maximised when each node is placed in its own community, giving the finest possible community structure [18]. By tuning this resolution parameter, one can explore community structure in a network at intermediate scales.

Selecting the right resolution values for examining the structure of a network is a difficult problem, for which several approaches exist. Many of these rely on running modularity maximisation for several values of γ and then using some criterion to select a subset of values for further analysis [158]. For instance, a common approach is to look for long plateaus in the number of detected communities (or other similar quantities [45]), as these correspond to resolution ranges where detected communities are relatively stable. When using a modularity-maximisation algorithm that is stochastic in nature [73], producing different partitions even for fixed γ, it can be useful to look for resolution ranges where these output partitions are similar to each other. (We use this approach when we apply modularity maximisation to multilayer consumer–product networks in Chapter 5.) When detected communities are used for prediction (see, for example, Section 3.3.2), one can adopt an approach that is common in machine learning and select the resolution value that has the best predictive performance on a training data set. Other methods include combining partitions obtained at different resolution values through a consensus-clustering approach [75] and deriving formulas for the "optimal" γ value by comparing modularity to statistical objective functions [105] (see also Chapter 5).

In this thesis, we use the Reichardt–Bornholdt [127] resolution parameter γ to control the sizes of detected communities, as this enables us to build upon existing work for community detection, especially in multilayer networks [98]. Using γ to control the relative importance of a null model when performing modularity maximisation is a common (and intuitive) way to explore the multiscale structure of networks and to get around some types of resolution limits. It also has the appealing property that it can be tied to the timescale of certain random-walker dynamics on networks [82–84]. However, one limitation of this approach is that for many networks, there does not exist a single resolution value that can simultaneously resolve communities of all sizes. Instead, as one increases the resolution, some large communities split before smaller ones become visible. Arenas et al. [7] proposed a way to mitigate this problem. In their approach, one starts from a weighted adjacency matrix $\boldsymbol{W}$

and then runs modularity maximisation on a new matrix $\boldsymbol{W}_r = \boldsymbol{W} + r\boldsymbol{I}$ for varying values of r, both positive and negative. (This amounts to adding self-loops of weight r to all nodes in the original network.) By increasing r, one tends to obtain smaller communities. With this method, it is possible to resolve communities at disparate scales using a single value of r. However, the resolution limit does not disappear entirely [160]. Other strategies for circumventing the resolution limit of modularity include reweighting the edges of a given network [19]. The existence of all of these different methods for exploring the structure of networks at different scales suggests that this is still an open problem.

Modularity maximisation in multilayer networks

Given the popularity of modularity maximisation for detecting communities in monolayer networks, it is no surprise that researchers have looked to extend it to the multilayer setting. The initial development came with [98], where the authors extended previous studies that described communities as bottlenecks to certain random-walker dynamics [82–84]. The key difference in multilayer settings is that dynamics spread along both intralayer and interlayer edges; without such coupling, detecting communities in multilayer networks is no different from doing so separately in each layer.

We now recall some of the notation from Section 2.3. We focus on fully-interconnected networks, in which each node is present in all layers, although the edges can be different. Let $\boldsymbol{A}^t$ denote the $N \times N$ adjacency matrix of the network in layer $t \in \{1, \ldots, T\}$. We assume for simplicity that the intralayer edges are undirected, and we note that one can introduce versions of multilayer modularity for directed situations. Assuming, as in [98], that all interlayer edges are diagonal, we let ω_i^{st} denote the weight of the coupling edge between copies of physical node i in two distinct layers s and t. (Note that we allow these edges to be directed, so it is not necessarily the case that $\omega_i^{st} = \omega_i^{ts}$.)

Detecting communities in a multilayer network amounts to assigning each node–layer tuple (i, t) to a community, which we label g_i^t. Analogously to the monolayer case, modularity is a function that measures the quality of a multilayer partition $\boldsymbol{g}$ of the nodes into communities, relative to some null model. Using the standard Newman–Girvan null model [110], the authors of [98] derived the following expression[3]

[3]We write (2.7) using somewhat different notation from [98].

for multilayer modularity:

$$\begin{aligned} Q(\boldsymbol{g}) &= \sum_{t=1}^{T} \sum_{i,j=1}^{N} \left(A_{ij}^{t} - \gamma_t \frac{d_i^t d_j^t}{2m_t} \right) \delta(g_i^t, g_j^t) + \sum_{t=1}^{T} \sum_{s \neq t} \sum_{i=1}^{N} \omega_i^{st} \delta(g_i^s, g_i^t) \\ &= Q_{\text{intra}}(\boldsymbol{g}) + Q_{\text{inter}}(\boldsymbol{g}) \,. \end{aligned} \tag{2.7}$$

In (2.7), d_i^t is the degree of node i in layer t, and the quantity m_t is the number of edges (or edge weight, for weighted networks) in layer t. The layer-specific resolution parameters γ_t control the importance of the null-model network relative to the observed network. Multilayer modularity is the sum of intralayer terms $Q_{\text{intra}}(\boldsymbol{g})$ (which add the monolayer modularity values that are contributed by each layer) and interlayer terms $Q_{\text{inter}}(\boldsymbol{g})$, which depend on the multilayer partition $\boldsymbol{g}$ and the coupling between the layers. One can perform community detection by finding multilayer partitions $\boldsymbol{g}$ that maximise (2.7). In practice, because this is an NP-hard problem [24], algorithms yield local maxima of $Q(\boldsymbol{g})$.

For bipartite networks, one obtains a similar formula by adjusting the terms corresponding to the null model (by using the Barber null model [11] in place of that of Newman and Girvan [110]) and by summing over appropriate node indices:

$$\begin{aligned} Q(\boldsymbol{g}) &= 2 \sum_{t=1}^{T} \sum_{i \in \mathcal{N}_C} \sum_{j \in \mathcal{N}_P} \left(A_{ij}^{t} - \gamma_t \frac{d_i^t d_j^t}{m_t} \right) \delta(g_i^t, g_j^t) + \sum_{t=1}^{T} \sum_{s \neq t} \sum_{i \in \mathcal{N}_C \cup \mathcal{N}_P} \omega_i^{st} \delta(g_i^s, g_i^t) \\ &= Q_{\text{intra}}(\boldsymbol{g}) + Q_{\text{inter}}(\boldsymbol{g}) \,. \end{aligned} \tag{2.8}$$

When all interlayer coupling parameters take the same value, $\omega_i^{st} = \omega$, note that the interlayer modularity term can be written as

$$Q_{\text{inter}}(\boldsymbol{g}) = \omega \sum_{t=1}^{T} \sum_{s \neq t} \sum_{i=1}^{N} \delta(g_i^s, g_i^t) \,.$$

For temporal networks in which interlayer edges connect only neighbouring layers, this expression takes the simpler form

$$Q_{\text{inter}}(\boldsymbol{g}) = \omega \sum_{t=1}^{T-1} \sum_{i=1}^{N} \delta(g_i^t, g_i^{t+1}) \,.$$

Bazzi et al. [18] introduced the notion of *persistence* of a multilayer partition $\boldsymbol{g}$, defined as

$$\text{Pers}(\boldsymbol{g}) = \sum_{t=1}^{T-1} \sum_{i=1}^{N} \delta(g_i^t, g_i^{t+1}) \,.$$

Thus, Pers($\boldsymbol{g}$) counts the number of nodes that stay in the same community between consecutive layers of the temporal network. (One can also define persistence for other types of multilayer structure.) It follows that maximising multilayer modularity is a trade-off between finding a close-to-optimal partition for each layer (i.e., by maximising intralayer modularity terms) and finding persistent communities. By adjusting the coupling parameter ω, one can control the relative contribution of persistence terms to the overall objective function.

It is convenient to introduce an alternative formulation of the modularity function in (2.7). For each layer t, we define a *modularity matrix* $\boldsymbol{B}^t$, whose entries are given by

$$B^t_{ij} = A^t_{ij} - \gamma_t \frac{d^t_i d^t_j}{2m_t} \,.$$

Similarly to the supra-adjacency matrix $\boldsymbol{\mathcal{A}}$ from Section 2.3, consider the multilayer modularity matrix

$$\boldsymbol{\mathcal{B}} = \begin{bmatrix} \boldsymbol{B}^1 & \boldsymbol{\Omega}^{12} & \dots & \boldsymbol{\Omega}^{1T} \\ \boldsymbol{\Omega}^{21} & \boldsymbol{B}^2 & \dots & \boldsymbol{\Omega}^{2T} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\Omega}^{T1} & \boldsymbol{\Omega}^{T2} & \dots & \boldsymbol{B}^T \end{bmatrix} .$$

Introducing global indexes for the nodes, $i' = i + (t-1)N$ (such that nodes in the first layer are labelled $1, \dots, N$, nodes in the second layer are labelled $N+1, \dots, 2N$, etc.), the modularity in equation (2.7) is

$$Q(\boldsymbol{g}) = \sum_{i',j'=1}^{NT} B_{i'j'} \delta(g_{i'}, g_{j'}) \,. \tag{2.9}$$

This general objective function is the one that is maximised in the software known as GenLouvain [73], a multilayer generalisation of the popular Louvain method for monolayer modularity maximisation [22].[4] Any matrix $\boldsymbol{\mathcal{B}}$ is a valid input to this algorithm. In particular, with this formulation, one can write modularity functions that correspond to various types of interlayer coupling (e.g., temporal, multiplex, multilevel), various types of intralayer structures (e.g., unipartite or bipartite, undirected or directed, and unweighted or weighted), and various null models. This highlights the flexibility of modularity-based community-detection methods, which allow different types of network structure and different choices of null models.

[4]Having an algorithm that is able to perform community detection at this level of generality is important for some of the models that we consider in Chapter 5.

Louvain method for optimising modularity

The specific modularity-maximisation algorithm that we use is a variant of the greedy method described in [22], which is known in the literature as the "Louvain method". This method is very fast, and it has been shown to be feasible for networks with millions of nodes [22]; it also performs well on benchmarks [85]. The Louvain algorithm starts by assigning each node to its own community, and it then repeats two phases of successive merges and swaps of nodes that increase the value of modularity until attaining a local maximum. These two phases are as follows:

1. **Modularity maximisation.** For each node i, consider its neighbour j for which the gain in modularity from removing i from its community and placing it in the community of j is maximal. If this gain is positive, move i to the new community. Go through all of the nodes repeatedly until no further gain in modularity is possible.

2. **Community aggregation.** Build a new network whose nodes are the communities identified in the previous step. Edge weights in the new network are given by the sums of relevant edge weights in the original one, such that modularity is preserved [6].

We use a MATLAB implementation of this modularity-maximisation heuristic provided in [73]. This particular variant of the Louvain algorithm, known as GENLOUVAIN, differs in a few key ways from the original version in [22]. The main difference is that GENLOUVAIN works on any modularity matrix $\boldsymbol{B}$, rather than just on those based on a Newman–Girvan null model. The added level of generality results in slower computations (because one can no longer implement the fast modularity updates from [22] when a node switches communities), but it also provides a flexible algorithm that can be applied to networks with various complications, including bipartite and multilayer networks. There are also several less substantive differences between the two implementations. GENLOUVAIN randomises the order in which nodes are considered in the modularity-maximisation phase, whereas the original Louvain method [22] uses by default a deterministic node order based on pre-defined labels. It is also possible to alter the modularity-maximisation phase so that, instead of moving a node to the community that yields the largest increase in modularity, the algorithm considers all moves that would result in an increase; and it then selects one such move with a probability proportional to the resulting increase in modularity. The new algorithm, which we (and others [17]) call GENLOUVAINRAND, alleviates some problematic behaviour of GENLOUVAIN for multilayer networks [18].

Because of the stochastic components of both GenLouvain and GenLouvain-Rand, and because these algorithms only find local maxima of a modularity function, each different run typically returns a different partition of a given network into communities, even for fixed γ and ω. This is desirable behaviour, as it allows one to better explore the complicated modularity landscape [56] than by relying on a single (deterministic) output. The challenge lies in interpreting (and possibly combining) partitions obtained across different runs.

Consensus clustering

Consensus clustering [146] provides one way to extract robust community assignments from a set of partitions obtained through repeated runs of a community-detection algorithm, such as GenLouvain. By combining information from different runs, a consensus partition summarises patterns that are consistent across a variety of community-detection outcomes.

Our implementation of consensus clustering follows the procedure in [87], with the null model from [70]. (A monolayer version of this null model, which the authors called the "permutation null model", also appeared in [75].) Given a sample of partitions of a network's nodes into communities, the first step is to construct a *co-classification matrix* (also called an *association matrix*) $\boldsymbol{C}$, whose $(i,j)^{\text{th}}$ entry is equal to the fraction of partitions in the given sample in which nodes i and j belong to the same community. More precisely, given a sample $\{\boldsymbol{g}^{(1)}, \ldots, \boldsymbol{g}^{(L)}\}$ of L partitions, we set

$$C_{ij} = \frac{1}{L} \sum_{l=1}^{L} \delta(g_i^{(l)}, g_j^{(l)}) \,,$$

where $\delta(\cdot,\cdot)$ is the Kronecker delta. From this association matrix, one can construct a new weighted network, in which the weight between two nodes i and j is equal to C_{ij}. (To ensure that the new network is reasonably sparse, it is common to set all entries of $\boldsymbol{C}$ that are below some pre-defined threshold to 0 [15, 87].) Given a choice of null model for this new network, one then applies community detection again to generate a new sample of partitions and a corresponding association matrix. The process is repeated until the association matrix is binary and perfectly block diagonal, such that community assignments do not change from run to run. This final output is the consensus partition.

Lancichinetti and collaborators [87] proposed using a uniform null model for community detection in networks constructed from association matrices. This is a better choice than using a Newman–Girvan null model [110], which tends to penalise large

sets of nodes even when they have high co-classification values. However, a uniform null model tends to produce many small communities in the consensus partition. We obtain better results by using the permutation null model from [70, 75]. Qualitatively, this null model subtracts from a co-classification value C_{ij} the expected co-classification value if one were to reshuffle community labels within each partition from a given sample, keeping the community sizes in each partition fixed. Given a sample $\{\boldsymbol{g}^{(1)}, \ldots, \boldsymbol{g}^{(L)}\}$ of L partitions, the expectation of C_{ij} can be approximated as

$$\mathbb{E}[C_{ij}] \approx \frac{1}{L} \sum_{l=1}^{L} \mathbb{P}(g_i^{(l)} = g_j^{(l)}) = \frac{1}{L} \sum_{l=1}^{L} \sum_{r=1}^{K^{(l)}} \mathbb{P}(g_i^{(l)} = r, g_j^{(l)} = r)\,, \tag{2.10}$$

where $K^{(l)}$ is the number of communities in the l^{th} partition. Additionally, let $n_r^{(l)}$ be the number of nodes in community r in the l^{th} partition, and let N be the number of nodes in the entire network. Under a permutation null model that preserves community sizes [70, 75], the probability that two nodes i and j are both in community r is

$$\mathbb{P}(g_i^{(l)} = r, g_j^{(l)} = r) = \begin{cases} \frac{n_r^{(l)}(n_r^{(l)}-1)}{N(N-1)}\,, & \text{if } i \neq j\,, \\ \frac{n_r^{(l)}}{N}\,, & \text{if } i = j\,. \end{cases}$$

One can now detect communities in the network induced by the association matrix $\boldsymbol{C}$ by maximising the modularity from equation (2.4), where one sets the entries of the null matrix $\boldsymbol{P}$ to the values calculated from (2.10).

A similar approach works for multilayer networks. For a fully-interconnected multilayer network with T layers and N nodes in each layer, one can define an $NT \times NT$ association matrix in the usual way. The null model suggested in [70] is one that permutes community labels within each layer of a given multilayer partition, keeping community sizes fixed in each layer. Under this null model, a node i in layer s and a node j in layer t both have the same community label r with probability

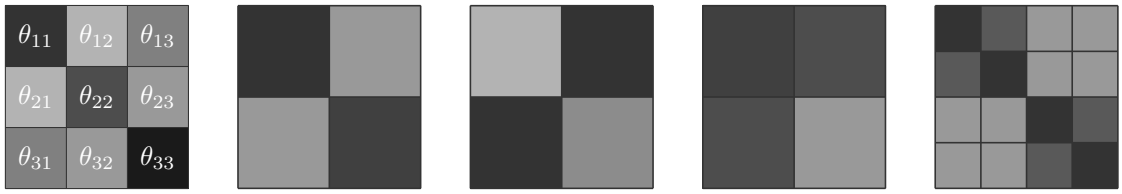
$$\mathbb{P}(g_{i,s}^{(l)} = r, g_{j,t}^{(l)} = r) = \begin{cases} \frac{n_{r,s}^{(l)} n_{r,t}^{(l)}}{N^2}\,, & \text{if } s \neq t\,, \\ \frac{n_{r,s}^{(l)}(n_{r,s}^{(l)}-1)}{N(N-1)}\,, & \text{if } i \neq j \text{ and } s = t\,, \\ \frac{n_{r,s}^{(l)}}{N}\,, & \text{if } i = j \text{ and } s = t\,, \end{cases}$$

where $g_{i,s}^{(l)}$ is the community of node i in layer s of the l^{th} partition (and $g_{j,t}^{(l)}$ is defined analogously) and $n_{r,s}^{(l)}$ denotes the number of nodes in community r in layer s of the l^{th} partition.

We use consensus clustering in Chapter 5 to obtain representative partitions for both monolayer and multilayer consumer–product networks.

2.4.2 Stochastic Block Models

Broadly speaking, stochastic block models (SBMs) are ensembles of random graphs in which nodes belong to different clusters (i.e., "blocks" or "groups"), and edges occur with probabilities that depend on this underlying block structure [1, 97, 118]. Given a forward model for sampling networks from a particular SBM, one is typically interested in solving the inverse problem of fitting the model to a given network. This *inference* process returns the most likely block structure (or a probability distribution over possible block structures) of the network, akin to how one uses modularity maximisation to detect communities. However, SBMs go beyond the detection of purely assortative communities (i.e., with a larger internal density than external density). Other examples of structures that can be detected using SBMs are disassortative structure (in which there are more edges between blocks than within blocks), core–periphery structure, hierarchical structure, and more (see Figure 2.2). Many variants of SBMs have been proposed in the literature. Examples that go beyond the most basic versions include SBMs for networks with overlapping blocks [52], for annotated networks [65, 109], for weighted networks [3, 119], and for various types of multilayer structure [34, 116, 153]. In this chapter, we introduce basic variants of SBMs for unweighted monolayer networks. We discuss more complicated SBMs (including weighted and multilayer SBMs) in Chapters 4–6.



(a) Generic SBM (b) Assortative (c) Disassortative (d) Core–periphery (e) Hierarchical

Figure 2.2: (a) Visualisation of edge-propensity parameters for a generic SBM with 3 blocks. (b)–(e) Specific types of structure that can be encoded and inferred with SBMs. (The hierarchical structure in panel (e) is also assortative, just more complicated than the one from panel (b).)

Basic variants

SBMs have long been studied in the social sciences [5]. However, because traditional SBMs produce networks with Poisson degree distributions, they tend to provide poor fits to real-world networks, whose degree distributions often have long tails. A key

breakthrough came with a 2011 paper by Karrer and Newman [76], who proposed a *degree-corrected* SBM that can fit arbitrary degree distributions. In this section, we present two versions of their model, which use either Bernoulli or Poisson random variables to sample edges between nodes.

In basic variants of degree-corrected SBMs, there are three sets of parameters (the number K of blocks is assumed to be fixed): a vector $\boldsymbol{g}$ of block assignments, so that g_i gives the label of node i; a vector $\boldsymbol{\sigma} = (\sigma_i)_{1 \leq i \leq N}$, where σ_i controls the expected degree of node i; and a $K \times K$ matrix $\boldsymbol{\Theta} = (\theta_{rs})_{1 \leq r,s \leq K}$, where θ_{rs} controls the propensity of nodes in block r to form edges with nodes in block s. Throughout this thesis, we occasionally refer to a pair of blocks (r, s) as an *edge bundle* [3]. Each edge bundle (r, s) is described by a corresponding edge-propensity parameter θ_{rs}.

Given two nodes i and j, the number of edges between i and j in the forward model is a Bernoulli or Poisson random variable with mean $\pi_{ij} = \sigma_i \sigma_j \theta_{g_i g_j}$. (Non-degree-corrected models omit the factors σ_i and σ_j, so that $\pi_{ij} = \theta_{g_i g_j}$.) We also need to specify a normalisation for the quantities σ_i; the choice employed in [76] is to set

$$\sum_{i=1}^{N} \sigma_i \delta(g_i, r) = 1 \text{ for } r \in \{1, \ldots, K\}\,, \tag{2.11}$$

where $\delta(\cdot, \cdot)$ is the Kronecker delta. (Other normalisations are also possible.)

Virtually all existing SBMs assume that edges are sampled independently according to the probabilities π_{ij} to generate a network. Thus, for a Bernoulli model, the probability of sampling a particular adjacency matrix $\boldsymbol{A}$ is

$$\mathbb{P}(\boldsymbol{A}|\boldsymbol{g}, \boldsymbol{\sigma}, \boldsymbol{\Theta}) = \prod_{1 \leq i \leq j \leq N} (\sigma_i \sigma_j \theta_{g_i g_j})^{A_{ij}} (1 - \sigma_i \sigma_j \theta_{g_i g_j})^{1 - A_{ij}}\,. \tag{2.12}$$

Similarly, using a Poisson model, we have

$$\mathbb{P}(\boldsymbol{A}|\boldsymbol{g}, \boldsymbol{\sigma}, \boldsymbol{\Theta}) = \prod_{1 \leq i \leq j \leq N} \frac{(\sigma_i \sigma_j \theta_{g_i g_j})^{A_{ij}} e^{-\sigma_i \sigma_j \theta_{g_i g_j}}}{(A_{ij})!}\,. \tag{2.13}$$

In the regime of sparse networks (see Section 2.1), Poisson models are approximately the same as Bernoulli models [122, 171]. Inference using Poisson SBMs turns out to be simpler, which is why they tend to be the preferred model in practice.[5]

In the simplest type of SBM inference, one seeks parameter values $\boldsymbol{g}$, $\boldsymbol{\sigma}$, and $\boldsymbol{\Theta}$ that maximise the log-likelihood $\mathcal{L} = \log \mathbb{P}(\boldsymbol{A}|\boldsymbol{g}, \boldsymbol{\sigma}, \boldsymbol{\Theta})$ for a given network with adjacency matrix $\boldsymbol{A}$. For the Poisson model, finding maximum-likelihood (ML) estimates of $\boldsymbol{\sigma}$

[5]However, for the correlated multilayer SBMs that we consider in Chapter 6, Bernoulli models turn out to be simpler.

and $\boldsymbol{\Theta}$ turns out to be straightforward when $\boldsymbol{g}$ is fixed. With the normalisation (2.11), Karrer and Newman showed [76] that these estimates are

$$\widehat{\sigma}_i = \frac{d_i}{\kappa_r}\,, \tag{2.14}$$

$$\widehat{\theta}_{rs} = m_{rs}\,, \tag{2.15}$$

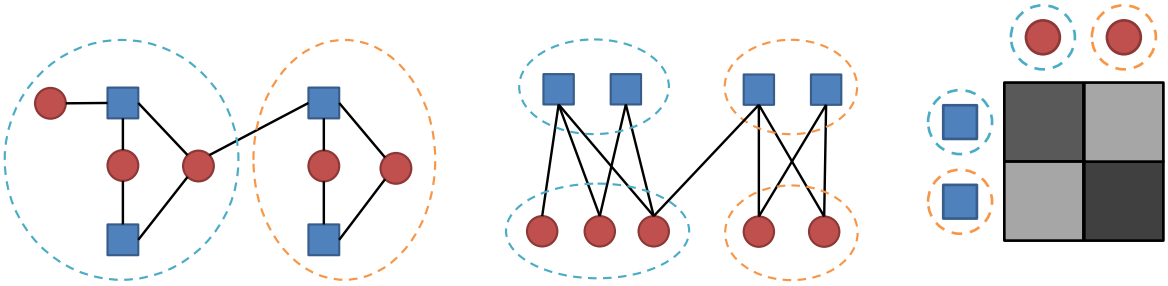
where d_i is the degree of node i, the quantity $\kappa_r = \sum_{i=1}^{N} d_i \delta(g_i, r)$ is the sum of degrees of nodes in block r, and $m_{rs} = \sum_{i,j=1}^{N} A_{ij}\delta(g_i, r)\delta(g_j, s)$ is the number of edges between blocks r and s (or twice the number of edges if $r = s$). Because of equation (2.14), some papers [109] do not use the additional parameters σ_i to control expected degrees; instead, they let the edge probabilities $\pi_{ij} = d_i d_j \theta_{g_i g_j}$ be proportional to the actual degrees, d_i and d_j, and they infer only the parameters $\boldsymbol{\theta}$ and $\boldsymbol{g}$.

SBMs for bipartite networks

It is possible to apply standard SBMs to bipartite networks. A successful inference algorithm should be able to determine that there are two sets of nodes that are perfectly disassortative, meaning that there are no edges between nodes in these sets. However, it is usually more efficient to use an algorithm that knows about this bipartite structure a priori. We follow the approach proposed by Larremore and collaborators in [88], where they extended the degree-corrected Poisson SBM from [76] to a bipartite setting. However, we describe a Bernoulli model instead, as we later use this model for annotated-networks inference in Chapter 4.

Larremore et al. [88] proposed an SBM in which blocks only include nodes of one type; for us, that means that each block contains either only customers or only products.[6] This may appear to depart significantly from community detection in bipartite networks, where (by construction) one detects mixed communities of both customers and products. However, as illustrated in Figure 2.3, one can go from a set of mixed communities to two sets of blocks that separate customers from products. If the original communities correspond to sets of nodes that are densely connected internally and sparsely connected to other sets, then the resulting bipartite SBM will also appear assortative, with larger edge probabilities on the diagonal than off the diagonal (see Figure 2.3(c)).

[6] One benefit of assigning nodes of different types to different blocks is that it makes the model easier to write down. With a mixed approach, for any pair of blocks (r, s), one has to specify two edge propensities: one that controls the probability of customers in block r to connect to products in block s, and one that controls the probability of customers in block s to connect to products in block r.



(a) Mixed blocks (b) Separate blocks (c) Bipartite SBM

Figure 2.3: There are different ways to characterise the structure of a bipartite network. One can assign nodes either to (a) mixed communities of customers and products, with a large number of edges within these communities; or to (b) separate blocks of customers and products, with a larger number of edges between some pairs of blocks. The network structure in panels (a) and (b) is identical. (c) Edge-propensity parameters of a bipartite SBM describing the network structure in (b). Note that customer block 1 connects preferentially to product block 1, and customer block 2 connects preferentially to product block 2.

We now write down the likelihood for a bipartite Bernoulli SBM. Let K_C and K_P denote the number of customer and product blocks. Given block assignments $\boldsymbol{g}$ (which include both customer assignments $\boldsymbol{g}_C$ and product assignments $\boldsymbol{g}_P$) and connection probabilities $\boldsymbol{\Theta}$, we generate network edges using independent Bernoulli trials with the following probabilities:

$$\pi_{cp} = \begin{cases} d_c d_p \theta_{g_c g_p} \,, & \text{if } c \in \mathcal{N}_C \text{ and } p \in \mathcal{N}_P \,, \\ 0 \,, & \text{if } c, p \in \mathcal{N}_C \text{ or } c, p \in \mathcal{N}_P \,. \end{cases} \tag{2.16}$$

Note that this model respects the bipartite structure of a network, as it never places an edge between two customer nodes or between two product nodes. Because of the independence assumption, the overall probability that this model generates a particular adjacency matrix $\boldsymbol{A}$ is a product over all pairs of customers and products:

$$\mathbb{P}(\boldsymbol{A}|\boldsymbol{\Theta}, \boldsymbol{g}) = \prod_{\substack{c \in \mathcal{N}_C, \\ p \in \mathcal{N}_P}} \pi_{cp}^{A_{cp}} (1 - \pi_{cp})^{1 - A_{cp}} = \prod_{\substack{c \in \mathcal{N}_C, \\ p \in \mathcal{N}_P}} \left(d_c d_p \theta_{g_c g_p} \right)^{A_{cp}} \left(1 - d_c d_p \theta_{g_c g_p} \right)^{1 - A_{cp}} .$$

As in the non-bipartite setting, one can maximise this likelihood with respect to the parameters $\boldsymbol{g}$ and $\boldsymbol{\Theta}$ to fit the model to a given network with adjacency matrix $\boldsymbol{A}$.

Inference methods

SBMs can be used either as forward models for generating networks or to examine the structure of a given network. For the latter task, one can determine parameters $\boldsymbol{g}$ and

$\boldsymbol{\Theta}$ that maximise the likelihood function (this is the method of maximum-likelihood estimation) or parameters that correspond to the mode of a posterior distribution that one obtains after imposing some prior distributions on these parameters (this is the method of maximum-a-posteriori estimation). Intuitively, in both cases, one seeks an SBM that provides the best fit to the data. Proposed methods for maximising the likelihood or posterior probability for specific types of SBMs include heuristic algorithms [76], various types of expectation-maximisation (EM) algorithms [34, 52], and belief propagation (known as the "cavity method" in the physics literature [97]) [101, 109]. Instead of getting point estimates of the parameters, it is also possible (although often more difficult) to approximate their entire posterior distribution. In this category of inference algorithms, we highlight a Monte Carlo algorithm by Peixoto [114] and various extensions in subsequent papers [65, 115, 116, 119, 120]. An alternative to Monte-Carlo sampling is offered by variational Bayesian methods, such as the one in [3], in which one approximates the posterior distribution as a product of simpler distributions.

We discuss several of these methods in Chapter 4, where we investigate the structure of annotated networks. In particular, we discuss extensions of the EM algorithm from [109] (which also uses belief propagation in one of the inference steps) and of the variational Bayesian algorithm for weighted networks from [3].

Number of blocks

Throughout this section, we have assumed that we are given the number of blocks K (or K_C and K_P for a bipartite network). When studying empirical networks with unknown mesoscale structure, the correct value of K is typically unknown, so an important question is how to determine appropriate values from the data. (This is also an important question for other forms of data clustering, such as for selecting the number of clusters for the K-means algorithm.)

Simply comparing likelihoods or posterior probabilities that one obtains for different values of K is ineffective, as models with a larger number of parameters always provide better fits to data. The Akaike information criterion (AIC) and the Bayesian information criterion (BIC), which incorporate penalty terms for the number of parameters in a model, are two popular methods for statistical model selection. However, due to assumptions that are unrealistic for network models, they tend to work poorly for selecting the number of blocks in SBMs [118, 163]. Alternative approaches include ones based on Bayes factors [3], minimum-description length [114], and full Bayesian inference [102, 129]. We give a detailed discussion of the Bayesian approach from [129]

in Chapter 4 as part of our investigation of annotated networks. Rather than taking a sequential approach, in which one determines K as part of a separate step from inferring the other SBM parameters, it is also possible to formulate a non-parametric approach that combines the two steps [115].

2.4.3 Modularity Versus SBMs for Community Detection

Both modularity maximisation and SBM inference have been used successfully to analyse the structure of real networks [139]. Each of these methods has strengths and weaknesses, and being aware of what these are is important for choosing which approach to use for a given problem.

Due to its simplicity, modularity maximisation remains a popular choice among users of community-detection algorithms. Fast implementations, which can handle networks with tens of millions of nodes and billions of edges, are readily available in many popular programming languages including Python, Java, C, and others. (This is important for our industrial partners, as they handle data sets with millions of customers and tens of thousands of products.) With modularity maximisation, the number of communities (which is typically unknown) does not need to be specified as an algorithm input. Another advantage of modularity is that it can be extended relatively easily to weighted, signed, or multilayer networks, to settings in which a non-standard null model is appropriate, and to other more complicated scenarios. Shortcomings of modularity are well-documented, and they include the well-known resolution limit [47] (which is especially pronounced in networks with a broad distribution of community sizes [86]), degeneracies of the modularity landscape [56] (meaning that one can obtain markedly different partitions of a network into communities that have comparable modularity values), and counterintuitive properties such as the fact that one can obtain large-modularity partitions for trees and other networks with no natural community structure [10].

SBMs have many advantages, and they have been gaining steadily in popularity. They can be used to infer mesoscale structures that are much more general than those detected by maximising modularity. They do not suffer from the resolution limits highlighted in [47] (although some SBMs have other types of resolution limits [115]). The statistical approach makes it possible to estimate the entire posterior distribution of block assignments, rather than return a single partition. Model-selection techniques make it possible to compare different models (e.g., with overlapping versus non-overlapping blocks [117]) and select the one best explained by the underlying data; this is not something that one can do easily with modularity-based methods. One

limitation of SBMs is the assumption that, degree-correction aside, nodes in the same block are stochastically equivalent. Also, edges are assumed to be independent, whereas triangles and other small-scale structures are common in empirical networks. Another drawback of SBMs is that they are difficult to generalise to networks with nonstandard features, including weighted, multilayer, and spatial networks. From a practical standpoint, algorithms based on SBM inference are substantially slower than the fastest modularity-maximisation algorithms, and they require more expert knowledge to use.

2.5 Validation

2.5.1 Measures of Partition Similarity

When studying mesoscale structures in a network, it is often useful to compare different ways of partitioning its set of nodes. For example, we may wish to compare partitions from different runs of a community-detection heuristic (e.g., the GenLouvain algorithm), partitions obtained using different values of the resolution and interlayer coupling parameters, or partitions obtained for different modelling choices (e.g., using different edge weights). It is also useful to be able to compare communities detected algorithmically with partitions based on metadata.[7] When testing algorithms on synthetic networks with known community structure, it is essential to be able to quantify how well a particular algorithm recovered the true structure of an input network.

There are many ways of measuring pairwise similarity between distinct partitions of the same set of objects (in our case, these objects are nodes of a given network) [159]. The most common similarity scores can be divided broadly into two classes: those based on pair counting [151], and those based on information theory [156]. Among the former, we mention the Rand index, the Jaccard index, and their adjusted variants. Among the latter, common choices include mutual information (with its various normalisations) and variation of information (VI), which has the advantage of being a metric in the mathematical sense. Throughout this thesis, we use *normalised mutual information* (NMI) and (to a lesser extent) *adjusted mutual information* (AMI) to measure partition similarity.

NMI is commonly used in network science to assess the performance of community-detection algorithms on benchmark networks [85]. It calculates the amount of infor-

[7] As argued in [113], one needs to be cautious about treating existing metadata as "ground truth" that should be recovered by community-detection algorithms. We return to this issue in Chapter 4.

mation common to two partitions relative to the total information in both. The given partitions, which we call X and Y, are used to estimate the empirical probability distributions $\mathbb{P}(X = x)$, $\mathbb{P}(Y = y)$, and $\mathbb{P}(X = x, Y = y)$ (or $\mathbb{P}(X)$, $\mathbb{P}(Y)$, $\mathbb{P}(X, Y)$ for short). More precisely, the quantity $\mathbb{P}(X = x, Y = y)$ is the joint probability that an object is classified in subset x in partition X and in subset y in partition Y, and $\mathbb{P}(X = x)$ and $\mathbb{P}(Y = y)$ are the corresponding marginal probabilities. Additionally, let

$$
\begin{aligned}
I(X, Y) &= \sum_{x \in X, y \in Y} \mathbb{P}(X, Y) \log \frac{\mathbb{P}(X, Y)}{\mathbb{P}(X)\mathbb{P}(Y)}\,, \\
H(X) &= -\sum_{x \in X} \mathbb{P}(X) \log \mathbb{P}(X)\,, \\
H(Y) &= -\sum_{y \in Y} \mathbb{P}(Y) \log \mathbb{P}(Y)\,,
\end{aligned}
$$

denote, respectively, the mutual information between partitions X and Y, and their respective individual entropies. In this thesis, we use the version of NMI given by

$$
\mathrm{NMI}(X, Y) = \frac{2I(X, Y)}{H(X) + H(Y)}\,. \tag{2.17}
$$

Defined in this way, NMI takes values in the interval $[0, 1]$. A value of 0 indicates that X and Y are independent (so that knowing one partition provides no information on the other), whereas a value of 1 indicates that one can completely determine one partition from the other one. There are several other normalised variants of mutual information that are commonly used in the literature [156], but we use the expression (2.17) throughout this thesis.

NMI tends to overstate similarities between partitions that contain a large number of subsets, and it is sometimes useful to use versions of mutual information that are corrected for chance (although these are generally much slower to compute). By analogy to the adjusted Rand index [66], Vinh et al. [156] defined the adjusted mutual information (AMI) between two partitions X and Y as

$$
\mathrm{AMI}(X, Y) = \frac{\mathrm{NMI}(X, Y) - \mathbb{E}[\mathrm{NMI}(\widetilde{X}, \widetilde{Y})]}{1 - \mathbb{E}[\mathrm{NMI}(\widetilde{X}, \widetilde{Y})]}\,. \tag{2.18}
$$

In this expression, $\mathbb{E}[\mathrm{NMI}(\widetilde{X}, \widetilde{Y})]$ denotes the expected NMI between two partitions $\widetilde{X}$ and $\widetilde{Y}$ that are random permutations of the original partitions X and Y. (Thus, the permuted partitions preserve the number of objects in each subset of the original partitions.) Using the version of NMI from equation (2.17), we have

$$
\mathrm{AMI}(X, Y) = \frac{I(X, Y) - \mathbb{E}[I(\widetilde{X}, \widetilde{Y})]}{(H(X) + H(Y))/2 - \mathbb{E}[I(\widetilde{X}, \widetilde{Y})]}\,. \tag{2.19}
$$

The expected mutual information $\mathbb{E}[I(\widetilde{X}, \widetilde{Y})]$ can be calculated as described in [156], without actually applying random permutations to generate $\widetilde{X}$ and $\widetilde{Y}$. Note that the AMI is equal to 1 for two equivalent partitions X and Y, and it is equal to 0 if $\mathrm{NMI}(X, Y)$ is equal to its expectation under the permutation null model.

When measuring similarity between a pair of multilayer partitions, which are two-dimensional rather than one-dimensional, there are two common ways to proceed. One approach is to unravel the two-dimensional partitions into one-dimensional vectors and compute similarity as usual. We choose a second approach, which was also proposed in [17]. Let $\boldsymbol{g}_1$ and $\boldsymbol{g}_2$ be two partitions of a network with T layers, and let $\boldsymbol{g}_1^t$ and $\boldsymbol{g}_2^t$ denote the induced partitions in layer t. Given a similarity function $\mathcal{S}(\cdot, \cdot)$ that takes as inputs two vectors, we can define a layer-averaged similarity score between $\boldsymbol{g}_1$ and $\boldsymbol{g}_2$:

$$\langle \mathcal{S}(\boldsymbol{g}_1, \boldsymbol{g}_2) \rangle = \frac{1}{T} \sum_{t=1}^{T} \mathcal{S}(\boldsymbol{g}_1^t, \boldsymbol{g}_2^t) \,. \tag{2.20}$$

Choosing $\mathcal{S}$ to be the NMI from equation (2.17), the formula (2.20) gives a way of calculating layer-averaged NMI between two multilayer partitions. (Layer-averaged AMI is defined analogously.) We use these similarity scores in Chapter 5 for some numerical experiments on multilayer networks.

2.5.2 Edge Prediction

Link prediction or edge prediction is a process in which one uses a model (such as an SBM) to assign probabilities to unobserved edges (i.e., node pairs (i, j) for which the entry A_{ij} of the adjacency matrix is unknown) or to missing edges (i.e., node pairs (i, j) for which $A_{ij} = 0$) [92]. All pairs of nodes whose probability of interaction is at least as large as some specified threshold are then considered to have edges between them, whereas those interactions with probabilities below the threshold are regarded as non-edges. (Although less commonly, it is also possible to use this process to delete spurious edges from networks.) Edge prediction is useful for filling in incomplete data sets, such as protein-interaction networks, in which edges are often established as a result of costly experiments, or terrorist-association networks, which are typically constructed based on partial knowledge [28]. In the context of bipartite consumer–product networks (or any other type of user–item network), edge-prediction techniques provide candidates for personalised product recommendations.

A sophisticated approach to edge prediction entails prescribing a second generative model for missing (and possibly spurious) edges and then inferring the parameters of this model from the data in a manner similar to how one performs community

detection using SBMs [118]. There are also simpler approaches, such as ones that are based on fitting generative models to networks, including various types of SBMs [28, 34, 57, 58]. For example, suppose that we have inferred some block structure $\boldsymbol{g}$ and edge propensities $\boldsymbol{\Theta}$ for a monolayer network with adjacency matrix $\boldsymbol{A}$ using a degree-corrected Bernoulli SBM.[8] For any two nodes i and j, the probability that they are adjacent according to the model is

$$\mathbb{P}(A_{ij} = 1) = d_i d_j \theta_{g_i g_j} \,. \tag{2.21}$$

The pairs (i, j) for which these probabilities are large, but which are not adjacent in the actual network (i.e., with $A_{ij} = 0$), would then produce a list of likely candidates for missing edges. Similarly, pairs (i, j) for which these probabilities are small but which are adjacent in the given network (i.e., with $A_{ij} = 1$) are perhaps spurious edges.

Edge prediction can be used to validate models. A common approach is cross validation, which consists of using part of a network data set to fit a model (e.g., an SBM) and then using that model to predict edges and non-edges in the unseen data. We perform cross-validation in this way in Chapter 6. For our application specifically, we can use edge prediction to rank customers based on their likelihood of purchasing a given product. In Section 3.3.2, we use data from a past promotional campaign to test whether this ranking is predictive of which customers are more likely to redeem a coupon.

[8] It is also possible to determine $\boldsymbol{g}$ using a method other than an SBM (e.g., by maximising modularity) and only use an SBM to fit edge-propensity parameters and to assign probabilities to edges.

Chapter 3

Product-Purchase Networks

3.1 Data

3.1.1 Transaction Data

Our industrial collaborator, dunnhumby, provided access to "pseudonymised" transaction data from stores belonging to a major retailer in the UK. The data was pseudonymised for general research purposes (i.e., by replacing personally identifiable information with numeric IDs), such that no individual shoppers can be identified. Groceries are typically purchased at a household level, rather than an individual customer level. We use household IDs assigned in the database to track purchases made by the same household over time. For convenience, throughout the rest of this thesis, we apply the term "customer" to each of these unique household IDs.

The transaction data that we have access to covers the period from 1 March 2013 to 30 May 2016 and includes stores of different sizes in urban and suburban areas. Only purchases in which customers scanned their loyalty card are part of this database. For each store, the transaction data consists of a unique household ID, a basket ID that can be used to identify items bought on the same shopping trip, the date when the purchase was made, the product (also stored as a unique ID) and the number of items purchased, the regular price of the items, and the "net price " (i.e., the amount spent by the customer, which includes any discounts). In Table 3.1, we illustrate the structure of these data using some fictional transactions.

The shopping data sets that we use in our case studies throughout this thesis are inherently incomplete. First, many consumers shop in more than one supermarket chain, but we only have access to data from one of these retailers. Second, we only analyse transactions linked to a loyalty card, meaning that any customers without such a card are missing altogether from our data. Inclination to sign up for loyalty

programmes varies across consumer types (for instance, price-sensitive customers may be more likely to sign up), so this introduces certain biases in the data that is available. Third, even consumers who are frequent loyalty card users make occasional purchases without scanning their card. This happens, for example, because they forgot the card at home or because they do not find it worthwhile to take it out when they are buying a small number of items. (It is worth noting, however, that for the largest store format that we describe in Section 3.2.3, a significant majority of purchases link back to a loyalty card.) Finally, with the exception of Section 3.3.2, all the other case studies in this thesis use transaction data from individual stores. (We made this choice to obtain networks of manageable sizes.) As a result, findings from these restricted data sets cannot be generalised to all UK customers. Despite all of these limitations, we believe that some of our qualitative findings — such as the fact that price-sensitive customers buy more frozen items than upscale customers — hold more broadly.

Table 3.1: Illustration of the structure of our transaction data

Store ID	Household ID	Basket ID	Date	Product	Quantity	Price	Net Price
123	12345	99999	01-Mar-13	Oranges	3	£0.60	£0.60
123	12345	99999	01-Mar-13	Yoghurt	1	£2.00	£1.50
123	12345	78987	05-Mar-13	Frozen pizza	2	£5.00	£4.00
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

3.1.2 Customer and Product Metadata

In addition to the timestamped transaction data, there is additional information about customers and products, which we use both to improve our community-detection methods and for validation purposes.

In retail, it is common to organise products in a hierarchy, where similar products are grouped into increasingly generic categories (see Table 3.2). Products that are close together in such a hierarchy are typically sold next to each other in a store. At the lowest hierarchical level, each unique bar code corresponds to a different product, so the same food items in different size packaging or different flavours are treated as separate products. The next three levels of product categories go from more specific ("Level 1") to more generic ("Level 3"). We use this product hierarchy both in Chapter 4, where we perform inference in annotated networks, and in Chapter 5, where we perform community detection in multilevel networks. Additionally, at various points,

we compare Level 3 product categories to algorithmically-detected communities of products to determine the composition of the latter.

Table 3.2: Example hierarchy for several products

Product	Level 1	Level 2	Level 3
Blueberries 150 g	Blueberries	Chilled fruit	Produce
Organic strawberries 300 g	Strawberries	Chilled fruit	Produce
Brie 200 g	Soft cheese	Continental cheese	Cabinet provisions
Farmhouse loaf 350 g	White bread	Fresh bakery bread	Bakery
Salmon fillets 200 g	Salmon and tuna	Pre-packed fish	Meat
Premium ice cream 500 ml	Luxury and sorbet	Ice cream	Frozen foods

In terms of customers, existing "segmentations" (i.e., categories) of customers provide useful baseline groups to compare to communities that we obtain algorithmically. The main segmentation that we were provided for this purpose divides customers into categories based on purchasing habits (value items or premium items, exotic or mainstream, fresh or frozen, etc.). Purchasing segments come in three hierarchical levels, which we call `purchasing-low` (13 possible labels), `purchasing-medium` (6 possible labels), and `purchasing-high` (3 possible levels, corresponding roughly to price-sensitive, mid-range, and high-end consumers). A second segmentation, based on how frequently people shop and how much they spend in each trip, is useful for generating a restricted data set, from which we remove customers with relatively few purchases (see Section 3.2.3). We have also considered a third customer segmentation, based on demographics such as age and household composition (student or working, single or couple, having younger or older children, etc.). However, we have found that this data is relatively sparse (as it is provided voluntarily when customers sign up to a loyalty programme), and it does not appear to be correlated significantly with our detected communities. Therefore, we do not analyse this segmentation any further.

3.2 Network Construction

3.2.1 Time Windows

We aggregate transactions over fixed time windows to obtain bipartite networks of customers and products. We analyse the resulting structures both as single-layer networks (Section 3.3.2 and Chapter 4) and as multilayer networks (Chapter 5 and

Chapter 6). The length of each time window should be long enough for the resulting network to be representative of longer-term shopping habits, but short enough to avoid seasonal effects and customers changing their behaviour over time. We believe that using three-month time periods is sensible for our data, though this choice has not been optimised.

3.2.2 Edge Weights

In a given time window, edges connect customers to products that they have bought during that time. Edges corresponding to purchases that are more frequent or that involve a larger number of items should have higher weights, indicating stronger affinities between the respective customers and products. We implement three different weighting schemes, which we summarise in Table 3.3.

Table 3.3: Three ways to calculate the weight from customer c to product p

Name	Description
Item count	total number of items of product p purchased by customer c
Item penetration	fraction of all the items bought by customer c that were product p
Basket penetration	fraction of all baskets of customer c that include product p

There are various ways to motivate these weightings. Item count is perhaps the most obvious choice, but it tends to give higher weights to customers that shop more frequently, even though they are not necessarily more loyal to specific products than the less frequent shoppers. The other two weightings address this issue in different ways. Item penetration overweights products that are bought in large quantities, whereas basket penetration is larger for products that are purchased regularly. Other choices of weights are also possible. For instance, the total item count for a specific product can be scaled by the total item count in the associated product category to assess loyalty to a specific brand over similar alternatives. In our work, we use exclusively item-penetration and basket-penetration weightings to analyse consumer–product networks.

Whenever possible, we apply community-detection algorithms to weighted networks, as they include additional information compared to their unweighted counterparts. However, many existing methods, especially many of the ones that are based on SBMs, work only on unweighted networks. Therefore, at various points in Chapters

4–6, we can only analyse unweighted consumer–product networks. In those situations, we apply percentile thresholds to go from a weighted network to an unweighted one. For example, using a 50^{th}-percentile threshold removes half of the edges (those in the bottom half of the weight distribution) from the original network. Although the thresholding process entails information loss, it nevertheless results in communities that correlate strongly to communities in the original weighted network [162].

3.2.3 Data Sets

To construct data sets that are manageable in size, we consider transactions from three individual stores. (The only exception is our work from Section 3.3.2, where we analyse a network with 100,000 customers spread across the UK.) We summarise these data sets in Table 3.4. In our analysis, we include a smaller, convenience store (hereafter store `C1`) located in a suburban area and two larger supermarkets (hereafter stores `S1` and `S2`) on the outskirts of medium-sized towns. Because of our focus on individual stores, any conclusions regarding consumer behaviour drawn from our analysis do not necessarily apply to the whole of the UK; our goal is not to examine issues related to consumer behaviour on a representative customer sample, but rather to illustrate the methods that we present in this thesis on a real data set.

For store `C1`, we include all transactions in which customers used a loyalty card, and we consider the two-year time period from 1 March 2013 to 31 March 2015. We only include purchases of grocery items (primarily food products and a small number of non-food essentials), as these products are purchased repeatedly and thus yield more data for analysis. The resulting data set has 21,368 customers and 5,732 products, although time periods shorter than the two-year sample typically have a much smaller subset of these. One should also keep in mind that loyalty-card transaction data is a biased and incomplete subset of all purchases, and this is especially pronounced for smaller stores (as customers find it less worthwhile to take out their loyalty card when buying a small number of items).

Table 3.4: Summary of data sets: transactions from three stores

Store label	Size/type	Time period	Customers	Products	Notes
`C1`	Small/convenience	Mar '13 – Mar '15	21,368	5,732	unfiltered
`S1`	Large/supermarket	Mar '13 – May '16	853	3,705	filtered
`S2`	Large/supermarket	Mar '13 – May '16	1,462	4,950	filtered

When keeping all of the transactions in a store, there is a significant amount of noise due to infrequent shoppers and niche products that are only bought a handful of times. For example, more than 50% of customers who shop in store `C1` during the three-month time period July–September 2014 only made a single trip to this store during this period. Additionally, larger stores receive many more customers and contain a much wider range of products than smaller convenience stores, which yields much larger data sets. In light of these observations, we filter the data from stores `S1` and `S2` to include only the most loyal customers and products that are purchased with sufficient regularity. More precisely, for each of the stores `S1` and `S2`, we use an existing customer segmentation based on loyalty (see our discussion in Section 3.1.2) to obtain a sample of shoppers who are regular customers in those two stores throughout the entire time period from 1 March 2013 to 31 May 2016; this retains slightly more than 1% of the initial number of customers. On the product side, we keep all products that sold at least once in every full month between March 2013 and May 2016; this choice reduces their numbers by about 75%. This process also excludes seasonal items sold around Christmas, Easter, and other yearly events.

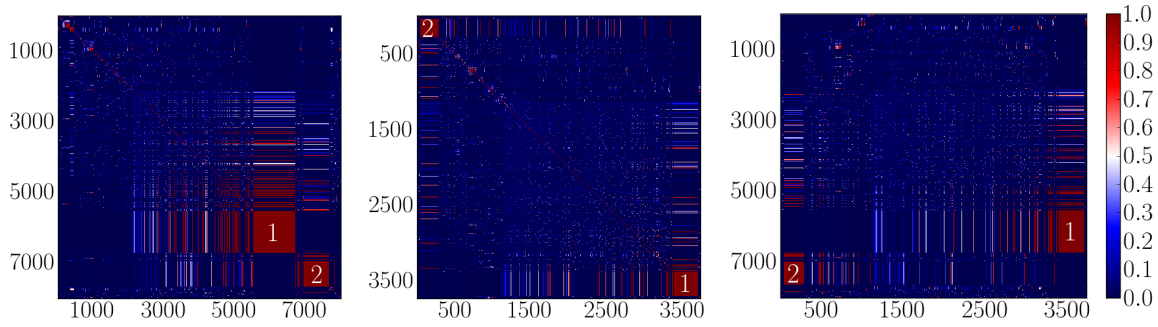
3.3 Exploratory Community Structure

3.3.1 Customer and Product Communities in a Small Store

In this section, we explore some community-detection results for store `C1` (see Section 3.2.3), which is a relatively small, suburban convenience store. Our goal is to show that by running even a basic version of a community-detection algorithm, we find sets of customers and products that have distinct characteristics.

We examine the 3-month time period between July and September 2014, which was chosen to avoid overlap with major holidays such as Christmas and Easter. We use item penetration weighting to generate a weighted network, and then we use modularity maximisation with the bipartite null model [11] and $\gamma = 1$ to detect communities (see Section 2.4.1). Rather than analyse the output of a single algorithm run, we perform modularity maximisation 100 times to generate an association matrix (see Section 2.4.1). In Figure 3.1, we show a visualisation of this association matrix. For clarity, we break down the matrix into three components: co-classification of customer nodes versus customer nodes, of product nodes versus other products, and of customers versus products. We have sorted the original matrix to get it close to a

block-diagonal structure,[1] with larger co-classification values within these blocks and smaller values outside of them.



(a) Customers vs. customers (b) Products vs. products (c) Customers vs. products

Figure 3.1: Association matrices, showing co-classification of (a) pairs of customer nodes; (b) pairs of product nodes; and (c) pairs of one customer node and one product node. Reordering rows and columns reveals two sets of nodes, which include both customers and products, that are co-classified in at least 93 of the 100 trials. (These sets of nodes are labelled with 1 and 2.)

As is visually apparent from Figure 3.1, many pairs of nodes are not robustly assigned to the same community across a majority of the runs. This is perhaps to be expected for this data set, in which the median customer shops just once during the 3-month time period that we used to construct the network. (This observation also motivated our construction of the more restricted data sets from Section 3.2.3.) However, there are two sets of nodes, which include both customers and products, that consistently belong to the same two communities in at least 93% of the runs. (In fact, almost all pairs of nodes in each of the two sets are co-classified in all trials.) We now provide brief characterisations of these two sets of customers and products.

In Tables 3.5 and 3.6, we give some summary statistics for customers and products in these two blocks of nodes relative to those in the rest of the network. Block 1 has customers who shop more often, buy more items each time, and spend more money compared to the average customer. Using the purchasing segmentations from Section 3.1.2, we find that these customers are overrepresented in categories associated to home cooking, healthy habits, and exotic tastes. Indeed, looking at the Level 3 (`L3`) categories of products in block 1, we find a larger proportion of produce, meat, and foreign cuisine relative to the whole network. From Table 3.6, we see that products

[1] We used MATLAB's `symrcm` function, an implementation of the reverse Cuthill–McKee algorithm, which permutes rows and columns of a sparse symmetric matrix to place larger values as close to the diagonal as possible (see `http://uk.mathworks.com/help/matlab/ref/symrcm.html`).

in block 1 are slightly cheaper than the mean product in the network. The mean number of items sold is larger for products in block 1 than for all products, although the pattern is reversed if we look at the median. Thus, block 1 includes both highly popular and niche products.

Table 3.5: Customer profiles

Property	All nodes	Block 1	Block 2
Number of baskets (mean)	3.0	3.9	3.5
Items per shop (mean)	6.3	10.0	5.2
Spend per shop (£, mean)	9.2	11.9	6.2

Table 3.6: Product profiles

Property	All nodes	Block 1	Block 2
Items sold (mean)	38.5	78.0	51.9
Items sold (median)	17.0	9.5	37.5
Unique customers (mean)	24.5	42.9	34.8
Unique customers (median)	11.0	7.0	28.0
Average price (£, mean)	2.8	2.5	1.9

Customers in block 2 have a different profile. Although they shop slightly more often than a typical customer, they tend to spend less money each time and buy fewer items. They are overrepresented in purchasing categories that are broadly associated to loyalty to well-known brands and to children's products. Compared to other customers, they buy more snacks, soft drinks, ready meals, and products displayed at aisle ends (which are usually on offer). As Table 3.6 indicates, these products are both cheaper and more popular (based on the number of items sold and the number of unique customers that buy them) compared to the entire sample of products in the network. Interestingly, if we exclude all discounted items from the transaction data before applying community detection, block 2 of customer and product nodes does not emerge as clearly; by contrast, block 1 is largely unaffected, suggesting that these customers are less likely to buy items that are on promotion.

These two consumer profiles — one corresponding to upscale customers who buy fresh ingredients for cooking at home and another corresponding to price-sensitive customers who buy ready meals and products on offer — emerge repeatedly in our examinations of consumer–product networks. This pattern occurs even when we look at different stores or different community-detection methods (e.g., SBMs for annotated networks in Chapter 4), and it also arises for both monolayer and multilayer networks

(e.g., temporal and multilevel structures in Chapter 5). The consistency of this pattern provides further evidence that the communities that we detect in these networks are meaningful.

3.3.2 Validation Using Promotional Campaign Data

In this section, we use data from a promotional campaign run by our industrial collaborators to assess the merits of community-detection techniques for product recommendations.

Campaign data

We have been provided access to data from a promotional campaign in which coupons for yoghurts from a particular brand (which, for convenience, we call brand X) were mailed to a sample of 100,000 customers across the UK. These customers were initially divided into six "cells" based on various characteristics. Cells 1–4 consist of customers who have bought yoghurts from brand X consistently throughout the 13-week training period. Cells 5 and 6 consist of customers whose purchase of yoghurt from brand X declined during the training period. One coupon, which we call `C1`, was mailed to customers in cells 1–4, and a second coupon, `C2`, was sent to customers in cells 5 and 6. (The promotion on coupon `C2` was slightly more generous than that of coupon `C1` to entice customers to return to the brand.) Each coupon could be redeemed for one of 7 products, most of which are different flavours of the same type of yoghurt.

As this is a campaign that has already ended, the data indicates, for each customer, whether they ended up using their coupon. Overall, about 25% of the 100,000 customers redeemed a coupon.

Customer ranking

Given these data, our goal is to propose a model that predicts, for each customer, whether they end up redeeming their coupon. Specifically, we do so by ranking customers based on their affinity to the products on promotion; if higher-ranked customers redeem more coupons than lower-ranked customers, then the corresponding ranking has predictive power, which suggests that the same approach may be useful to identify relevant customers for future campaigns.

We examine two rankings. One is a simple baseline that ranks customers based on how much they spent, during the whole training period, in the relevant Level 1

category that contains all 7 products included in the promotion.[2] The second ranking is based on network analysis: we construct a consumer–product network, we detect communities in this network, and we use edge prediction (see Section 2.5.2) to order customers from larger to smaller edge probabilities.

We construct a network using the 100,000 customers that were initially included in the campaign and all food products (approximately 28,000) available for purchase during the training period. We aggregate transactions over the whole 13-week training period using item-penetration weighting (see Section 3.2.2). The resulting network has about 21 million edges. We use the GenLouvain algorithm [73] with $\gamma = 1$ to detect communities in this weighted network. (We use the "default" resolution value $\gamma = 1$ to show that one obtains useful results even with a relatively unsophisticated approach. However, one would ideally apply one of the methods from Section 2.4.1 to optimise the choice of γ.) Using the resulting community labels $\boldsymbol{g}$, we fit a bipartite degree-corrected SBM (see Section 2.4.2) to estimate the edge-propensity parameters $\boldsymbol{\Theta} = (\theta_{rs})_{1 \leq r,s \leq K}$, where $K = 12$ is the number of communities of each type of node (i.e., customer or product) detected using modularity maximisation.[3] For each customer c in the sample and for each product p that is among the 7 products included in the promotion, we then use the bipartite, degree-corrected Bernoulli SBM from Section 2.4.2 to compute the probabilities

$$\mathbb{P}(A_{cp} = 1) = d_c d_p \theta_{g_c g_p} ,$$

where d_c and d_p denote node degrees. Finally, for each customer c, we calculate the maximum probability across the sample $\mathcal{P}$ of 7 products:

$$\pi_c = \max_{p \in \mathcal{P}} \left\{ d_c d_p \theta_{g_c g_p} \right\} .$$

We then rank customers in decreasing order of the probabilities π_c.

Results

We now have two ways of ranking customers based on their affinity to the products included in the campaign. We compare the two rankings using *gains charts* [2], which are commonly used in customer analytics to assess the performance of predictive

[2]This ranking was provided to us with the rest of the data set, and we were told explicitly to use it as a baseline to compare against more sophisticated rankings.

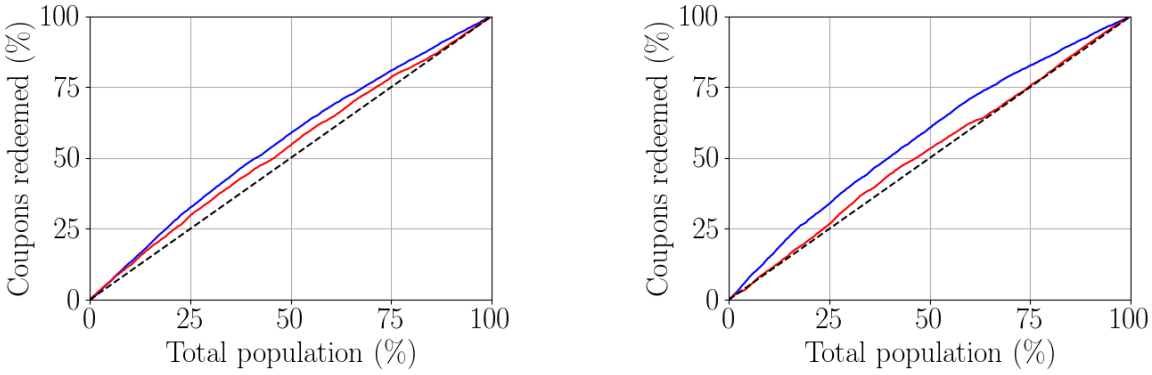
[3]One may ask why we did not fit a degree-corrected SBM directly to infer both a block structure $\boldsymbol{g}$ and the edge-propensity parameters $\boldsymbol{\Theta}$. One reason is that modularity maximisation is significantly faster than SBM inference on a network of the size that we consider here. A second reason is that incorporating weights is straightforward with modularity, but it is tricky with SBMs (see also our discussion in Chapter 4).

models.[4] In a gains chart, one plots the percentage of positive responses — in our case, the percentage of coupons redeemed — as a function of the population size, which is also expressed as a percentage. (Note that the dependent variable is divided by the total number of coupons redeemed in the given data set, rather than by the total number of customers.) In tracing this function, one adds individuals to an increasing population based on a given ranking, so one considers higher-ranked customers before lower-ranked ones. A ranking with no predictive power appears on a gains chart as a diagonal line corresponding to the identity function. A ranking that is perfectly predictive ranks all customers who have redeemed a coupon above those who have not redeemed a coupon. On a gains chart, perfect prediction corresponds to a piecewise linear function with two pieces: a line that goes from $(0\%, 0\%)$ to $(f\%, 100\%)$, where $f\%$ is the percentage of the population that redeemed a coupon; and a line that goes from $(f\%, 100\%)$ to $(100\%, 100\%)$.

In Figure 3.2, we show gains charts for coupon `C1` and the corresponding customer cells using both our ranking based on edge prediction and the baseline ranking provided with the campaign data. In all four panels, our model outperforms the baseline. Our ranking has similar performance across all four cells, whereas the spending-based ranking appears to perform slightly better for high spenders (cells 1 and 3) than for medium spenders (cells 2 and 4). Similarly, in Figure 3.3, we show gains charts for coupon `C2` and customer cells 5 and 6. Here, too, our ranking shows predictive power, whereas the performance of the baseline ranking is nearly indistinguishable from random.

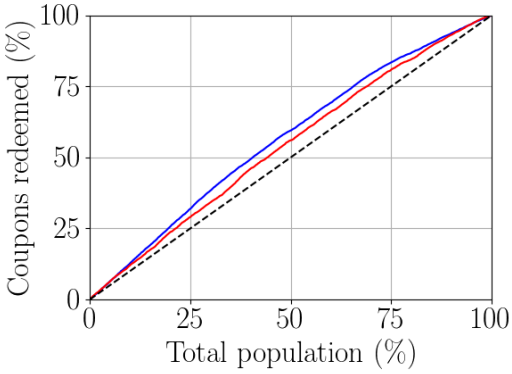
This case study illustrates that methods based on edge prediction in consumer–product networks can be effective at identifying relevant customers for a given set of products. An interesting finding is that this method of ranking customers appears to perform similarly for all six customer cells, whereas the predictive edge of the baseline ranking seems to vanish for customers with lower spending in the relevant category (i.e., customers in cells 5 and 6). We also emphasise that the community-detection step of our analysis is fairly unsophisticated (modularity maximisation in a weighted network with $\gamma = 1$) in comparison to the models that most researchers use for edge prediction in networks [92]. It is possible that more complicated methods for community detection (e.g., SBMs) would yield better predictive performance, although at an increased computational cost.

[4]Gains charts are similar to receiver-operating-characteristic (ROC) curves, which we use extensively in Chapter 6.

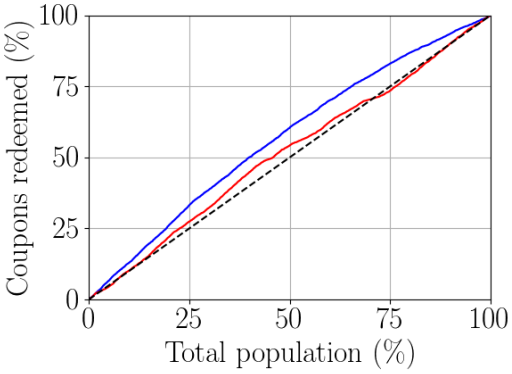


(a) Cell 1, coupon `C1`

(b) Cell 2, coupon `C1`

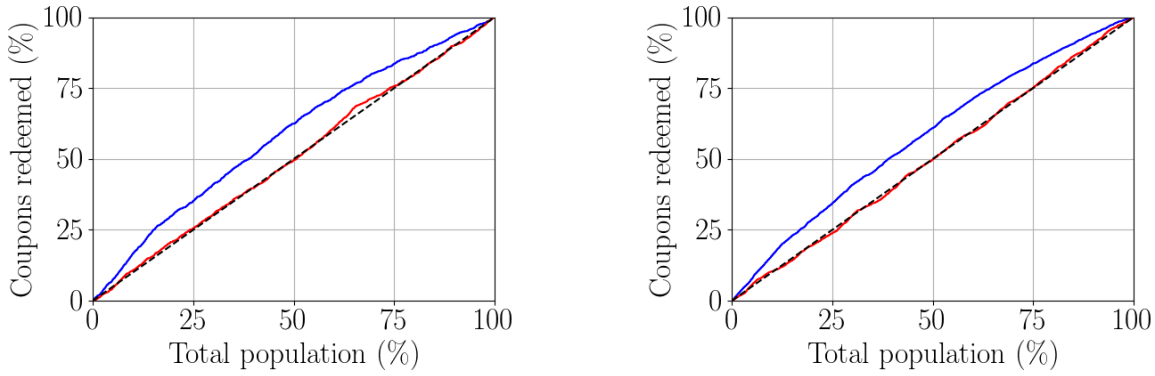


(c) Cell 3, coupon `C1`



(d) Cell 4, coupon `C1`

Figure 3.2: Gains charts for coupon `C1` and cells 1–4. Dashed diagonal lines indicate the expected performance of a random model. In all panels, our ranking based on edge prediction in networks (blue curves) has better predictive performance than a ranking based on customer spending (red curves).



(a) Cell 5, coupon `C2`

(b) Cell 6, coupon `C2`

Figure 3.3: Gains charts for coupon `C2` and cells 5 and 6. Dashed diagonal lines indicate the expected performance of a random model. In both panels, our ranking based on edge prediction in networks (blue curves) has better predictive performance than a ranking based on customer spending (red curves), which appears to be only marginally better than random.

Chapter 4

Community Detection in Annotated Bipartite Networks

4.1 Motivation

Many network data sets include additional information besides a list of (weighted or unweighted) edges, which is what is traditionally viewed as network data. For example, edges can be timestamped or they can represent different kinds of interactions, and both nodes and edges can have additional attributes or "labels" associated to them. Both edge and node *metadata* can be used to improve analyses of network structure (and also dynamics or function). The former can be used to construct multilayer networks (e.g., one can use edge timestamps to construct temporal networks), which we discuss further in Chapter 5. In this chapter, we study networks with node metadata, known as annotated networks in some previous work [65, 109].

In the context of community detection, node metadata is often used for validation. Researchers frequently treat (discrete) metadata as "ground truth" that they aim to recover using community-detection algorithms. When detected communities do not correlate well with metadata, this is often attributed to an algorithm or method failing to find the right structure [48]. A recent paper by Peel et al. [113] argued that treating metadata as ground truth is problematic in many cases. They indicated that there are several reasons other than poor algorithm performance for observing low correlation between detected communities and so-called ground truth: (1) the metadata are unrelated to the network structure; (2) the communities and the metadata capture different features of a network's mesoscale structure; or (3) the community structure (or other mesoscale structure) is sufficiently weak not to be detectable by any algorithm. Additionally, if one wants to treat metadata as ground truth, it is unclear how to proceed when there are multiple types of (possibly contradictory) metadata

associated to a set of nodes; picking a single set of metadata to use for validation and disregarding the rest does not seem like a great way to proceed. Finally, if one takes the view that some structure induced by metadata is the correct one, it is unclear why one would go through the trouble of performing community detection at all.

Instead of treating metadata as ground truth for community detection, one can incorporate it into a network model and use it to find better communities than using the network data alone. There are now several algorithms in the literature for performing community detection in annotated networks [30, 65, 109, 144, 161, 165, 169]. In this work, we extend the approach by Newman and Clauset [109], but we briefly summarise several other papers here. The method proposed by Hric et al. [65] is similar to the one of Newman and Clauset (both are based on SBMs), with one key difference: in the former, annotations are modelled as arising from a network's mesoscale structure, while in the latter mesoscale structure arises from node annotations. We believe that for our application, node metadata — such as product categories or prices — tend to influence network structure (i.e., what customers purchase) rather than the other way around, thus justifying our choice to extend the method from [109] over the one from [65]. Stanley et al. [144] developed an algorithm for fitting SBMs to networks with multiple continuous attributes, which they modelled as mixtures of multivariate normal random variables. Not all existing approaches are based on SBM inference. Yang et al. [165] used different generative models in which both network structure and the (binary-valued) annotations arise from latent community structure. In their approach, communities are assortative and overlapping, with pairs of nodes being more likely to form an edge if they share membership in multiple communities. Xie and Wang [161] recast a special case of the likelihood function from [109] as a modified modularity function that is a linear combination of standard modularity and a conditional-entropy term that relates annotations to community structure.[1] Another modularity-based approach was presented by Combe et al. in [30], who employed optimisation of a combination of standard Newman–Girvan modularity (see Chapter 2) and a modularity with a null model informed by (real-valued) annotations.

Setting aside specific choices of inference methods for annotated networks, there is a broader question about the role that metadata should play in analysing topological network characteristics such as community structure. Part of the original appeal of community detection was that one could obtain useful information about a system simply by studying the relevant network. (The most famous example of this is the

[1] Relating SBMs to modularity-like objective functions (and the insights gained in the process) form the topic of Chapter 5.

Karate Club network [110].) Thus, one view is that a network's connectivity patterns are paramount. At the other extreme, one could make no qualitative distinction between network data and node or link metadata; this is the view that Hric et al. take in [65]. One pitfall of this second approach is the possibility of adding so much metadata to a network that its topology ends up playing no role in the output of an inference algorithm. To mitigate this behaviour, some researchers have proposed objective functions with tunable parameters for controlling the importance of network topology versus annotations [42, 99]. (This parameter is akin to regularisation constants that appear in other types of optimisation problems.) We believe that the right approach depends on the specific application and on the strength of an underlying network's mesoscale structure. When this structure is strong (e.g., there are clearly-defined communities), annotations may not be necessary. (However, even in these "easy" scenarios, metadata can help resolve ambiguities when there are several options for a node's community assignment.) When this structure is weak, annotations are especially useful in guiding algorithms towards useful solutions, even beyond detectability thresholds [109].

Incorporating node metadata into product-purchase networks is useful for several reasons. First, doing so can help select between several competing divisions of a network into communities. Including node annotations pushes community-detection algorithms towards finding partitions that better align with some provided metadata, and this makes it easier to interpret the detected communities. Second, incorporating appropriate metadata allows one to incorporate additional domain-specific knowledge into a network model. In our simple network models from Chapter 3, any two products are considered equally distinct: black grapes are no more similar to white grapes than they are to ice cream. Using product categories as annotations on the product nodes, as we do in Section 4.4, introduces evaluations of similarity between products. (Another way to incorporate knowledge about product hierarchies into network structure is by using "multilevel networks", which we discuss in Chapter 5.) Third, when promoting a new product, communities detected using annotated networks help address the "cold-start" problem of recommender systems [170]. This problem refers to the fact that for a new product, one lacks the required information (i.e., customer purchases) to incorporate the product into a network and infer its community assignment that way. With annotations, metadata of that node (such as its product category) gives prior probabilities for it to belong to the existing product communities. One can use this information to identify the most likely customers to

buy that product, and these customers can then be offered coupons to encourage initial purchases.

In this chapter, we extend several tools for analysing annotated networks to bipartite networks. These include the inference algorithm from [109] for detecting communities in annotated networks, and its extension to weighted networks (Section 4.2); the Bayesian method from [129] for determining the number of blocks (i.e., communities) in a network (Section 4.3); and the statistical test from [113] for establishing whether some given metadata is informative of the structure of a network (Section 4.4.2). In Section 4.4, we use these methods to analyse shopping networks in which product nodes are annotated with their corresponding product category, and we evaluate the impact of these metadata on the inferred mesoscale structures.

4.2 Model and Statistical Inference

4.2.1 Unweighted Networks

In this section, we explain how to extend the approach proposed by Newman and Clauset [109] to bipartite networks.[2] This method is based on a degree-corrected Bernoulli SBM (see Section 2.4.2), modified to account for annotations. Although the derivation in this section allows both customer and product nodes to have associated metadata, we only impose annotations on the products for the applications in Section 4.4. The more general framework can easily handle this special case by giving all customer nodes the same, non-informative label.

Table 4.1 summarises the notation that we use in this section. The inputs to the inference algorithm are the rectangular adjacency matrix $\boldsymbol{A} = (A_{cp})_{c\in\mathcal{N}_C, p\in\mathcal{N}_P}$, the customer and product annotations $\boldsymbol{x}_C = (x_c)_{c\in\mathcal{N}_C}$ and $\boldsymbol{x}_P = (x_p)_{p\in\mathcal{N}_P}$, and the numbers of customer and product blocks K_C and K_P. (We discuss the task of selecting the number of blocks in Section 4.3.) There are three sets of parameters that define the annotated SBM, so the corresponding inference algorithm returns three sets of outputs. These are vectors $\boldsymbol{g}_C$ and $\boldsymbol{g}_P$ that indicate the block assignments of customer and product nodes, a $K_C \times K_P$ matrix $\boldsymbol{\Theta}$ that describes the propensities of nodes in different blocks to form edges, and matrices $\boldsymbol{\Gamma}_C$ and $\boldsymbol{\Gamma}_P$ that encode the a priori probabilities of nodes with different metadata labels to belong to different blocks.

Newman and Clauset [109] employed a maximum-likelihood (ML) approach to infer the parameters of their proposed annotated SBM. They applied an expectation-

[2]For concreteness, we identify the two types of nodes in these bipartite networks as "customers" and "products", but the derivations in this chapter apply to any bipartite network.

maximisation (EM) algorithm [40] to perform this maximisation in an iterative way, using belief propagation (BP) [59] for the expectation step of that algorithm. We now explain how to modify their approach to handle bipartite networks, noting that the most involved part of this process is the derivation of the BP equations.

Table 4.1: Summary of notation for unweighted, annotated SBMs.

	Notation	Meaning
Inputs	$\mathcal{N}_C$, $\mathcal{N}_P$	Sets of customer/product nodes
	$N_C = \lvert\mathcal{N}_C\rvert$, $N_P = \lvert\mathcal{N}_P\rvert$	Number of customer/product nodes
	$\boldsymbol{A} = (A_{cp})_{c\in\mathcal{N}_C, p\in\mathcal{N}_P}$	Network adjacency matrix
	$\boldsymbol{x}_C = (x_c)_{c\in\mathcal{N}_C}$, $\boldsymbol{x}_P = (x_p)_{p\in\mathcal{N}_P}$	Customer/product metadata labels
	$\mathcal{L}_C$, $\mathcal{L}_P$	Sets of customer/product labels
	$L_C = \lvert\mathcal{L}_C\rvert$, $L_P = \lvert\mathcal{L}_P\rvert$	Number of unique customer/product labels
	$\mathcal{K}_C$, $\mathcal{K}_P$	Sets of customer/product blocks
	$K_C = \lvert\mathcal{K}_C\rvert$, $K_P = \lvert\mathcal{K}_P\rvert$	Number of customer/product blocks
Param.	$\boldsymbol{g}_C$, $\boldsymbol{g}_P$	Vectors of block assignments
	$\boldsymbol{\Theta} = (\theta_{rs})$	Matrix of edge-propensity parameters
	$\boldsymbol{\Gamma}_C$, $\boldsymbol{\Gamma}_P$	Block-metadata probability matrices
Other	$m = \sum_{c\in\mathcal{N}_C, p\in\mathcal{N}_P} A_{cp}$	Number of edges
	d_c $(c\in\mathcal{N}_C)$, d_p $(p\in\mathcal{N}_P)$	Degrees of nodes c and p
	κ_r $(r\in\mathcal{K}_C)$, κ_s $(s\in\mathcal{K}_P)$	Total degree of nodes in blocks r and s

Deriving the likelihood

Let us ignore annotations for the moment. Using a degree-corrected Bernoulli SBM for bipartite networks (see Section 2.4.2), the likelihood of an adjacency matrix $\boldsymbol{A}$ given edge-propensity parameters $\boldsymbol{\Theta}$ and block assignments $\boldsymbol{g}$ (which include both customer assignments $\boldsymbol{g}_C$ and product assignments $\boldsymbol{g}_P$) is

$$\mathbb{P}(\boldsymbol{A}|\boldsymbol{\Theta},\boldsymbol{g}) = \prod_{\substack{c\in\mathcal{N}_C,\\ p\in\mathcal{N}_P}} \pi_{cp}^{A_{cp}}(1-\pi_{cp})^{1-A_{cp}} = \prod_{\substack{c\in\mathcal{N}_C,\\ p\in\mathcal{N}_P}} \left(d_c d_p \theta_{g_c g_p}\right)^{A_{cp}} \left(1-d_c d_p \theta_{g_c g_p}\right)^{1-A_{cp}}. \tag{4.1}$$

Where do annotations come in? Newman and Clauset [109] imposed a prior on $\boldsymbol{g}$ that depends on the node labels $\boldsymbol{x}$ and which favours block structures that better align with these annotations. We take the same approach, with minor adjustments to account for the fact that we have two sets of annotations corresponding to the two different node types (i.e., customers and products). Let γ_{rx} denote the prior probability

that a customer (product) node with label x is assigned to customer (product) block r. We normalise these probabilities by setting $\sum_{r\in\mathcal{K}_C}\gamma_{rx}=1$ for $x\in\mathcal{L}_C$ (customers) and $\sum_{s\in\mathcal{K}_P}\gamma_{sx}=1$ for $x\in\mathcal{L}_P$ (products). In the forward model, we generate a vector $\boldsymbol{g}$ by independently assigning each node to a block according to these probabilities. This process yields the following prior on $\boldsymbol{g}$:

$$\mathbb{P}(\boldsymbol{g}|\boldsymbol{\Gamma},\boldsymbol{x})=\prod_{c\in\mathcal{N}_C}\gamma_{g_c x_c}\prod_{p\in\mathcal{N}_P}\gamma_{g_p x_p}\,. \tag{4.2}$$

To efficiently maximise the likelihood (4.1) while accounting for the functional form of the $\boldsymbol{g}$-prior from (4.2), Newman and Clauset [109] used an EM algorithm [40] to maximise the quantity

$$\mathbb{P}(\boldsymbol{A}|\boldsymbol{\Theta},\boldsymbol{\Gamma},\boldsymbol{x})=\sum_{\boldsymbol{g}}\mathbb{P}(\boldsymbol{A}|\boldsymbol{\Theta},\boldsymbol{g})\mathbb{P}(\boldsymbol{g}|\boldsymbol{\Gamma},\boldsymbol{x})=\mathbb{E}_{\boldsymbol{g}}\Big[\mathbb{P}(\boldsymbol{A}|\boldsymbol{\Theta},\boldsymbol{g})\Big]\,, \tag{4.3}$$

where the expectation is taken over the random variable $\boldsymbol{g}$ whose distribution is the one specified in equation (4.2).

It is instructive to consider some limiting cases to understand how the prior on $\boldsymbol{g}$ affects what is being maximised in (4.3). To simplify the discussion, and without loss of generality, we only consider product nodes. If the γ_{sx} corresponding to different labels x and a fixed group s are all equal, so that $\gamma_{sx}\equiv\gamma_s$, the prior on $\boldsymbol{g}_P$ becomes

$$\mathbb{P}(\boldsymbol{g}_P|\boldsymbol{\Gamma}_P,\boldsymbol{x}_P)=\prod_{p\in\mathcal{N}_P}\gamma_{g_p}=\prod_{s\in\mathcal{K}_P}\gamma_s^{n_s}\,, \tag{4.4}$$

where n_s is the number of (product) nodes in group s. This expression defines a uniform prior on $\boldsymbol{g}_P$, conditioned on the expected size of block s being equal to $\gamma_s N_P$ [129]. In particular, as one may expect, the labels play no influence. Equation (4.4) also describes the effects of incorporating a prior when there is only one annotation.

In contrast, assume that for a particular label x, exactly one γ_{sx} is equal to 1. The prior probability of a particular set of block assignments $\boldsymbol{g}_P$ is then nonzero only when all nodes with the label x are assigned to the same block. In other words, the label x is perfectly informative of the block membership of nodes that carry that label.

Note that this framework does not impose a one-to-one (or even a one-to-many) correspondence between blocks and annotations. Multiple labels can be associated to the same block s (i.e., γ_{sx} can take large values for several different $x\in\mathcal{L}_P$), and nodes with one label can belong with large probability to more than one block.

EM algorithm for maximising the likelihood

We now explain how to maximise the expression (4.3) (or, more conveniently, its logarithm) with respect to the parameters $\boldsymbol{\Theta}$ and $\boldsymbol{\Gamma}$. In the process, we also obtain estimates of the posterior block-assignment probabilities. Consider $q_{\boldsymbol{g}} \geq 0$ with $\sum_{\boldsymbol{g}} q_{\boldsymbol{g}} = 1$. From Jensen's inequality,

$$\log \left[\sum_{\boldsymbol{g}} \mathbb{P}(\boldsymbol{A}|\boldsymbol{\Theta}, \boldsymbol{g}) \mathbb{P}(\boldsymbol{g}|\boldsymbol{\Gamma}, \boldsymbol{x}) \right] \geq \sum_{\boldsymbol{g}} q_{\boldsymbol{g}} \log \frac{\mathbb{P}(\boldsymbol{A}|\boldsymbol{\Theta}, \boldsymbol{g}) \mathbb{P}(\boldsymbol{g}|\boldsymbol{\Gamma}, \boldsymbol{x})}{q_{\boldsymbol{g}}} \tag{4.5}$$

$$= \sum_{\boldsymbol{g}} \left[q_{\boldsymbol{g}} \log \mathbb{P}(\boldsymbol{A}|\boldsymbol{\Theta}, \boldsymbol{g}) + q_{\boldsymbol{g}} \log \mathbb{P}(\boldsymbol{g}|\boldsymbol{\Gamma}, \boldsymbol{x}) - q_{\boldsymbol{g}} \log q_{\boldsymbol{g}} \right].$$

Equality holds if

$$q_{\boldsymbol{g}} = \frac{\mathbb{P}(\boldsymbol{A}|\boldsymbol{\Theta}, \boldsymbol{g}) \mathbb{P}(\boldsymbol{g}|\boldsymbol{\Gamma}, \boldsymbol{x})}{\sum_{\boldsymbol{g}'} \mathbb{P}(\boldsymbol{A}|\boldsymbol{\Theta}, \boldsymbol{g}') \mathbb{P}(\boldsymbol{g}'|\boldsymbol{\Gamma}, \boldsymbol{x})} \,. \tag{4.6}$$

The inequality (4.5) is at the core of the EM approach. The algorithm alternates between an expectation step, in which one determines $q_{\boldsymbol{g}}$ from (4.6), and a maximisation step, in which one maximises the right-hand side of (4.5) with respect to $\boldsymbol{\Theta}$ and $\boldsymbol{\Gamma}$ by setting the corresponding derivatives to 0.[3] As shown in the paper that first introduced the EM algorithm [40], this iterative approach ultimately converges to a local maximum (but rarely to a global maximum) of the log-likelihood function. It is common practice to run the algorithm from different initialisations of $q_{\boldsymbol{g}}$, and then use the instance with the largest likelihood value. When the EM algorithm is applied to SBM inference, the maximisation step typically consists of simple update formulas for the parameters. Given $q_{\boldsymbol{g}}$, let

$$q_r^c = \sum_{\boldsymbol{g}} q_{\boldsymbol{g}} \delta(g_c, r) \,, \quad q_s^p = \sum_{\boldsymbol{g}} q_{\boldsymbol{g}} \delta(g_p, s)$$

denote the marginal probabilities, respectively, for customer node c to belong to block r and for customer node p to belong to block s. Similarly, one can define the two-vertex marginals

$$q_{rs}^{cp} = \sum_{\boldsymbol{g}} q_{\boldsymbol{g}} \delta(g_c, r) \delta(g_p, s) \,,$$

which give the joint probabilities that customer node c is in block r and that product node p is in block s. To determine values of θ_{rs} that maximise the right-hand side of (4.5), note that only terms involving the log-likelihood $\log \mathbb{P}(\boldsymbol{A}|\boldsymbol{\Theta}, \boldsymbol{g})$ depend on these parameters. One can show that each of these terms is a concave function with respect

[3] One also needs to compute second derivatives to confirm that the resulting stationary points are, in fact, maxima (rather than minima or saddle points) of the original function.

to the parameters θ_{rs} (and this function is defined on an open interval), so to find a maximum it is sufficient to set the following derivatives to 0:

$$\frac{\partial}{\partial\theta_{rs}}\sum_{\boldsymbol{g}} q_{\boldsymbol{g}} \log \mathbb{P}(\boldsymbol{A}|\boldsymbol{\Theta},\boldsymbol{g}) = 0 \qquad \Leftrightarrow \qquad \sum_{\substack{c\in\mathcal{N}_C,\\ p\in\mathcal{N}_P}} q_{rs}^{cp}\left[A_{cp} - \frac{(1-A_{cp})d_c d_p\theta_{rs}}{1-d_c d_p\theta_{rs}}\right] = 0\,. \quad (4.7)$$

One obtains such an equation for each edge bundle $(r,s)\in\mathcal{K}_C\times\mathcal{K}_P$. For sufficiently large and sparse networks (see discussion in Section 2.1), $d_c d_p\theta_{rs}\ll 1$. Keeping only first-order terms in $d_c d_p\theta_{rs}$, we find that

$$\frac{(1-A_{cp})d_c d_p\theta_{rs}}{1-d_c d_p\theta_{rs}} \approx (1-A_{cp})d_c d_p\theta_{rs}\,.$$

Substituting this approximation into equation (4.7) above, we obtain

$$\theta_{rs} \approx \frac{\sum_{\substack{c\in\mathcal{N}_C,\\ p\in\mathcal{N}_P}} q_{rs}^{cp}A_{cp}}{\sum_{\substack{c\in\mathcal{N}_C,\\ p\in\mathcal{N}_P}} q_{rs}^{cp}(1-A_{cp})d_c d_p}\,.$$

For sparse networks, the numerator has $\mathcal{O}(N_C+N_P)$ terms, but the denominator has $\mathcal{O}\left(N_C N_P\right)$ terms, making it slow to compute in practice. Thus, as in [109], we approximate this denominator as follows:

$$\sum_{\substack{c\in\mathcal{N}_C,\\ p\in\mathcal{N}_P}} q_{rs}^{cp}(1-A_{cp})d_c d_p \approx \sum_{\substack{c\in\mathcal{N}_C,\\ p\in\mathcal{N}_P}} q_{rs}^{cp}d_c d_p \approx \Big(\sum_{c\in\mathcal{N}_C} q_r^c d_c\Big)\Big(\sum_{p\in\mathcal{N}_P} q_s^p d_p\Big)\,.$$

The first approximation relies on ignoring the factor $1-A_{cp}$, effectively going from a sum over just non-edges (i.e., node pairs (c,p) with $A_{cp}=0$) to a sum over both edges and non-edges. In the sparse case, this adds $\mathcal{O}(N_C+N_P)$ additional terms to a sum that otherwise consists of $\mathcal{O}(N_C N_P)$ terms, which should only introduce a small error for most networks. The second approximation relies on the observation that $q_{rs}^{cp}\approx q_r^c q_s^p$ for most node pairs (c,p), because block assignments of nodes that are far apart in a network are largely uncorrelated. With this approximation, we find the final update equation for θ_{rs}:

$$\theta_{rs} \approx \frac{\sum_{\substack{c\in\mathcal{N}_C,\\ p\in\mathcal{N}_P}} q_{rs}^{cp}A_{cp}}{\Big(\sum_{c\in\mathcal{N}_C} q_r^c d_c\Big)\Big(\sum_{p\in\mathcal{N}_P} q_s^p d_p\Big)}\,. \quad (4.8)$$

Equation (4.8) is similar to the one obtained by Newman and Clauset [109] for non-bipartite networks; only the summation indices differ, to account for the existence of two node types in the bipartite case.

The parameters γ_{rx} are subject to the constraints $\sum_{r\in\mathcal{K}_C}\gamma_{rx}=1$ for $x\in\mathcal{L}_C$ and $\sum_{s\in\mathcal{K}_P}\gamma_{sx}=1$ for $x\in\mathcal{L}_P$, so we use Lagrange multipliers to find values that

maximise[4] (4.5). Only the terms involving the log-prior $\log \mathbb{P}(\boldsymbol{g}|\boldsymbol{\Gamma}, \boldsymbol{x})$ depend on these parameters, and we obtain (with no need for any approximations) the following update equations:

$$\gamma_{rx} = \frac{\sum_{c \in \mathcal{N}_C} \delta(x, x_c) q_r^c}{\sum_{c \in \mathcal{N}_C} \delta(x, x_c)} \quad \text{for } r \in \mathcal{K}_C,\ x \in \mathcal{L}_C\,, \tag{4.9}$$

$$\gamma_{sx} = \frac{\sum_{p \in \mathcal{N}_P} \delta(x, x_p) q_s^p}{\sum_{p \in \mathcal{N}_P} \delta(x, x_p)} \quad \text{for } s \in \mathcal{K}_P,\ x \in \mathcal{L}_P\,. \tag{4.10}$$

These formulas — which are also very similar to their non-bipartite counterparts [109] — make intuitive sense: for customer nodes (and analogously for products), the inferred probability for a node with label x to belong to block r is the total probability with which nodes labelled x belong to group r, divided by the number of nodes with that label.

Equations (4.8)–(4.10) comprise the maximisation step of the EM algorithm. We now turn to the algorithm's expectation step, for which we use belief propagation.

Belief-propagation equations

The expectation step of the EM algorithm consists of estimating the posterior distribution over block assignments, given current estimates of the SBM parameters $\boldsymbol{\Theta}$ and $\boldsymbol{\Gamma}$. One way to determine this distribution is by calculating the posterior probabilities $q_{\boldsymbol{g}}$ from equation (4.6). However, this is impractical even for relatively small networks, as the number of possible block-assignment vectors $\boldsymbol{g}$ is exponential in the number of nodes. It is also not necessary, as the update equations (4.8)–(4.10) involve only the single-vertex marginals q_r^c and q_s^p and the two-vertex marginals q_{rs}^{cp}; this is a much smaller number of quantities to handle. The method of belief propagation produces estimates of these probabilities.

For the derivation in this section, we take an approach similar to Yan et al. [163], who first applied BP to inference in degree-corrected SBMs.[5] (See also Moore's review article [97].) When applied to SBMs, belief propagation works by having different nodes send "messages" to each other about the extent to which they belong to different blocks. (These are the "beliefs" that give the method its name.) Specifically, let $\mu_r^{c \to p}$

[4]To show that partial derivatives of the Lagrangian yield maxima of the original function, one needs to study the Hessian (called a "bordered Hessian" [68]) of this Lagrangian. Because certain minors of the bordered Hessian have alternating signs, it follows that stationary points of the original function are local maxima (rather than local minima or saddle points).

[5]Newman and Clauset did not provide any details in [109] on the belief-propagation portion of their algorithm. However, their code — which we adapted to bipartite networks — has the final version of the relevant equations, so we are able to confirm that their derivation must have been similar.

denote the probability that customer node c is in block r when product node p is absent from the network. Similarly, let $\mu_s^{p\to c}$ be the probability that product node p is in block s when customer node c is absent from the network. Below, we derive the relevant equations for the messages sent by customer nodes; the ones corresponding to product nodes are analogous.

As in [163], let $h(d_c, d_p, \theta_{rs}, A_{cp})$ be a function specifying the probability that A_{cp} takes the observed value (0 or 1 for unweighted networks) when node c is in block r and node p is in block s. For a degree-corrected Bernoulli SBM,

$$h(d_c, d_p, \theta_{rs}, A_{cp}) = (d_c d_p \theta_{rs})^{A_{cp}} (1 - d_c d_p \theta_{rs})^{1-A_{cp}} .$$

For an annotated SBM, the messages update in an iterative fashion according to the equation

$$\mu_r^{c\to p} = \frac{1}{Z^{c\to p}} \gamma_{r,x_c} \prod_{p' \in \mathcal{N}_P \setminus \{p\}} \sum_{s \in \mathcal{K}_P} \mu_s^{p'\to c} h(d_c, d_{p'}, \theta_{rs}, A_{cp'}) , \tag{4.11}$$

where $Z^{c\to p}$ is a normalisation constant ensuring that $\sum_{r\in\mathcal{K}_C} \mu_r^{c\to p} = 1$ (with $c \in \mathcal{N}_C$ and $p \in \mathcal{N}_P$). The formula (4.11) differs in two ways from the one in [163]. The first difference is that we account for the bipartite network structure, as messages are sent only between nodes of different types. The second difference is that we incorporate an annotation-dependent prior γ_{r,x_c}, whereas the prior in [163] is specified in terms of probabilities γ_r for a node to belong to each existing block. Qualitatively, the formula (4.11) says that the messages that c sends to a node p are based on a combination of the messages that c receives from all other product nodes besides p. These messages are weighted by the function h, which amplifies messages about block assignments that are consistent with the observed values $A_{cp'}$.

As Yan et al. indicated, equation (4.11) requires $\mathcal{O}(N_P K_P)$ calculations for each customer node c and customer block r, so using it for computations is impractical. We can make a simplification by noting that the messages $\mu_r^{c\to p}$ are approximately equal (up to $\mathcal{O}(1/N)$ terms, where $N = N_C + N_P$) for nodes p that are not adjacent to c. Indeed, if $A_{cp} = 0$, we have

$$\mu_r^{c\to p} = \frac{1}{Z^{c\to p}} \gamma_{r,x_c} \frac{1}{\sum_{s\in\mathcal{K}_P} \mu_s^{p\to c} h(d_c, d_p, \theta_{rs}, 0)} \prod_{p' \in \mathcal{N}_P} \sum_{s \in \mathcal{K}_P} \mu_s^{p'\to c} h(d_c, d_{p'}, \theta_{rs}, A_{cp'})$$

and

$$\begin{aligned} \sum_{s\in\mathcal{K}_P} \mu_s^{p\to c} h(d_p, d_c, \theta_{rs}, 0) &= \sum_{s\in\mathcal{K}_P} \mu_s^{p\to c} (1 - d_c d_p \theta_{rs}) \\ &= 1 - \sum_{s\in\mathcal{K}_P} \mu_s^{p\to c} d_c d_p \theta_{rs} = 1 - \mathcal{O}(1/N) . \end{aligned}$$

Therefore, when $A_{cp} = 0$, we have $\mu_r^{c\to p} \approx q_r^c$, where

$$q_r^c = \frac{1}{Z^c}\gamma_{r,x_c} \prod_{p\in\mathcal{N}_P} \sum_{s\in\mathcal{K}_P} \mu_s^{p\to c} h(d_c, d_p, \theta_{rs}, A_{cp})$$

are the marginal probabilities of node c belonging to each of the blocks $r \in \mathcal{K}_C$. The normalisation constant Z^c ensures that $\sum_{r\in\mathcal{K}_C} q_r^c = 1$. It follows that the messages $\mu_r^{c\to p}$ are given approximately by

$$\mu_r^{c\to p} \approx \frac{\gamma_{r,x_c}}{Z^{c\to p}} \prod_{\substack{p'\in\mathcal{N}_P\setminus\{p\}\\ A_{cp'}\neq 0}} \sum_{s\in\mathcal{K}_P} \mu_s^{p'\to c} h(d_c, d_{p'}, \theta_{rs}, A_{cp'}) \prod_{\substack{p'\in\mathcal{N}_P\setminus\{p\}\\ A_{cp'}= 0}} \sum_{s\in\mathcal{K}_P} q_s^{p'} h(d_c, d_{p'}, \theta_{rs}, 0)\,. \tag{4.12}$$

Taking logarithms, we obtain the following set of recursive equations for the messages and the one-vertex marginals:

$$\begin{aligned}\log \mu_r^{c\to p} = \log\gamma_{r,x_c} - \log Z^{c\to p} &+ \sum_{\substack{p'\in\mathcal{N}_P\setminus\{p\}\\ A_{cp'}\neq 0}} \log\left(\sum_{s\in\mathcal{K}_P} \mu_s^{p'\to c} h(d_c, d_{p'}, \theta_{rs}, A_{cp'})\right)\\ &+ \sum_{\substack{p'\in\mathcal{N}_P\setminus\{p\}\\ A_{cp'}= 0}} \log\left(\sum_{s\in\mathcal{K}_P} q_s^{p'} h(d_c, d_{p'}, \theta_{rs}, 0)\right),\end{aligned} \tag{4.13}$$

$$\log q_r^c = \log\gamma_{r,x_c} - \log Z^c + \sum_{p\in\mathcal{N}_P} \log\left(\sum_{s\in\mathcal{K}_P} \mu_s^{p\to c} h(d_c, d_p, \theta_{rs}, A_{cp})\right). \tag{4.14}$$

Although we have reduced the number of messages from $\mathcal{O}(N_C N_P)$ to $\mathcal{O}(N_C + N_P + m)$, where m is the number of edges, the expressions above still require $\mathcal{O}(N_C + N_P)$ operations to calculate *each* message and one-vertex marginal, pushing the total number of calculations to $\mathcal{O}(N^2)$ per BP iteration. Thus, we need to make further simplifications for this procedure to be feasible for larger networks. We note that we can add or subtract terms that do not depend on r to the expressions from (4.13) and (4.14), as long as we normalise them so that $\sum_{r\in\mathcal{K}_C} \mu_r^{c\to p} = 1$ and $\sum_{r\in\mathcal{K}_C} q_r^c = 1$ after we update all messages and marginals. With this observation, we can write

$$\begin{aligned}\log \mu_r^{c\to p} = \log\gamma_{r,x_c} &+ \sum_{\substack{p'\in\mathcal{N}_P\setminus\{p\}\\ A_{cp'}\neq 0}} \log\left(\sum_{s\in\mathcal{K}_P} \mu_s^{p'\to c}\theta_{rs}\right)\\ &+ \sum_{\substack{p'\in\mathcal{N}_P\setminus\{p\}\\ A_{cp'}= 0}} \log\left(1 - \sum_{s\in\mathcal{K}_P} q_s^{p'} d_c d_{p'}\theta_{rs}\right) + (\text{const.})\end{aligned}$$

$$\approx \log \gamma_{r,x_c} + \sum_{\substack{p' \in \mathcal{N}_P \setminus \{p\} \\ A_{cp'} \neq 0}} \log \left(\sum_{s \in \mathcal{K}_P} \mu_s^{p' \to c} \theta_{rs} \right) - \sum_{\substack{p' \in \mathcal{N}_P \setminus \{p\} \\ A_{cp'} = 0}} \sum_{s \in \mathcal{K}_P} q_s^{p'} d_c d_{p'} \theta_{rs} + (\text{const.}) \,,$$

where $\kappa_s = \sum_{p \in \mathcal{N}_P} d_p q_s^p$ is the total expected degree of product nodes in block s. The approximation arises from a first-order Taylor expansion of the logarithm, for which we note that the quantity $\sum_{s \in \mathcal{K}_P} q_s^{p'} d_c d_{p'} \theta_{rs}$ is small for sparse networks. We can make further approximations to the expression for $\log \mu_r^{c \to p}$:

$$\log \mu_r^{c \to p} \approx \log \gamma_{r,x_c} + \sum_{\substack{p' \in \mathcal{N}_P \setminus \{p\} \\ A_{cp'} \neq 0}} \log \left(\sum_{s \in \mathcal{K}_P} \mu_s^{p' \to c} \theta_{rs} \right) - d_c \sum_{s \in \mathcal{K}_P} \kappa_s \theta_{rs} + (\text{const.}) \,. \quad (4.15)$$

To obtain (4.15), note that

$$\sum_{\substack{p' \in \mathcal{N}_P \setminus \{p\} \\ A_{cp'} = 0}} d_{p'} q_s^{p'} = \sum_{p' \in \mathcal{N}_P \setminus \{p\}} d_{p'} q_s^{p'} - \sum_{\substack{p' \in \mathcal{N}_P \setminus \{p\} \\ A_{cp'} \neq 0}} d_{p'} q_s^{p'} \approx \sum_{p' \in \mathcal{N}_P} d_{p'} q_s^{p'} \equiv \kappa_s \,,$$

which holds for sparse networks. This approximation discards $\mathcal{O}(1)$ terms (i.e., those with $A_{cp'} \neq 0$) from a sum of $\mathcal{O}(N_P)$ terms of comparable magnitude. Updating each message $\mu_r^{c \to p}$ using equation (4.15) is now a constant-time operation, which implies that overall one BP iteration takes $\mathcal{O}(N_C + N_P + m) = \mathcal{O}(N)$ computations. (Recall that we only need to calculate messages $\mu_r^{c \to p}$ for nodes c and p that are adjacent, as for all other node pairs we have $\mu_r^{c \to p} \approx q_r^c$.)

A similar derivation yields the following approximation for the logarithms of the one-vertex marginals:

$$\log q_r^c \approx \log \gamma_{r,x_c} + \sum_{\substack{p' \in \mathcal{N}_P \\ A_{cp'} \neq 0}} \log \left(\sum_{s \in \mathcal{K}_P} \mu_s^{p' \to c} \theta_{rs} \right) - d_c \sum_{s \in \mathcal{K}_P} \kappa_s \theta_{rs} + (\text{const.}) \,. \quad (4.16)$$

Each of these update equations requires $\mathcal{O}(1)$ calculations, so updating all one-vertex marginals requires $\mathcal{O}(N)$ operations.

The two-vertex marginals can be computed as in [163] from

$$q_{rs}^{cp} \propto h(d_c, d_p, \theta_{rs}, A_{cp}) \mu_r^{c \to p} \mu_s^{p \to c} \,, \quad (4.17)$$

with $\sum_{(r,s) \in \mathcal{K}_C \times \mathcal{K}_P} q_{rs}^{cp} = 1$ for all $c \in \mathcal{N}_C$ and $p \in \mathcal{N}_P$. One only needs to compute these expressions for node pairs (c, p) with $A_{cp} = 1$, so the overall runtime of this computation is $\mathcal{O}(N)$.

To summarise, equations (4.15)–(4.17) form the basis of the expectation step of the EM algorithm. (These equations are analogous to those for unipartite networks that were implemented by Newman and Clauset in the code that accompanies their paper [109].) This completes the description of our inference algorithm for unweighted bipartite networks with node annotations. We apply this algorithm to consumer–product networks in Section 4.4.

4.2.2 Weighted Networks

The method from Section 4.2.1 works only on unweighted networks. For our application, this forces us to discard valuable information about the weights of connections (as measured by one of the weighting schemes from Section 3.2.2) between customers and products. Using unweighted networks also forces us to select a threshold to discard some of the smaller-weight edges, which introduces an additional tuning parameter. It is thus preferable to have an inference algorithm that works directly on weighted adjacency matrices.

In this section, we combine the approach of Newman and Clauset [109] with the weighted SBM proposed by Aicher et al. [3]. A recent paper by Murphy et al. [99] proposed an approach similar to ours for combining these two studies. Our work differs in several ways from [99]: (1) we focus on sparse, rather than dense, networks; (2) we incorporate degree correction; (3) we use discrete, rather than continuous annotations; and (4) we consider networks with bipartite structure. We also assume, as in Section 4.2.1, that the block structure is informed by the annotations via a prior on $\boldsymbol{g}$, whereas Murphy et al. modelled the reverse scenario of annotations arising from the latent block structure[6] (similar to the work by Hric et al. [65]).

Likelihood for sparse, weighted SBMs

Let $\boldsymbol{W} = (W_{cp})_{c\in\mathcal{N}_C,\, p\in\mathcal{N}_P}$ denote a weighted adjacency matrix corresponding to a bipartite network, and let $\boldsymbol{A} = (A_{cp})_{c\in\mathcal{N}_C,\, p\in\mathcal{N}_P}$ be its corresponding unweighted adjacency matrix. Aicher et al. [3] (and also other authors [119]) assumed that the edges and the weights are generated independently, conditioned on the vector $\boldsymbol{g}$ of

[6]The final equations obtained by Murphy et al. [99] are in fact analogous to ours, owing to an inconsistency between their model and their computations. More precisely, although they initially prescribed a conditional probability distribution $\mathbb{P}(\boldsymbol{x}|\boldsymbol{g})$ for generating annotations given the block structure, they normalised the relevant parameters such that $\sum_{\boldsymbol{g}} \mathbb{P}(\boldsymbol{g}|\boldsymbol{x}) = 1$. The latter normalisation is consistent with a partition arising from the annotations, rather than the other way around.

block assignments. This leads to a log-likelihood of the form

$$\log \mathbb{P}(\boldsymbol{A}, \boldsymbol{W} | \boldsymbol{g}, \boldsymbol{\theta}) = \alpha \log \mathbb{P}(\boldsymbol{A} | \boldsymbol{g}, \boldsymbol{\theta}^{(e)}) + (1 - \alpha) \log \mathbb{P}(\boldsymbol{W} | \boldsymbol{g}, \boldsymbol{\theta}^{(w)}) \,, \tag{4.18}$$

where $\boldsymbol{\theta}^{(e)}$ and $\boldsymbol{\theta}^{(w)}$, respectively, parametrise the edge distributions and the weight distributions, and $\alpha \in [0, 1]$ controls the relative importance of the two terms.

Let $\mathcal{E}$ be the set of observed interactions (including both edges and non-edges), and let $\mathcal{W} \subset \mathcal{E}$ be the set of observed weights.[7] For our application, we have

$$\begin{aligned} \mathcal{E} &= \{(c, p) : c \in \mathcal{N}_C,\, p \in \mathcal{N}_P\} = \mathcal{N}_C \times \mathcal{N}_P \,, \\ \mathcal{W} &= \{(c, p) : c \in \mathcal{N}_C,\, p \in \mathcal{N}_P,\, A_{cp} \neq 0\} \subset \mathcal{E} \,. \end{aligned}$$

As in [3], we use exponential-family probability distributions to model both edge existence and edge weights in (4.18). In this case, the two distributions can be parametrised in terms of their *(natural) sufficient statistics* $\boldsymbol{T}_e$ and $\boldsymbol{T}_w$ and *natural parameters* $\boldsymbol{\eta}_e$ and $\boldsymbol{\eta}_w$:

$$\log \mathbb{P}(\boldsymbol{A}, \boldsymbol{W} | \boldsymbol{g}, \boldsymbol{\theta}) = \alpha \sum_{(c,p) \in \mathcal{E}} \boldsymbol{T}_e(A_{cp}) \cdot \boldsymbol{\eta}_e \left(\boldsymbol{\theta}^{(e)}_{g_c g_p} \right) + (1 - \alpha) \sum_{(c,p) \in \mathcal{W}} \boldsymbol{T}_w(W_{cp}) \cdot \boldsymbol{\eta}_w \left(\boldsymbol{\theta}^{(w)}_{g_c g_p} \right) . \tag{4.19}$$

The sufficient statistics $\boldsymbol{T}_e$ and $\boldsymbol{T}_w$ depend only on the data (i.e., on the matrices $\boldsymbol{A}$ and $\boldsymbol{W}$), and the natural parameters $\boldsymbol{\eta}_e$ and $\boldsymbol{\eta}_w$ depend only on the quantities $\boldsymbol{\theta}^{(e)}$ and $\boldsymbol{\theta}^{(w)}$ that we used to parametrise the distributions. To give an example, we use a degree-corrected Poisson model to model edge existence and lognormal distributions to model weights. For the former,

$$\log \mathbb{P}(\boldsymbol{A} | \boldsymbol{g}, \boldsymbol{\theta}^{(e)}) = \sum_{(c,p) \in \mathcal{E}} \left(A_{cp} \log \theta^{(e)}_{g_c g_p} - d_c d_p \theta^{(e)}_{g_c g_p} \right) + (\text{const.}) \,,$$

ignoring terms that depend on d_c, d_p, and A_{cp} (which are known quantities). Thus, the sufficient statistics and natural parameters are

$$\boldsymbol{T}_e(A_{cp}) = (A_{cp}, -d_c d_p) \text{ and } \boldsymbol{\eta}_e \left(\theta^{(e)}_{g_c g_p} \right) = \left(\log \theta^{(e)}_{g_c g_p}, \theta^{(e)}_{g_c g_p} \right) .$$

For the latter, assume that the edge weights in each edge bundle $(r, s) \in \mathcal{K}_C \times \mathcal{K}_P$ follow a lognormal distribution with mean parameter μ_{rs} and standard deviation parameter

[7] It is possible to use this notation to model unobserved interactions (i.e., node pairs $(c, p) \in \mathcal{N}_C \times \mathcal{N}_P$ for which the value of A_{cp} is unknown), and it is possible, in principle, for some of the weights to be equal to 0 even though the corresponding edges are present in the network (i.e., $A_{cp} = 1$ and $W_{cp} = 0$). We do not consider either of these scenarios in our derivation.

σ_{rs}. (Thus, $\boldsymbol{\theta}^{(w)}$ is shorthand notation for these means and standard deviations.) We then have

$$\begin{aligned}\log \mathbb{P}(\boldsymbol{W}|\boldsymbol{g},\boldsymbol{\theta}^{(w)}) &= \sum_{(c,p)\in\mathcal{W}} \log\left[\frac{1}{\sqrt{2\pi}\sigma_{g_c g_p} W_{cp}} e^{-\frac{\left(\log W_{cp}-\mu_{g_c g_p}\right)^2}{2\sigma^2_{g_c g_p}}}\right] \\ &= \sum_{(c,p)\in\mathcal{W}} \left[-\frac{\left(\log W_{cp}-\mu_{g_c g_p}\right)^2}{2\sigma^2_{g_c g_p}} - \log\sigma_{g_c g_p}\right] + (\text{const.})\,,\end{aligned}$$

up to additive constants that depend on known quantities. Thus,

$$\begin{aligned}\boldsymbol{T}_w(W_{cp}) &= \left(\log W_{cp},\ \log^2 W_{cp},\ 1\right)\,, \\ \boldsymbol{\eta}_w(\mu_{g_c g_p},\sigma_{g_c g_p}) &= \left(\frac{\mu_{g_c g_p}}{\sigma^2_{g_c g_p}},\ -\frac{1}{2\sigma^2_{g_c g_p}},\ -\frac{\mu^2_{g_c g_p}}{2\sigma^2_{g_c g_p}} - \log\sigma_{g_c g_p}\right)\,.\end{aligned}$$

As noted in [3], it is convenient to group the edge-existence and edge-weight terms from equation (4.19) into the compact form

$$\log \mathbb{P}(\boldsymbol{A},\boldsymbol{W}|\boldsymbol{g},\boldsymbol{\theta}) = \sum_{(c,p)\in\mathcal{E}} \boldsymbol{T}(A_{cp},W_{cp})\cdot\boldsymbol{\eta}\left(\boldsymbol{\theta}^{(e)}_{g_c g_p},\boldsymbol{\theta}^{(w)}_{g_c g_p}\right). \tag{4.20}$$

In this expression, the vector $\boldsymbol{T}$ is approximately[8] a concatenation of the vectors $\alpha\boldsymbol{T}_e$ and $(1-\alpha)\boldsymbol{T}_w$, and $\boldsymbol{\eta}$ is a concatenation of $\boldsymbol{\eta}_e$ and $\boldsymbol{\eta}_w$. The quantities $\boldsymbol{T}$ and $\boldsymbol{\eta}$ then correspond to the sufficient statistics and natural parameters of another distribution from the exponential family.

Conjugate priors

Rather than simply maximising the likelihood (4.19), Aicher et al. used a Bayesian approach to estimate the joint posterior distribution of $\boldsymbol{\theta}^{(e)}$, $\boldsymbol{\theta}^{(w)}$, and $\boldsymbol{g}$. This requires that we impose priors on the SBM parameters $\boldsymbol{\theta}^{(e)}$ and $\boldsymbol{\theta}^{(w)}$, in addition to an annotation-driven prior on the block assignments $\boldsymbol{g}$ that we also used in Section 4.2.1. A crucial (but reasonable) assumption is that the prior factorises as

$$\pi(\boldsymbol{\theta},\boldsymbol{g}) = \pi_e(\boldsymbol{\theta}^{(e)})\pi_w(\boldsymbol{\theta}^{(w)})\pi_{\boldsymbol{g}}(\boldsymbol{g})\,,$$

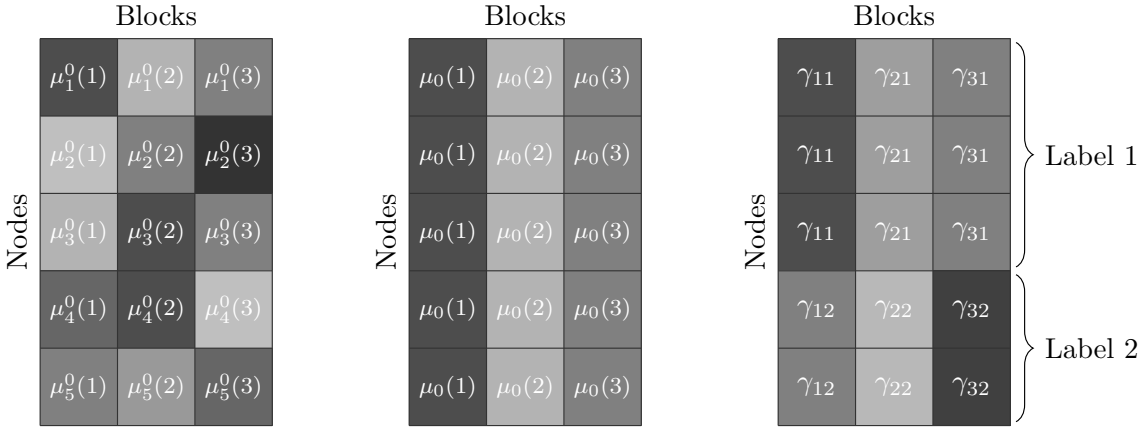
meaning that $\boldsymbol{g}$, $\boldsymbol{\theta}^{(e)}$, and $\boldsymbol{\theta}^{(w)}$ are sampled independently from their respective prior distributions.

[8] One has to ensure that non-edges $(c,p)\in\mathcal{E}\setminus\mathcal{W}$ only contribute to the edge-existence term. This can be achieved by setting the corresponding entries of $\boldsymbol{T}$ to 0.

We first describe the priors for the SBM parameters $\boldsymbol{\theta}^{(e)}$ and $\boldsymbol{\theta}^{(w)}$. For an exponential-family distribution parametrised by $\boldsymbol{T}$ and $\boldsymbol{\eta}$, the standard prior is

$$\pi(\boldsymbol{\theta}) = \frac{1}{Z} \exp(\boldsymbol{\tau}_0 \cdot \boldsymbol{\eta}(\boldsymbol{\theta}))\,, \tag{4.21}$$

where $\boldsymbol{\tau}_0$ is a given vector of hyperparameters and Z is a normalising constant. In fact, this is a conjugate prior for $\boldsymbol{\theta}$: the posterior distribution, obtained after multiplication by the likelihood, is from the same family of distributions as the prior and it takes the form $\pi^*(\boldsymbol{\theta}) \propto \exp(\boldsymbol{\tau} \cdot \boldsymbol{\eta}(\boldsymbol{\theta}))$. As we show later in this section, the parameters $\boldsymbol{\tau}$ of the posterior can be easily computed from $\boldsymbol{\tau}_0$ and the sufficient statistics $\boldsymbol{T}$. The conjugate prior for a Poisson distribution (which we use to model edge existence) is a Gamma distribution, which has two hyperparameters $\boldsymbol{\tau}_0$, and the conjugate prior for a lognormal distribution with unknown mean and variance (which we use to model edge weights) is a Normal–Gamma distribution, which has four hyperparameters.



(a) Different priors $\mu_i^0(r)$ (b) Single prior $\mu_0(r)$ (c) Annotated prior $\mu_i^0(r) = \gamma_{r,x_i}$

Figure 4.1: Illustration of different types of categorical priors that one can impose on $\boldsymbol{g}_C$ and $\boldsymbol{g}_P$. (a) General categorical prior, where $\mu_i^0(r)$ is the probability that node i (customer or product) belongs to block r, subject to $\sum_r \mu_i^0(r) = 1$. (b) Single categorical prior $\mu_0(r)$ for the probability of a node to be in block r, with $\sum_r \mu_0(r) = 1$. The quantities $\mu_0(r)$ also correspond to the expected fraction of nodes assigned to block r in the prior distribution. (c) Annotation-induced prior, with $\mu_i^0(r) = \gamma_{r,x_i}$ giving the probability of node i with label x_i to be in block r. Nodes with the same label have identical prior distributions.

For the block assignments $\boldsymbol{g}$, Aicher et al. used a categorical prior μ that, in the most general case, has $N_C K_C$ parameters for the customer nodes and $N_P K_P$ parameters for the product nodes. Specifically, $\mu_c^{(0)}(r)$ is the prior probability that

customer node $c \in \mathcal{N}_C$ is assigned to customer block $r \in \mathcal{K}_C$. (One can similarly define $\mu_p^{(0)}(s)$ for products, with $p \in \mathcal{N}_P$ and $s \in \mathcal{K}_P$.) Figure 4.1(a) illustrates this general prior. However, it is impractical to have to specify this many parameters for the prior distribution. Instead, it is common [3, 118, 129] to choose the same prior $\mu_0(\cdot)$ for all customer nodes (respectively, for all product nodes), as illustrated in Figure 4.1(b). In their paper, Aicher et al. used a flat prior, which corresponds to setting $\mu_C^{(0)}(r) = 1/K_C$ $(r \in \mathcal{K}_C)$ and $\mu_P^{(0)}(s) = 1/K_P$ $(s \in \mathcal{K}_P)$ in the bipartite case.

The annotation-induced prior from [109], illustrated in Figure 4.1(c), sits neatly between these two extremes, with nodes that have the same annotation sharing a prior distribution described by identical parameters. In this case, there are $(K_C L_C + K_P L_P)$ parameters in total that specify the priors for customer and product nodes.

In the applications that we discuss in Section 4.4, only product nodes are annotated. In this case, the prior on block assignments takes the following form:

$$\pi_{\boldsymbol{g}}(\boldsymbol{g}) = \prod_{c \in \mathcal{N}_C} \mu_C^0(g_c) \prod_{p \in \mathcal{N}_P} \gamma_{g_p, x_p} \,.$$

One can take the Bayesian paradigm a step further and also impose priors on the parameters $\mu_C^0(r)$ and γ_{sx}. (The Dirichlet distribution is the natural choice for these priors.) We follow Aicher et al. by setting $\mu_C^0(r) = 1/K_C$ $(r \in \mathcal{K}_C)$ for customer nodes, and we infer the parameters γ_{sx} (with $s \in \mathcal{K}_P$ and $x \in \mathcal{L}_P$) as described next.

Variational-Bayes method for approximating the posterior distribution

Given the likelihood and the relevant priors, the posterior distribution π^* of the parameters follows from Bayes' law:

$$\pi^*(\boldsymbol{g}, \boldsymbol{\theta}) \propto \mathbb{P}(\boldsymbol{A}, \boldsymbol{W} | \boldsymbol{g}, \boldsymbol{\theta}) \pi(\boldsymbol{g}, \boldsymbol{\theta}) \,.$$

Estimating the full posterior distribution π^* is difficult in practice. One possibility is to determine the maximum a posteriori (MAP) parameter estimates, which correspond to the mode of the posterior distribution. Another option, employed by Aicher et al. [3], is to find an analytical approximation of the posterior distribution using a method called "variational Bayes". More precisely, we seek to approximate the posterior π^* by a factorisable distribution of the form

$$\pi^*(\boldsymbol{g}, \boldsymbol{\theta}) \sim Q(\boldsymbol{g}, \boldsymbol{\theta}) = Q_{\boldsymbol{g}}(\boldsymbol{g}) Q_{\boldsymbol{\theta}}(\boldsymbol{\theta}) \,.$$

Following [3], we assume that this posterior takes the form

$$Q(\boldsymbol{g}, \boldsymbol{\theta}) = \prod_{c \in \mathcal{N}_C} \mu_c(g_c) \prod_{p \in \mathcal{N}_P} \mu_p(g_p) \prod_{(r,s) \in \mathcal{K}_C \times \mathcal{K}_P} \frac{1}{Z(\boldsymbol{\tau}_{rs})} \exp(\boldsymbol{\tau}_{rs} \cdot \boldsymbol{\eta}(\boldsymbol{\theta}_{rs})) \,,$$

where $\mu_c(r)$ (for $c \in \mathcal{N}_C$ and $r \in \mathcal{K}_C$) and $\mu_p(s)$ (for $p \in \mathcal{N}_P$ and $s \in \mathcal{K}_P$) denote the posterior block-membership probabilities for customers and products. These are the key quantities that we wish to infer.

The best such approximation to the posterior is one that minimises the Kullback–Leibler (KL) divergence between π^* and Q. As shown in [3], minimising the KL divergence is equivalent to maximising the following functional with respect to the parameters of Q:

$$\mathcal{G}(Q) = \mathbb{E}_Q[\log \mathbb{P}(\boldsymbol{A}, \boldsymbol{W} | \boldsymbol{g}, \boldsymbol{\theta})] + \mathbb{E}_Q \left[\log \left(\frac{\pi(\boldsymbol{g}, \boldsymbol{\theta})}{Q_{\boldsymbol{g}}(\boldsymbol{g}) Q_{\boldsymbol{\theta}}(\boldsymbol{\theta})} \right) \right] . \tag{4.22}$$

Essentially, the first term maximises the expectation of the likelihood under the approximation Q, and the second term is a form of regularisation, which keeps Q close to the prior distribution π.

In this model, as in most SBMs, all edges between two pairs of blocks $(r, s) \in \mathcal{K}_C \times \mathcal{K}_P$ are described by the same set of parameters $\boldsymbol{\theta}_{rs}^{(e)}$ and $\boldsymbol{\theta}_{rs}^{(w)}$ (notwithstanding degree correction). For notational convenience, let $\mathcal{B} = \mathcal{K}_C \times \mathcal{K}_P$ denote the set of all such edge bundles. After some calculations, we can rewrite the functional (4.22) as

$$\begin{aligned} \mathcal{G}(Q) = &\sum_{b \in \mathcal{B}} \log \frac{Z(\boldsymbol{\tau}_b)}{Z(\boldsymbol{\tau}_0)} + \sum_{b \in \mathcal{B}} (\langle \boldsymbol{T} \rangle_b + \boldsymbol{\tau}_0 - \boldsymbol{\tau}_b) \cdot \langle \boldsymbol{\eta} \rangle_b \\ &+ \sum_{c \in \mathcal{N}_C} \sum_{\boldsymbol{g}} \mu_c(g_c) \log \frac{\mu_0(g_c)}{\mu_c(g_c)} + \sum_{p \in \mathcal{N}_P} \sum_{\boldsymbol{g}} \mu_p(g_p) \log \frac{\gamma_{g_p, x_p}}{\mu_p(g_p)} , \end{aligned} \tag{4.23}$$

where $\langle \boldsymbol{T} \rangle_b$ and $\langle \boldsymbol{\eta} \rangle_b$ are the expectations of the sufficient statistics and natural parameters, respectively, of edge bundle b under the approximation Q. More precisely,

$$\begin{aligned} \langle \boldsymbol{T} \rangle_b &= \alpha \sum_{\substack{(c,p) \in \mathcal{E} \\ (g_c, g_p) = b}} \mu_c(g_c) \mu_p(g_p) \boldsymbol{T}_e(A_{cp}) + (1 - \alpha) \sum_{\substack{(c,p) \in \mathcal{W} \\ (g_c, g_p) = b}} \mu_c(g_c) \mu_p(g_p) \boldsymbol{T}_w(W_{cp}) , \\ \langle \boldsymbol{\eta} \rangle_b &= \frac{\partial}{\partial \boldsymbol{\tau}} \log Z(\boldsymbol{\tau}) \Big|_{\boldsymbol{\tau} = \boldsymbol{\tau}_b} . \end{aligned}$$

The first equality follows easily from the definition of expectation, but the second one is less obvious. Because $Z(\boldsymbol{\tau}_b)$ is the normalisation constant for the distribution $\exp(\boldsymbol{\tau}_b \cdot \boldsymbol{\eta}(\boldsymbol{\theta}_b))$, we have

$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{\tau}} \log Z(\boldsymbol{\tau}) \Big|_{\boldsymbol{\tau} = \boldsymbol{\tau}_b} &= \frac{\frac{\partial}{\partial \boldsymbol{\tau}} Z(\boldsymbol{\tau}) \Big|_{\boldsymbol{\tau} = \boldsymbol{\tau}_b}}{Z(\boldsymbol{\tau}_b)} = \frac{\frac{\partial}{\partial \boldsymbol{\tau}} \int_{\boldsymbol{\theta}_b} \exp(\boldsymbol{\tau} \cdot \boldsymbol{\eta}(\boldsymbol{\theta}_b)) d\boldsymbol{\theta}_b \Big|_{\boldsymbol{\tau} = \boldsymbol{\tau}_b}}{Z(\boldsymbol{\tau}_b)} \\ &= \int_{\boldsymbol{\theta}_b} \boldsymbol{\eta}(\boldsymbol{\theta}_b) \frac{\exp(\boldsymbol{\tau}_b \cdot \boldsymbol{\eta}(\boldsymbol{\theta}_b))}{Z(\boldsymbol{\tau}_b)} d\boldsymbol{\theta}_b = \mathbb{E}_Q \left[\boldsymbol{\eta}(\boldsymbol{\theta}_b) \right] = \langle \boldsymbol{\eta} \rangle_b . \end{aligned}$$

The equalities in the second line follow from the definition of expectation.

The final step of the derivation is to determine the values of $\boldsymbol{\tau}$, $\boldsymbol{\mu}$, and $\boldsymbol{\Gamma}$ that minimise (4.23) by setting the corresponding partial derivatives to 0. (For $\boldsymbol{\mu}$ and $\boldsymbol{\Gamma}$, we have to use Langrange multipliers to enforce the constraints $\sum_{r\in\mathcal{K}_C}\mu_c(r)=1$, $\sum_{s\in\mathcal{K}_P}\mu_p(s)=1$, and $\sum_{s\in\mathcal{K}_P}\gamma_{sx}=1$.) After some calculations, we find the following update equations for these parameters:

$$\boldsymbol{\tau}_b = \boldsymbol{\tau}_0 + \langle \boldsymbol{T}\rangle_b\,, \qquad b\in\mathcal{B}\,, \tag{4.24}$$

$$\gamma_{sx} = \frac{\sum_{p\in\mathcal{N}_P}\mu_p(s)\delta(x_p,x)}{\sum_{p\in\mathcal{N}_P}\delta(x_p,x)}\,, \qquad s\in\mathcal{K}_P, x\in\mathcal{L}_P\,, \tag{4.25}$$

$$\mu_c(r) \propto \mu_0(r)\exp\left(\sum_{b\in\mathcal{B}}\frac{\partial\langle\boldsymbol{T}\rangle_b}{\partial\mu_c(r)}\cdot\langle\boldsymbol{\eta}\rangle_b\right), \qquad c\in\mathcal{N}_C, r\in\mathcal{K}_C\,, \tag{4.26}$$

$$\mu_p(s) \propto \gamma_{s,x_p}\exp\left(\sum_{b\in\mathcal{B}}\frac{\partial\langle\boldsymbol{T}\rangle_b}{\partial\mu_p(s)}\cdot\langle\boldsymbol{\eta}\rangle_b\right), \qquad p\in\mathcal{N}_P, s\in\mathcal{K}_P\,. \tag{4.27}$$

For equations (4.26) and (4.27), it is assumed implicitly that the parameters are normalised such that $\sum_{r\in\mathcal{K}_C}\mu_c(r)=1$ and $\sum_{s\in\mathcal{K}_P}\mu_p(s)=1$. Note also that because the three sets of parameters are interdependent, we need to alternate between updating $\boldsymbol{\tau}_b$ and γ_{sx} with $\mu_c(r)$ and $\mu_p(s)$ fixed and updating $\mu_c(r)$ and $\mu_p(s)$ with $\boldsymbol{\tau}_b$ and γ_{sx} fixed. (This is Aicher et al.'s approach in [3], and it is in the spirit of the EM algorithm from Section 4.2.1.) Furthermore, the update equations for $\mu_c(r)$ and $\mu_p(s)$ are themselves recursive, which introduces a second, nested iteration within the outer iteration of the algorithm.

Finally, note that the formula (4.25) for the parameters γ_{sx} is identical to the one obtained in (4.10) for unweighted networks, with the quantities $\mu_c(r)$ and $\mu_p(s)$ playing the role of q_r^c and q_s^p.

4.3 Number of Blocks

Both methods from Section 4.2 require as inputs the number of customer blocks (K_C) and the number of product blocks (K_P). For our applications to shopping networks from Section 4.4, the task of determining the number of blocks does not play a significant role. (We explain our reasons for de-emphasising this aspect in that section.) However, because inferring the number of blocks is such an important problem in SBM inference (as well as in many other forms of data clustering), it is useful to discuss possible ways to address it, particularly for bipartite networks.

For most applications, it would be ideal to infer the number of blocks from the given data. For weighted networks, Aicher et al. proposed using Bayes factors [61] for this purpose; these can be approximated easily by evaluating the functionals $\mathcal{G}(Q)$ for different models.[9] For unweighted networks, simply comparing likelihood values is insufficient, as models with a larger number of blocks have a correspondingly larger number of parameters, and thus they are bound to fit data better. A solution is to use Bayesian inference. In this section, we extend the approach by Riolo et al. [129] (which built on their preceding work in [102]) to determine the number of blocks in a bipartite network. With this method, we infer a posterior distribution $\mathbb{P}(K_C, K_P|\boldsymbol{A})$ for the number of blocks, given an unweighted adjacency matrix $\boldsymbol{A}$. To avoid overfitting, it is key to impose priors on the number of blocks (and on the other model parameters) that penalise more complicated models.

Inferring the number of blocks in annotated networks is more difficult than doing so in networks without annotations, because the prior distributions on block assignments are more complicated in the former situation. In particular, the approach from [129] (which the authors claimed to supersede the earlier method from [102]) cannot be extended easily to annotated networks. We therefore focus on the problem of inferring the number of blocks in bipartite networks without annotations, and we briefly comment on annotated settings in Section 4.3.2.

4.3.1 Networks Without Annotations

In this section, we adapt the approach from [129] for bipartite networks. We start by writing the likelihood under the bipartite degree-corrected SBM described in Section 2.4.2. To be consistent with [129], and to simplify calculations considerably, we assume that edges are generated using a Poisson model rather than a Bernoulli model. In the limit of large, sparse networks (see Section 2.1), the two models give nearly identical results [122, 171]. For the Poisson model, the number of edges between two nodes $c \in \mathcal{N}_C$ and $p \in \mathcal{N}_P$ follows a Poisson distribution with mean $\xi_c \xi_p \theta_{g_c g_p}$. The degree parameters ξ_c and ξ_p are normalised such that

$$\sum_{c \in \mathcal{N}_C} \xi_c \delta(g_c, r) = n_r \,, \quad r \in \mathcal{K}_C \,, \tag{4.28}$$

$$\sum_{p \in \mathcal{N}_P} \xi_p \delta(g_p, s) = n_s \,, \quad s \in \mathcal{K}_P \,, \tag{4.29}$$

[9] This approach does require one to run the inference algorithm for different choices of K_C and K_P, and this is computationally expensive. It is preferable to have a faster algorithm for determining only the number of blocks (and not also the other model parameters), after which one can use a slower algorithm to infer block structure when the number of blocks is fixed.

where n_r and n_s denote the number of nodes in blocks r and s, respectively, for a given $\boldsymbol{g}$. The likelihood of a particular adjacency matrix $\boldsymbol{A}$ under this model is

$$\begin{aligned}\mathbb{P}(\boldsymbol{A}|\boldsymbol{\Theta},\boldsymbol{\xi},\boldsymbol{g},K_C,K_P) &= \prod_{\substack{c\in\mathcal{N}_C,\\ p\in\mathcal{N}_P}} \left(\xi_c\xi_p\theta_{g_cg_p}\right)^{A_{cp}} e^{-\xi_c\xi_p\theta_{g_cg_p}} \\ &= \prod_{c\in\mathcal{N}_C}\xi_c^{d_c}\prod_{p\in\mathcal{N}_P}\xi_p^{d_p}\prod_{(r,s)\in\mathcal{K}_C\times\mathcal{K}_P}\theta_{rs}^{m_{rs}}e^{-n_rn_s\theta_{rs}}\,, \end{aligned} \tag{4.30}$$

where m_{rs} is the number of edges between blocks r and s.

For the purpose of determining the number of blocks, the parameters $\boldsymbol{\Theta}$ and $\boldsymbol{\xi}$ are not important, so we can integrate them out after imposing priors on their values:

$$\mathbb{P}(\boldsymbol{A}|\boldsymbol{g},K_C,K_P) = \int_{\boldsymbol{\xi}}\int_{\boldsymbol{\Theta}}\mathbb{P}(\boldsymbol{A}|\boldsymbol{\Theta},\boldsymbol{\xi},\boldsymbol{g},K_C,K_P)\mathbb{P}(\boldsymbol{\xi}|\boldsymbol{g},K_C,K_P)\mathbb{P}(\boldsymbol{\Theta}|\boldsymbol{g},K_C,K_P)d\boldsymbol{\Theta}d\boldsymbol{\xi}\,. \tag{4.31}$$

Note that this formulation assumes implicitly that the priors for $\boldsymbol{\Theta}$ and $\boldsymbol{\xi}$ are conditionally independent, given the number of blocks and the block assignments.

As in [129], we assume a uniform prior for $\boldsymbol{\xi}$, subject to the constraints in equations (4.28)–(4.29). For each block r (of either customers or products), this amounts to sampling the vector $(\xi_c)_{g_c=r}$ uniformly from the regular simplex with n_r vertices. This choice yields the following prior:

$$\mathbb{P}(\boldsymbol{\xi}|\boldsymbol{g},K_C,K_P) = \prod_{r\in\mathcal{K}_C}(n_r-1)!\prod_{s\in\mathcal{K}_P}(n_s-1)!\,. \tag{4.32}$$

For the edge-propensity parameters $\boldsymbol{\Theta}$, Riolo et al. [129] used exponential priors $\mathbb{P}(\theta_{rs}) = e^{-\theta_{rs}/p}/p$. The only difference in the bipartite case is that p, defined as the mean edge probability observed in a network, is given by $p = m/N_CN_P$, where m is the observed number of edges in the network. This gives the prior

$$\mathbb{P}(\boldsymbol{\Theta}|\boldsymbol{g},K_C,K_P) = \prod_{(r,s)\in\mathcal{K}_C\times\mathcal{K}_P}\frac{1}{p}e^{-\theta_{rs}/p}\,. \tag{4.33}$$

We note that this choice of prior does not favour diagonally-dominant (i.e., assortative) structures of the matrix $\boldsymbol{\Theta}$.

Using the priors (4.32) and (4.33) to evaluate the integral (4.31), the integrated likelihood becomes (up to multiplicative constants)

$$\mathbb{P}(\boldsymbol{A}|\boldsymbol{g},K_C,K_P) \propto \prod_{r\in\mathcal{K}_C}\frac{n_r^{\kappa_r}(n_r-1)!}{(n_r+\kappa_r-1)!}\prod_{s\in\mathcal{K}_P}\frac{n_s^{\kappa_s}(n_s-1)!}{(n_s+\kappa_s-1)!}\prod_{(r,s)\in\mathcal{K}_C\times\mathcal{K}_P}\frac{m_{rs}!}{(pn_rn_s+1)^{m_{rs}+1}}\,. \tag{4.34}$$

In this expression, κ_r is the sum of degrees of customer nodes in block r, with κ_s defined analogously for products.

Using Bayes' rule, we can now obtain the posterior probability for the number of blocks:

$$\mathbb{P}(K_C, K_P|\boldsymbol{A}) = \sum_{\boldsymbol{g}} \mathbb{P}(\boldsymbol{g}, K_C, K_P|\boldsymbol{A}) \propto \sum_{\boldsymbol{g}} \mathbb{P}(\boldsymbol{A}|\boldsymbol{g}, K_C, K_P)\mathbb{P}(\boldsymbol{g}, K_C, K_P)\,. \quad (4.35)$$

The missing piece in this expression is the joint prior $\mathbb{P}(\boldsymbol{g}, K_C, K_P)$. Let $\boldsymbol{g}_C$ and $\boldsymbol{g}_P$ denote the vectors of customer and product block assignments, respectively. We assume that the priors $\mathbb{P}(\boldsymbol{g}_C, K_C)$ and $\mathbb{P}(\boldsymbol{g}_P, K_P)$ are independent; this is reasonable, as it amounts to assuming that block assignments of customer nodes provide no information about the partitioning of product nodes (or vice versa) before observing the network. With this observation, we can use the prior distributions derived for non-bipartite networks in [129] for both $\mathbb{P}(\boldsymbol{g}_C, K_C)$ and $\mathbb{P}(\boldsymbol{g}_P, K_P)$. Ignoring multiplicative constants, this gives

$$\mathbb{P}(\boldsymbol{g}, K_C, K_P) \propto \frac{\prod_{r\in\mathcal{K}_C} n_r!}{(N_C-2)^{K_C}} \frac{\prod_{s\in\mathcal{K}_P} n_s!}{(N_P-2)^{K_P}}\,. \quad (4.36)$$

For details on how this prior was derived, see [129]. We make two observations here about this prior. The first is that $\boldsymbol{g}_C$ and $\boldsymbol{g}_P$ are sampled uniformly, conditioned on the number of blocks and on the block sizes n_r. Sampling these vectors uniformly without first conditioning on block sizes would produce partitions with blocks of similar sizes, which is unrealistic for many real-world networks [118]. The second observation is that for large networks, the prior on the number of blocks approximately satisfies $\mathbb{P}(K_C) \propto 1/(K_C-1)!$ [129] (and similarly for K_P), which heavily penalises models with a larger number of blocks. As mentioned before, this is crucial to avoid overfitting.

We now have all of the ingredients to determine the posterior in equation (4.35). However, the sum over $\boldsymbol{g}$ has a number of terms that is exponential in the number of nodes, making it infeasible to compute exactly even for small networks. Instead, Riolo et al. [129] proposed a Markov Chain Monte Carlo (MCMC) scheme to approximate this posterior distribution numerically. In Algorithm 4.1, we provide the pseudocode for our adaptation of this scheme to bipartite networks. The Markov chain consists of two types of steps, which Riolo et al. [129] called type-1 and type-2 moves. In a type-1 move, one first samples two blocks r and s uniformly at random, and one then selects a node uniformly at random from block r to move to block s. Note that if block r only has one node, this move reduces the number of blocks by 1; otherwise, the number of blocks remains unchanged. In a type-2 move, one samples a block r uniformly

Algorithm 4.1: MCMC algorithm for estimating $\mathbb{P}(K_C, K_P|\boldsymbol{A})$.

```
function NumberOfBlocks(A)
  initialise g
  for i = 1, ..., NumSweeps do
    for j = 1, ..., N_C + N_P do
      X = Bernoulli(N_C / (N_C + N_P))
      if X = 1 then
        (n, k) = (N_C, K_C)                    ▷ Update customer blocks
      else
        (n, k) = (N_P, K_P)                    ▷ Update product blocks
      end if
      Y = Bernoulli(1 − 1/(n − 1))
      if Y = 1 then                            ▷ Type-1 move
        if k = 1 then
          continue                             ▷ Nothing to do
        end if
        sample r, s from 1, ..., k             ▷ Pick two blocks r and s
        sample u from 1, ..., n_r              ▷ Pick node in block r
        compute acceptance probability p_acc
        Z = Bernoulli(p_acc)
        if Z = 1 then
          move u to s
          if r is empty then
            k = k − 1
          end if
        end if
      else                                     ▷ Type-2 move
        sample r from 1, ..., k                ▷ Pick a block r
        sample u from 1, ..., n_r              ▷ Pick node in block r
        compute acceptance probability p_acc
        Z = Bernoulli(p_acc)
        if Z = 1 then
          move u to new block k + 1
          k = k + 1
        end if
      end if
    end for
  end for
  return K_C, K_P from each sweep
end function
```

at random, and then selects a node uniformly at random from block r to move to a newly-created block. Thus, a type-2 move generally increases the number of blocks by 1 (unless there is initially only one node in block r). For a bipartite network, one follows a similar sampling procedure. An additional step at the beginning of each move is to select whether that move affects customer nodes or product nodes. As indicated in Algorithm 4.1, one chooses a step that affects customer nodes with probability $N_C/(N_C + N_P)$, and one chooses a step that affects product nodes with

probability $N_P/(N_C + N_P)$. As explained in [129], it is sufficient to show that the proposed MCMC algorithm satisfies *ergodicity* and *detailed balance* [108] to ensure that it samples from the target posterior distribution $\mathbb{P}(K_C, K_P|\boldsymbol{A})$. One can show that both of these conditions hold for Algorithm 4.1, as long as one chooses suitable values for the acceptance probabilities p_{acc}. (These values depend on the integrated likelihood $\mathbb{P}(\boldsymbol{A}|\boldsymbol{g}, K_C, K_P)$.) See [102, 129] for more details on this for the unipartite setting; the bipartite case is similar.

4.3.2 Networks With Annotations

Inferring the number of blocks is more difficult in the presence of annotations. The issue resides in how one performs update steps within the MCMC algorithm for estimating the posterior $\mathbb{P}(K_C, K_P|\boldsymbol{A})$. A key feature of this algorithm is that at any step, there are no empty blocks; whenever the last node is removed from a block, one immediately decreases K_C or K_P by 1. When annotations are involved, however, it is possible (and, in fact, likely) that nodes with a particular label occupy only a subset of the available blocks. Suppose that we want to move a node with label x to an existing block r that currently has no other nodes with that label. In the language from Section 4.3.1, such a move is neither of type 1 nor of type 2 when restricting the set of nodes to those with label x.

We have found it much more straightforward to modify the original method of Newman and Reinert [102] (which preceded [129]), which allows empty blocks. The main change to the MCMC algorithm in [109] to adapt it for annotated networks is an additional step in which one first samples a label x (with probability proportional to the number of nodes with that label) and then selects a node with annotation x on which to perform a type-1 or type-2 move.

4.4 Applications to Shopping Data

4.4.1 Network Construction

We now apply the inference algorithms that we described in this chapter to the transactions from store `S1` (see Section 3.2.3) during the time period July–September 2015. We annotate product nodes with their Level 2 (`L2`) product categories (see Section 3.1.2), which offer a balance between category labels that are too generic (Level 3) and ones that are too specific (Level 1); we impose no annotations on the customer nodes. We use either item-penetration or basket-penetration weighting for

the edges (see Section 3.2.2). The resulting network has 853 customer nodes, 3705 product nodes, and 151 distinct annotations.

To use the method from Section 4.2.1, which works on unweighted networks, we apply thresholds corresponding to different percentiles of the weight distribution [162]. The results in this section use a 50^{th}-percentile threshold on the item-penetration weights, thus keeping half of the edges from the original weighted network. Additional results for different choices of threshold and for basket-penetration weighting are in Appendix A.2.

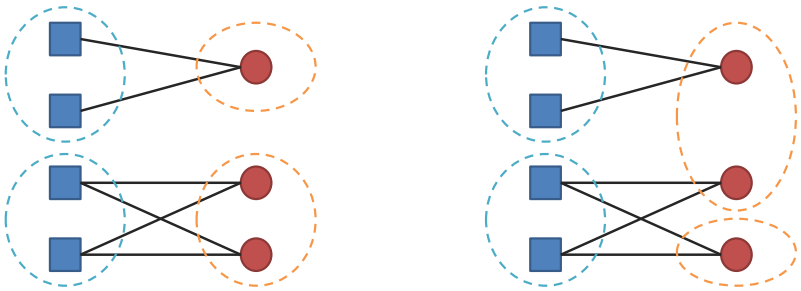
When applying the algorithm from Section 4.2.2, we parametrise the weights using lognormal distributions, which we found to provide better fits than other exponential-family distributions (e.g., Beta, Gamma, Weibull, and Pareto) that are supported on the nonnegative real numbers.

4.4.2 Significance of Annotations

Before performing inference on these annotated networks, we assess whether the available metadata is, in fact, informative about network structure at a statistically-significant level. Extending the "Blockmodel Entropy Significance Test" (BESTest) from [113] to bipartite networks, we compare the entropy[10] of our consumer–product network when nodes are partitioned according to the true metadata with the entropy values that we obtain for different permutations of these annotations (see Figure 4.2). If only a small fraction of the networks with shuffled metadata yield smaller entropy values than the true metadata (i.e., if the empirically-computed p-value is small), then we can reject the null hypothesis and establish that the annotations are indeed meaningful.

Because our shopping networks have annotations only on the product nodes, there are two choices to make when calculating entropies: (1) how to partition the customer nodes; and (2) whether to randomise this partition alongside the product-node annotations when computing p-values. We consider five different ways to partition the customers, three of which are based on the purchasing segmentations discussed in Section 3.1.2, one that consists of the customer communities identified in one run of the GenLouvain algorithm, and one in which all customer nodes are assigned to a single community. In each of these cases, we calculate a p-value for when the customer permutation is fixed and one for when the customer permutation is also randomised.

[10]Rather than calculating and comparing entropies, [113] showed that it is sufficient to compare likelihood values corresponding to one's model of choice for the network structure. We use the bipartite, degree-corrected SBM described in Section 2.4.2 for this calculation.



(a) Smaller entropy (b) Larger entropy

Figure 4.2: Toy bipartite network illustrating two ways to partition the product nodes (indicated by red disks). (a) This partition has a relatively small entropy, because it provides a good compression of the network data: the blocks induce two complete bipartite subgraphs that constitute the entire network. (b) The entropy is larger than in (a) if the product nodes are partitioned differently.

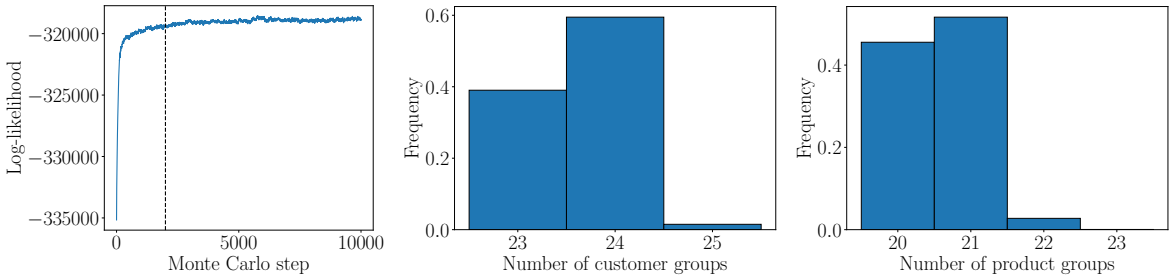
For each calculation, we consider 10^5 different permutations. Apart from the case in which all customers are assigned to a single community, every other test yields an estimated p-value smaller than 10^{-5}, so using the true product-category annotations gives a better compression of the network than any of the permuted annotations. This suggests that `L2` product categories are informative of the network's mesoscale structure, so it makes sense to take them into account for inference.

4.4.3 Results

Number of Blocks

We use Algorithm 4.1 to infer the number of blocks in unweighted and unannotated consumer–product networks. In Figure 4.3, we show the outputs of this algorithm when using item-penetration weighting and a 50^{th}-percentile threshold, as described in Section 4.4.1. (See Table A.1 in Appendix A for results that we obtained for basket-penetration weighting and for different choices of the threshold.) We run 10,000 Monte-Carlo sweeps (the default number in the original code from [129]), and we discard the first 2,000 iterations. Discarding some of the initial iterations (the so-called "burn-in period") is common practice in MCMC, allowing the chain to reach equilibrium and sample from the correct posterior distribution. Choosing an optimal length for the burn-in period is a complex issue that is outside the scope of this work. We justify our choice to discard the first 2,000 iterations by plotting the value of the posterior log-probability at each Monte Carlo step, as in Figure 4.3(a). We see that, beyond the burn-in period, this log-probability is approximately constant. In Figures 4.3(b) and 4.3(c), we show histograms of the number K_C of customer blocks and the number K_P

of product blocks in the remaining 8,000 Monte Carlo samples. The modes of these distributions correspond to $K_C = 24$ customer blocks and $K_P = 21$ product blocks. We obtain similar values for a wide choice of thresholds, as indicated in Table A.1.



(a) Log-probability (b) Number of customer blocks (c) Number of product blocks

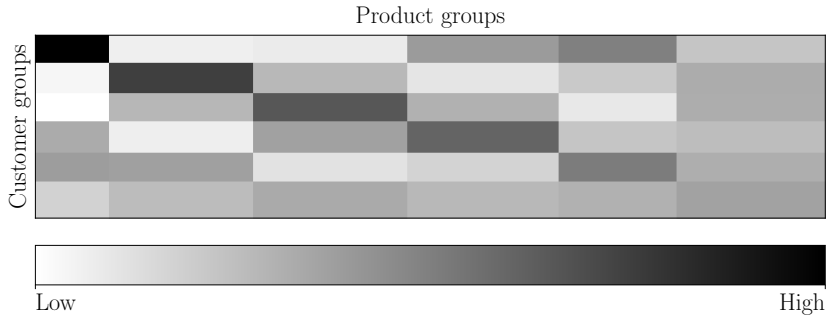
Figure 4.3: Results of Algorithm 4.1 on the customer–product network described in Section 4.4.1. (a) We run 10,000 Monte Carlo steps and discard the first 2,000 samples (the burn-in period, indicated by the dashed vertical line). (b) Distribution of K_C, the number of customer blocks. (c) Distribution of K_P, the number of product blocks.

Unfortunately, the belief-propagation algorithm from Section 4.2.1 does not converge in 800 iterations (to within a reasonably small numerical tolerance) for such large values of K_C and K_P. The log-likelihood continues to increase at each iteration, but it does so extremely slowly. This behaviour is somewhat expected, as the algorithm from Section 4.2.1 runs in $\mathcal{O}(K_C K_P)$ time; thus, each time we double both K_C and K_P, the algorithm runs about 4 times more slowly. Therefore, throughout the rest of this chapter, we examine results for $K_C = K_P = 6$. (An additional benefit of analysing results for a smaller number of blocks is that interpreting the outputs is significantly easier.) In Appendix A.3, we provide results for when K_C and K_P are set equal to the mode values that we infer using Algorithm 4.1.

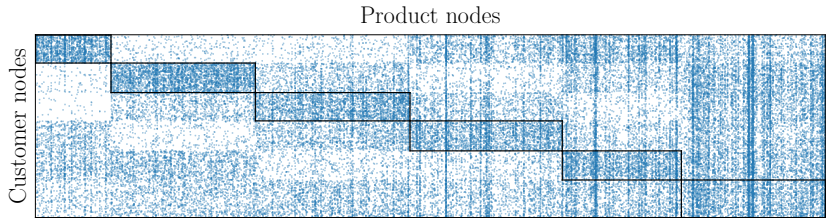
Partitions for Unweighted Networks

Setting $K_C = K_P = 6$, we run the algorithm from Section 4.2.1 for inference in annotated bipartite networks. This algorithm has three sets of outputs: the $K_C \times K_P$ matrix $\boldsymbol{\Theta}$ of edge-propensity parameters; the marginal probabilities q_r^c (respectively, q_s^p) for a customer (respectively, product) to belong to each block; and the $K_P \times L_P$ matrix $\boldsymbol{\Gamma}$ that relates the annotations on the product nodes to the inferred blocks.

The matrix $\boldsymbol{\Theta}$ summarises the patterns of connection between blocks inferred by the algorithm. Figure 4.4(a) shows this 6×6 matrix as a heatmap, with darker shades indicating larger probabilities for nodes in the respective blocks to form edges.



(a) Matrix of edge propensities



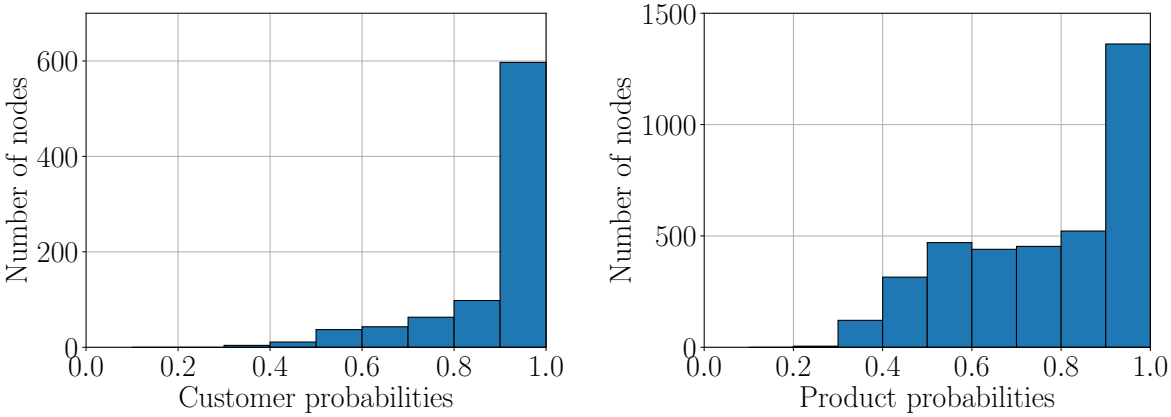
(b) Reordered adjacency matrix

Figure 4.4: Interaction patterns of $K_C = 6$ customer blocks and $K_P = 6$ product blocks. (a) Heatmap of $\boldsymbol{\Theta}$ values, with darker shades indicating higher propensities for forming edges between nodes in the corresponding blocks. (b) Sparsity plot of the adjacency matrix (before thresholding), where we have reordered its rows and columns based on a hard partitioning of the nodes obtained from taking the most likely block assignment for each customer and product. The adjacency matrix is significantly sparser than it appears in this plot; only about 3% of the entries are nonzero.

Figure 4.4(b) shows a sparsity plot of the adjacency matrix, in which we have reordered the nodes based on their most likely block assignments: the first rows correspond to customers most likely to be in block 1, followed by customers most likely to be in block 2, and so on. In both panels of Figure 4.4, the sizes of the rectangular blocks are proportional to the number of customer and product nodes that they contain. These visualisations reveal an approximately assortative structure,[11] with larger values of θ_{rs} along the diagonal than off the diagonal. It is important to note that we did not constrain the algorithm to search for such assortative structure, as we use a general SBM to fit to the data. Another pattern revealed by Figure 4.4(a) is that the diagonal entries vary in magnitude, with the strongest interaction occurring between the first block of customers and the first block of products. In contrast, the sixth product block appears to be a "catch-all" set of products that connect with roughly equal

[11]Recall from Section 2.4.2 (and from Figure 2.3) that assortativity for a bipartite network means that each customer block interacts preferentially with one product block, such that grouping these together would produce traditional communities of densely-connected sets of nodes.

probabilities to each of the customer blocks. A final noteworthy feature is the low level of interaction between the third customer group and the first product group, revealed clearly in both Figure 4.4(a) and Figure 4.4(b).



(a) Histogram of $\max_{r \in \mathcal{K}_C} q_r^c$ (customer nodes)

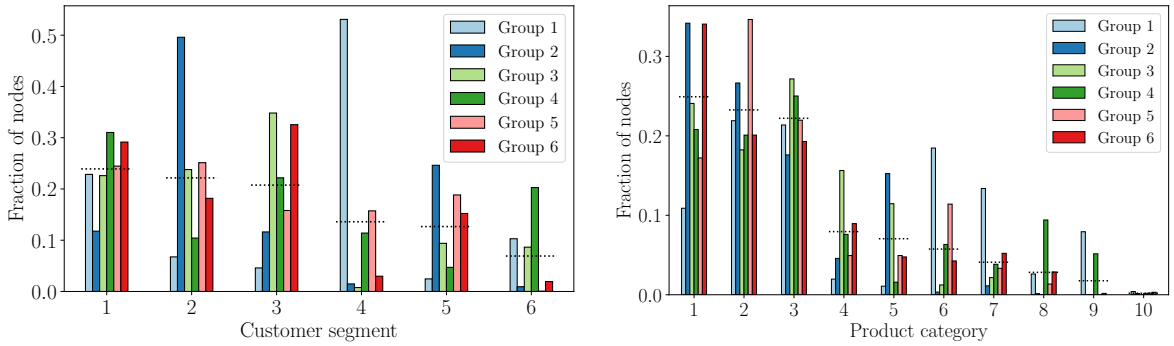
(b) Histogram of $\max_{s \in \mathcal{K}_P} q_s^p$ (product nodes)

Figure 4.5: Distributions of the largest block-membership probabilities for (a) customers and (b) products. These plots suggest that block assignments are more uncertain for products than for customers.

The algorithm from Section 4.2.1 also returns a "soft" partitioning of customer and product nodes into the specified number of blocks. Unlike for "hard" partitions, such as the ones that result from most of the methods from Chapter 2, a node can belong with nonzero probabilities to more than one block. These probabilistic block assignments are not a result of the algorithm searching for overlapping communities. Rather, this uncertainty arises from integrating over the posterior distribution to obtain the marginal probabilities q_r^c and q_s^p. (If we instead took the mode of this distribution as our solution, we would once again obtain a hard partition, in which every node is assigned to exactly one block.) As expected, one can go from a soft partition to a hard one by setting $g_c = \text{argmax}_{r \in \mathcal{K}_C} q_r^c$ for customer nodes $c \in \mathcal{N}_C$ and $g_p = \text{argmax}_{s \in \mathcal{K}_P} q_s^p$ for product nodes $p \in \mathcal{N}_P$.

In Figure 4.5, we show the distributions of the largest probabilities $q_{\max}^u = \max_{t \in \mathcal{K}} q_t^u$ separately for customers and products. Values in these histograms that are closer to 1 indicate more certainty than do smaller values concerning the optimal block assignments of the corresponding nodes. (Note that, by construction, the probabilities in these histograms are bounded below by $1/K_C$ for customers and by $1/K_P$ for products.) About 70% of customer nodes are assigned with probability 90% or higher to their best block , whereas only around 36% of products belong to their best block

with the same level of certainty. This suggests that clustering products based on the structure of the underlying bipartite network is more difficult than clustering customers. In fact, as we explain in Section 4.4.4, the former task would have been even harder without the annotations.



(a) Comparison to existing customer segments

(b) Comparison to existing product categories

Figure 4.6: (a) Proportion of customer nodes in each block that fall in each `purchasing-medium` segment. Bars of the same colour are scaled so that their heights add to 1. The horizontal dashed lines indicate the proportion of customers in the data set that belong to each segment (irrespective of block assignments). (b) Proportion of product nodes in each block that fall in each `L3` product category.

We can characterise the blocks identified by our algorithm by comparing them with existing metadata. Specifically, we compare customer blocks with the `purchasing-medium` segmentation,[12] and we compare product blocks with `L3` product categories. (See Section 3.1.2 for a reminder of what these represent.) To make this comparison, we first obtain a hard partition of the nodes into blocks by setting $g_c = \text{argmax}_{r \in \mathcal{K}_C} q_r^c$ for customer nodes and $g_p = \text{argmax}_{s \in \mathcal{K}_P} q_s^p$ for product nodes. The bar chart from Figure 4.6(a) shows, for each of the 6 algorithmically-inferred customer blocks, the distribution of nodes across the 6 `purchasing-medium` segments. We observe that more than 50% of customers in block 1 belong to a category of upscale customers (segment 4 in the figure). Similarly, nearly 50% of customers in block 2 are part of a price-sensitive category (segment 2 in the figure). Customers in the remaining blocks are also over-represented in some purchasing segments and under-represented in others. Figure 4.6(b) gives a similar comparison for product nodes. Products in block 1 — which are primarily purchased by block-1 customers — are over-represented across produce, meat, and deli counters `L3` categories (categories 6, 7, and 9 in the

[12]Note that this segmentation is built using all purchases of a particular customer (not just those in the specific store that we analyse in this chapter).

figure). Products in block 2 — which are primarily purchased by block-2 customers — are over-represented in a broad L3 category that includes snacks, soft drinks, and ready meals (category 1 in the figure) and in a category of frozen items (category 5 in the figure). These observations paint a similar picture to the one from Section 3.3.1, where we used modularity maximisation to examine community structure for a small convenience store. In both cases, the two distinct groups of upscale and price-sensitive consumers emerged very clearly. Additionally, in both cases, we found that the former category is more likely to buy fresh ingredients for cooking at home, whereas the latter category is more strongly associated with snacks and frozen items. All of these qualitative observations also hold for other threshold values and for basket-penetration weighting (see Appendix A.2).

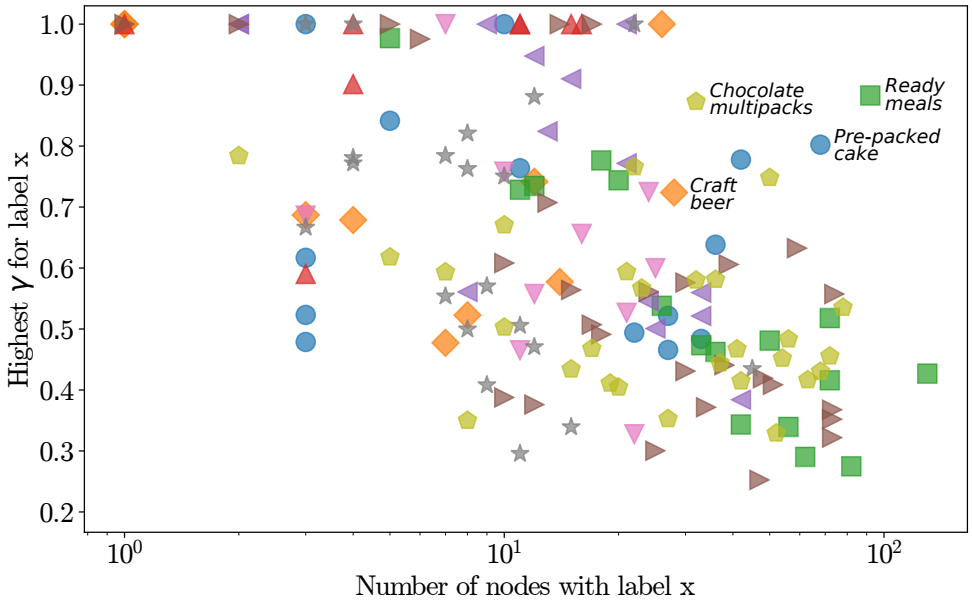


Figure 4.7: Scatter plot of the largest γ_{sx} for each L2 label x and $s \in \mathcal{K}_P$. The horizontal axis indicates the number of products ($n_x = \sum_{p \in \mathcal{N}_P} \delta(x_p, x)$) with the respective label. Markers of the same shape and colour correspond to L2 labels that are in the same L3 product category. (For example, blue disks correspond to the bakery category.) In the upper-right corner of the plot, we highlight a subset of labels that are informative of the network's block structure.

The final output of the inference algorithm is the matrix $\boldsymbol{\Gamma}$, which summarises how the annotations relate to the network's mesoscale structure. (This output is the most important one for gaining insights that are not accessible using simpler methods, such as K-means clustering of customers or products.) Recall that γ_{sx} is the prior probability that a product node with category label x belongs to community s. Labels

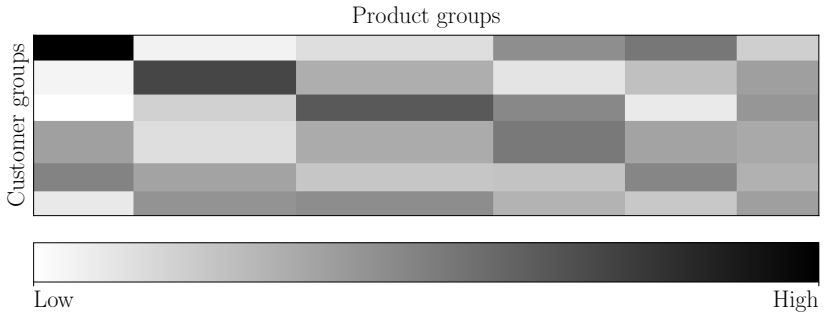
for which γ_{sx} is close to 1 for one particular block $s \in \mathcal{K}_P$ therefore indicate that products with that label are associated strongly to that block. In Figure 4.7, we show a scatter plot of the largest prior probability for each label; we denote this probability by $\gamma_x^{\max} = \max_{s \in \mathcal{K}_P} \gamma_{sx}$. On the horizontal axis, we indicate the number of products that have each label; this number is $n_x = \sum_{p \in \mathcal{N}_P} \delta(x_p, x)$. In general, we expect $\gamma_x^{\max}$ to decrease as n_x increases: when there are more products with a particular label, it is less likely that they are all associated strongly to the same block s. We observe this general downward trend in Figure 4.7, although there is considerable variance around this mean behaviour.

Figure 4.7 is a way to map different product-category annotations and identify ones that are especially informative of the given network's block structure. These are labels that appear near the top-right corner of the scatter plot (see Figure 4.7) and labels with $\gamma_x^{\max}$ very close to 1 (even if n_x is not particularly large for this latter category). We highlight the ready-meals `L2` category; even though there are close to 100 products with this annotation, they all belong to one block with a prior probability of about 88%. Other informative labels include herbs ($\gamma_x^{\max} = 1$, $n_x = 22$), frozen poultry ($\gamma_x^{\max} = 1$, $n_x = 26$), and almost all deli-counter subcategories (indicated using red triangles in Figure 4.7), such as fresh fish. All of these labels provide ideal opportunities for targeting a new product.

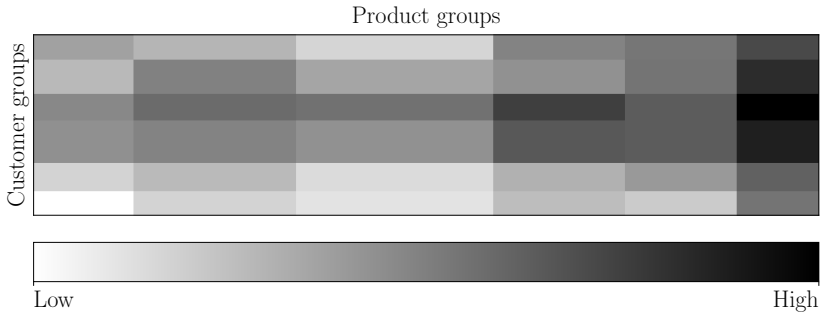
Partitions for Weighted Network

We now apply the inference algorithm from Section 4.2.2, which works on weighted networks (thus removing the need for a threshold). We continue to set $K_C = K_P = 6$. To obtain good results, we have found it useful to initialise the algorithm with the output probabilities q_r^c and q_s^p of the unweighted inference algorithm, rather than using a flat prior $\mu_c^{(0)}(r) = 1/K_C$ and $\mu_p^{(0)}(s) = 1/K_P$. (We note a similar approach in [44], where the authors used a uniform prior to generate an initial partition, which they then used to "seed" the algorithm in subsequent runs.)

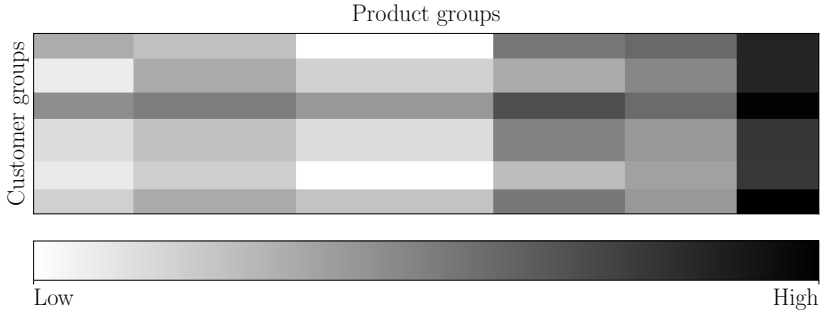
In Figure 4.8, we show the three sets of SBM parameters that are learned by the algorithm: the edge propensities $\boldsymbol{\Theta} = (\theta_{rs})$ for nodes in different blocks to form connections with each other; the mean parameters μ_{rs} of the lognormal distributions that parametrise the weights for different edge bundles; and the variance parameters σ_{rs}^2 of the same lognormal distributions. The matrix in Figure 4.8(a) tends to have larger values along the diagonal than off the diagonal (especially for the first three customer and product blocks), similarly to the corresponding matrix from Figure 4.4(a) that we inferred for an unweighted variant of the network. Also noteworthy is that edges



(a) Edge propensities θ_{rs} (degree-corrected Poisson model)



(b) Mean parameters μ_{rs} of the weight distributions (lognormal model)



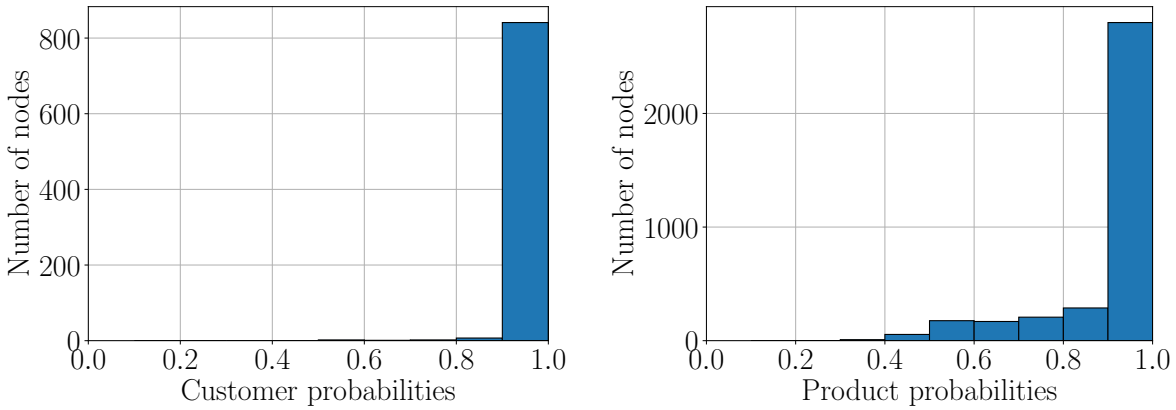
(c) Variance parameters σ^2_{rs} of the weight distributions (lognormal model)

Figure 4.8: SBM parameters inferred using the variational Bayesian algorithm from Section 4.2.2 in a consumer–product network with item-penetration weighting. We used a degree-corrected Poisson model for the edge distributions and lognormal distributions to model the weights.

between any of the customer blocks and the sixth product block have especially large weights with large standard deviations, as indicated by the large values in the last columns of panels (b) and (c). We conclude that this block corresponds to popular products that customers from all 6 blocks buy in large quantities.

In the two panels of Figure 4.9, we show histograms of the largest block-membership probabilities across all customer nodes and across all product nodes. Compared to the analogous plots from Figure 4.5 for the unweighted network, both of these distributions

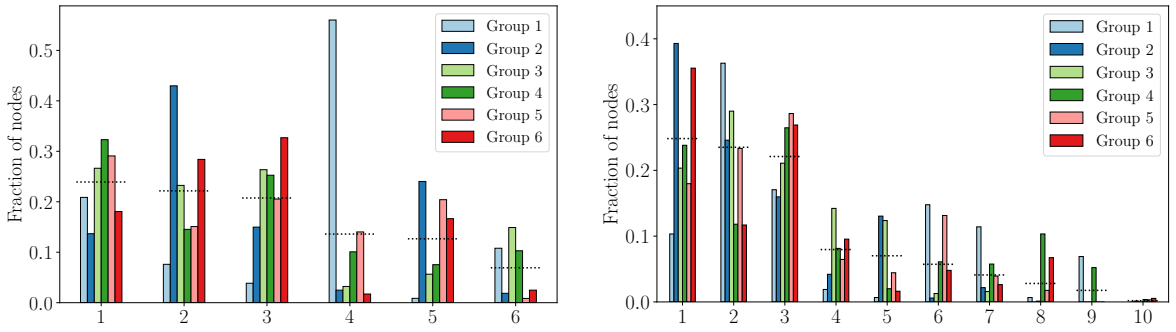
are shifted to the right. This suggests a greater certainty in the assignment of nodes to blocks with the weighted algorithm. Comparing panels (a) and (b) of Figure 4.9, it again appears that there is more uncertainty in the block assignments of products than in those of customers, although the difference is less pronounced than in Figure 4.5.



(a) Histogram of $\max_{r \in \mathcal{K}_C} q_r^c$ (customer nodes)

(b) Histogram of $\max_{s \in \mathcal{K}_P} q_s^p$ (product nodes)

Figure 4.9: Distributions of the largest block-membership probabilities for (a) customers and (b) products. These plots suggest that block assignments are more uncertain for products than for customers.



(a) Comparison to existing customer segments

(b) Comparison to existing product categories

Figure 4.10: (a) Proportion of customer nodes in each block that fall in each `purchasing-medium` segment. The horizontal dashed lines indicate the proportion of customers in the data set that belong to each segment (irrespective of block assignments). (b) Proportion of product nodes in each block that fall in each `L3` product category.

In Figure 4.10, we compare the inferred customer and product blocks with existing customer segments and product categories. This plot has similar features to the analogous one (Figure 4.6) for the unweighted setting. For example, customer blocks

1 and 2 overlap with upscale and price-sensitive segments, respectively. A large proportion of block-1 products are produce, meat, and deli counter items; and a large proportion of block-2 products are frozen items.

We make a few brief remarks about the advantages of performing inference in weighted, rather than unweighted, consumer–product networks. For weighted networks, it is no longer necessary to specify a threshold for removing smaller-weight edges. Performing inference in weighted networks also provides insights that are not accessible when only analysing their unweighted counterparts; for instance, we found from Figure 4.8(b) that products in block 6 are purchased in larger quantities than products in the other blocks. Despite these advantages, we have found that the weighted algorithm performs poorly on our shopping data (see also discussion in Section 4.5), unless we initialise it using block-membership probabilities inferred using the unweighted algorithm. For this reason, our analysis of shopping networks in this chapter (and in Appendix A) focuses primarily on unweighted settings.

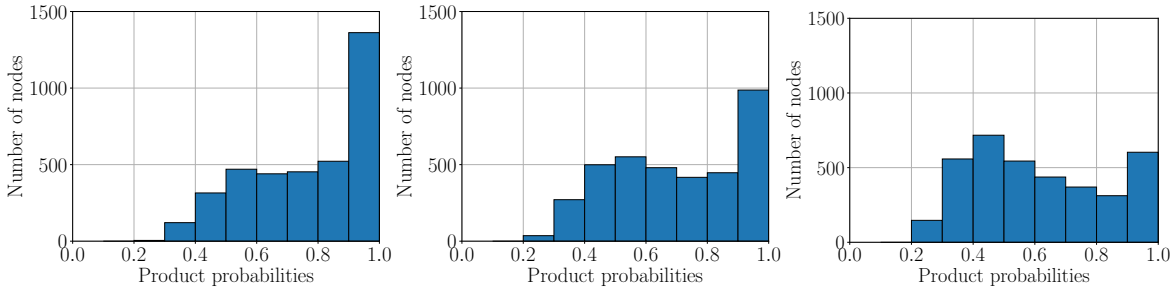
4.4.4 Impact of Annotations

In this section, we assess the impact of annotations on the solution of our unweighted SBM by comparing it to the solutions that we obtain for two different "null" networks: one in which there are no annotations on the product nodes and one in which the original labels are randomly permuted (keeping the same frequency counts of different labels) to remove any correlation between annotations and network structure.[13] Crucially, the algorithm still has access to the same bipartite network structure in these two cases, so it still detects meaningful customer and product blocks. In fact, the matrix $\boldsymbol{\Theta}$ of edge propensities retains a similar structure to the one from Figure 4.4(a).

The most notable impact of randomising or removing annotations is on the distribution of the largest block-membership probabilities for each node ($q^c_{\max} = \max_{r\in\mathcal{K}_C} q^c_r$ and $q^p_{\max} = \max_{s\in\mathcal{K}_P} q^p_s$). For products, these histograms shift significantly to the left when going from the true annotations to the permuted ones, and they also shift when going from permuted annotations to no annotations (see Figure 4.11); for customer nodes, there is no such significant change (see Figure 4.12).

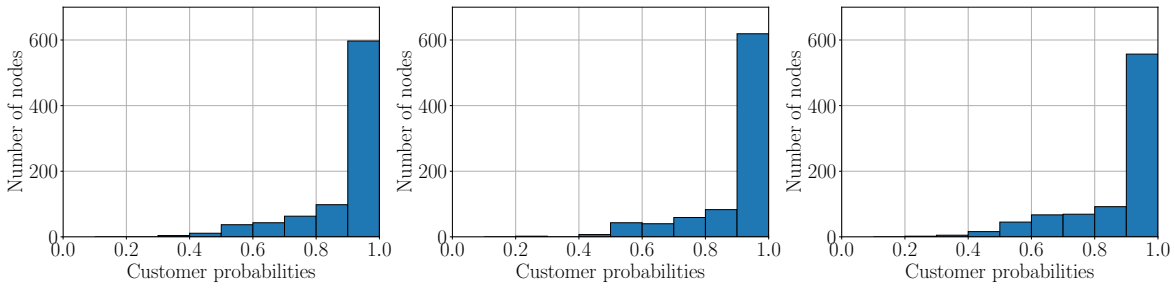
Table 4.2 gives the pairwise NMI scores between the partitions that we obtain for these three networks, calculated both across all nodes and separately for customers and products. The main observation about these values is that the two most-similar

[13]For the latter, there are many possible permutations, each of which yields a different null network. In this section, we show results for one such randomisation, but we checked that the same qualitative conclusions hold for several other permutations.



(a) True annotations (b) Permuted annotations (c) No annotations

Figure 4.11: Histograms of $\max_{s \in \mathcal{K}_P} q_s^p$ (for product nodes $p \in \mathcal{N}_P$) for (a) the true annotations; (b) one sample of permuted annotations; and (c) no annotations. Observe that the distribution shifts to the left when going from (a) to (b) and from (b) to (c), towards solutions with more uncertainty.



(a) True annotations (b) Permuted annotations (c) No annotations

Figure 4.12: Histograms of $\max_{r \in \mathcal{K}_C} q_r^c$ (for customer nodes $c \in \mathcal{N}_C$) for (a) the true annotations; (b) one sample of permuted annotations; and (c) no annotations. Unlike in Figure 4.11, these distributions do not change significantly as a result of randomising or removing the annotations.

partitions are the one with permuted annotations and the one with no annotations. This suggests that the algorithm finds structure that is especially different when the network incorporates true annotations compared to networks where only the topology is meaningful (i.e., because annotations are either missing or random).

4.5 Summary and Discussion

Our goal in this chapter was to analyse mesoscale structures in bipartite networks that incorporate additional data in the form of node annotations. Our starting point was the method proposed by Newman and Clauset [109] that we extended to bipartite networks. This method consists of a degree-corrected SBM in which node annotations affect the prior probabilities of different block partitions. The parameters

Table 4.2: NMI between partitions that we obtain using true, permuted, or no annotations.

	Partition 1	Partition 2	NMI
All nodes	true	permuted	0.50
	true	no annotations	0.56
	permuted	no annotations	0.62
Customers	true	permuted	0.64
	true	no annotations	0.66
	permuted	no annotations	0.74
Products	true	permuted	0.31
	true	no annotations	0.38
	permuted	no annotations	0.46

that describe this prior are determined by an inference algorithm along with the other model parameters, which include probabilistic assignments of nodes to blocks.

The method by Newman and Clauset [109] only works for unweighted networks. Therefore, we also combined the weighted SBM from [3] with the annotated prior from [109] to obtain an algorithm which works on weighted networks. In principle, this algorithm should yield better results on our weighted networks than the first algorithm does on thresholded versions of the same network data. However, we obtain mixed results, unless we use the solution of the unweighted algorithm to initialise the weighted one. These computational issues are not restricted to our data. The original code also fails to recover planted communities in some synthetic networks that are well inside the detectable regime; in these cases, the algorithm converges to an uninformative solution in which each node is equally likely to belong to each of the blocks. We believe that the main issue is that the code implements a first-order method (variational Bayes), in which the posterior distribution over block assignments factorises as a product of the probabilities $\mu_c(r)$ and $\mu_p(s)$. In reality, block memberships of nodes that are adjacent or otherwise connected via short paths in a network are not independent. By contrast, belief propagation is a second-order method, so one also considers the two-vertex marginals q_{rs}^{cp}. Thus, a method based on belief propagation is bound to be more accurate.[14]

[14] Aicher et al. [3] also implemented a version of their algorithm that uses BP, but they only did so for a subset of cases that does not include degree-corrected SBMs for sparse networks.

Another practical issue is that of computational efficiency. The algorithms from [3, 109] both scale as $\mathcal{O}(NK_CK_P)$ (and as $\mathcal{O}(NK^2)$ for non-bipartite networks). This leads to very slow convergence when we set $K_C \approx 20$ and $K_P \approx 20$, the numbers of blocks inferred using Algorithm 4.1. It is interesting to note that although belief-propagation methods were considered to be much more efficient than Monte-Carlo sampling just a few years ago [109, 163], it appears that the trend is reversing [65, 118, 129]. Thus, there is potential to develop faster algorithms for inference in annotated networks in the future.

Despite these computational challenges, our analysis of consumer–product networks using the methods from this chapter has led to several important findings. The first is that clustering products appears to be more difficult than clustering customers, as evidenced by the greater uncertainty of the block assignments of product nodes in the posterior distribution. The second, related finding is that including annotations helps reduce uncertainty in product block assignments, suggesting another benefit of incorporating metadata into statistical models of networks. (We expect that this previously-unknown behaviour also occurs in other empirical networks.) A third lesson is that different product categories vary considerably in how informative they are about a shopping network's structure. Although this is not especially surprising, it does mean that using annotated networks to address the "cold-start" problem of recommender systems (as described in Section 4.1) will work significantly better for some product categories (e.g., because they are strongly tied to just one block of customers) than for others. Finally, using general SBMs to examine the structure of these networks revealed an approximately assortative structure, as each customer block generally has strong ties to one product block but weaker ties to the remaining blocks. This finding suggests that methods such as modularity maximisation, which explicitly seek assortative structure, are adequate for these bipartite networks (although one still generally expects to learn more about a network by using an SBM).

The myriad recent algorithms for detecting mesoscale structures in annotated networks [30, 65, 109, 144, 161, 165, 169], the lack of agreement in the literature between the modelling assumption that annotations influence communities and the opposite assumption that communities influence annotations, and the mixed computational performance of some of these existing methods suggest that inference in annotated networks is still an open problem. Our contribution suggests extensions of some of these approaches to bipartite networks, and it also clarifies some of the ways in which incorporating annotations in a statistical network model can help improve results.

Chapter 5

Community Detection in Multilayer Networks

5.1 Motivation

Many real-world data sets are better represented as a collection of interrelated networks rather than as a single graph. Common examples include temporal networks [18, 63, 121], in which edges and/or nodes change in time, and multiplex networks (a generalisation of edge-coloured multigraphs), in which edges correspond to different types of interactions [98, 157]. A less-studied scenario (and one that we examine in detail) from the perspective of community detection is that of multilevel networks, in which layers comprise a hierarchy of monolayer networks and interlayer edges between nodes indicate inclusion relationships.

Detecting communities in multilayer networks is an active area of research. There are now several methods from a variety of approaches [34, 36, 74, 98, 116, 145] that have been employed on numerous applications. For temporal networks, multilayer community detection has been applied to the study of brain networks [20], financial correlations [18], scientific citations [64], biological contagions [135], and more. For multiplex networks, examples of previous studies include investigations of genes with different types of interactions [25], social support networks in Indian villages [34], and microbial interactions at different sites in the human body [145].

Multilayer network methods can also be used for studying customer behaviour. Because consumer preferences change over time, it is natural to use temporal networks to represent transactions over several (potentially overlapping) time windows. Additionally, one can use multiplex networks to represent the same set of nodes and edges in multiple layers, but with different weightings along the edges to emphasise distinct ways in which a particular transaction is important (e.g., number of items

purchased in the time window versus frequency of purchase). Lastly, one can use multilevel networks to encode transactions at different levels of aggregation, with customers connected to products in one layer and connected to product categories (originating from a given hierarchy, as in Section 3.1.2) in additional layers.

In all of these instances, it is important to note that the connectivity patterns in different layers tend to be highly interdependent. In terms of the examples above, a customer who buys yoghurt in one time period is likely to do so in the future, a customer who buys tea frequently most likely buys large amounts of tea relative to other customers, and a customer who buys strawberries has (by definition) made purchases in the "berries" product category. Detecting communities separately in each layer does not account for these interlayer dependencies, leading to intralayer partitions that vary significantly from one layer to the next. Additionally, single-layer methods require additional post-processing if one wishes to match communities that are detected in different layers. Therefore, they are not ideally suited to detect meaningful interlayer variations in the mesoscale organisation of multilayer networks.

The rest of this chapter is organised as follows.[1] In Section 5.2, we review two classes of methods for community detection in multilayer networks: ones that extend standard modularity maximisation for single-layer networks and ones that are based on SBM inference. In examining these existing approaches, we indicate some of their shortcomings that we aim to address in this chapter. Our contributions are twofold. First, in Section 5.3, we generalise work by Newman on monolayer networks [105], and we show that the problem of maximising modularity in multilayer networks is closely related to the problem of fitting particular multilayer SBMs to network data. We derive versions of this equivalence for various types of multilayer structure, including temporal, multilevel, and multiplex networks. This result sheds light on some of the assumptions that are implicit in multilayer modularity maximisation. Second, as a consequence of this result, we propose in Section 5.4 a principled method for determining two key sets of parameters for performing multilayer modularity maximisation. The selection of these tuning parameters has been a longstanding difficulty of multilayer modularity-maximisation algorithms. We test our approach on several synthetic networks (both temporal and multiplex) in Section 5.4.3, and we test it on a small multiplex social network in Section 5.4.4. We then apply our optimised modularity-maximisation algorithm to temporal and multilevel consumer–product networks in Section 5.5. We summarise our findings and conclude with some ideas for future work in Section 5.6.

[1] Most of this work has also appeared in [111], where I am the lead author.

5.2 Existing Approaches

5.2.1 Modularity Maximisation

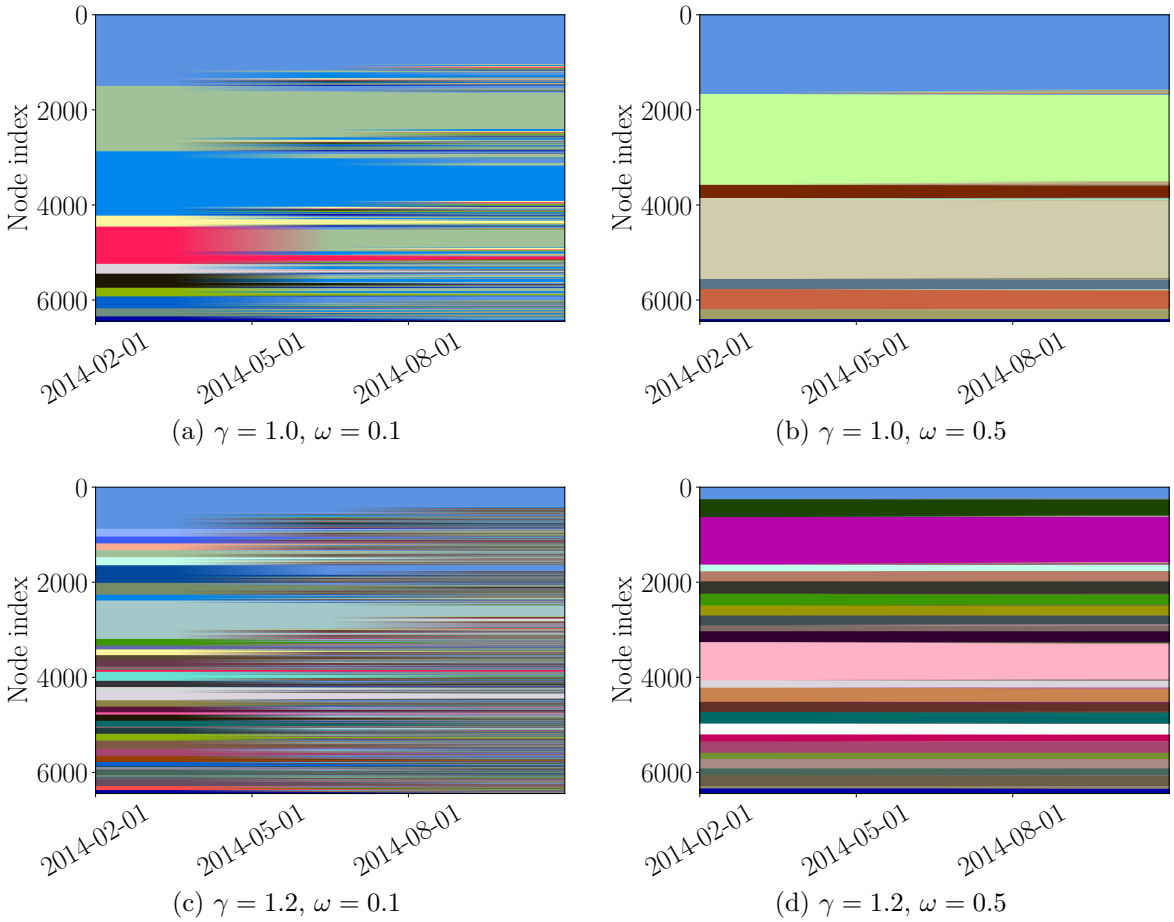
We briefly recall the formulation of multilayer modularity introduced by Mucha et al. [98] that we discussed in more detail in Section 2.4.1. We study fully-interconnected multilayer networks with diagonal interlayer couplings (see Section 2.3). Let T denote the number of layers in a given network, and let N denote the number of nodes in each layer. We write $\boldsymbol{A}^t$ for the $N \times N$ adjacency matrix of the network in layer $t \in \{1, \ldots, T\}$. Analogously to the single-layer case, multilayer modularity is a function that measures the quality of a multilayer partition $\boldsymbol{g}$ of the set of nodes into communities, relative to some null model. For an undirected, unipartite network with the Newman–Girvan null model [110], Mucha et al. [98] derived the following expression for multilayer modularity:

$$\begin{aligned} Q(\boldsymbol{g}) &= \sum_{t=1}^{T} \sum_{i,j=1}^{N} \left(A_{ij}^t - \gamma_t \frac{d_i^t d_j^t}{2m_t} \right) \delta(g_i^t, g_j^t) + \sum_{t=1}^{T} \sum_{s \neq t} \sum_{i=1}^{N} \omega_i^{st} \delta(g_i^s, g_i^t) \\ &= Q_{\text{intra}}(\boldsymbol{g}) + Q_{\text{inter}}(\boldsymbol{g}) \,. \end{aligned} \tag{5.1}$$

As a reminder, d_i^t is the degree of node i in layer t, the quantity m_t is the number of edges in layer t, and g_i^t is the community label of node i in layer t.

One of the main drawbacks of using modularity maximisation for community detection is that its definition involves two sets of unknown parameters: the resolution values γ_t and the interlayer coupling weights ω_i^{st}. To reduce the size of this parameter space, it is common to assume both that all layers have identical resolution parameters γ and that all interlayer edges have the same weight ω. Even in this simplified situation, determining appropriate values for γ and ω remains an open problem. To illustrate the importance of this issue, we show in Figure 5.1 visualisations of multilayer communities detected in a temporal consumer–product network constructed from purchases in store `S2` for four choices of (γ, ω). In general, we expect the number of communities to increase with the resolution γ, and the level of interlayer persistence, measured as the fraction of nodes that do not change communities across consecutive layers, to increase with the coupling ω. The panels of Figure 5.1 follow this general pattern. Additionally, they also show that different parameter choices can result in vastly different outputs. A key question, therefore, is how should one select parameter values that are "optimal" in some sense.

An initial idea might be to choose γ and ω to maximise the modularity in equation (5.1). However, in most cases, this simply suggests choosing small resolution values (to



(a) $\gamma = 1.0$, $\omega = 0.1$ (b) $\gamma = 1.0$, $\omega = 0.5$

(c) $\gamma = 1.2$, $\omega = 0.1$ (d) $\gamma = 1.2$, $\omega = 0.5$

Figure 5.1: "Drip plot" of multilayer community structure detected in a temporal consumer–product network with three layers for four different choices of γ and ω. In these visualisations, rows correspond to nodes, columns correspond to layers, and colours indicate community labels. These plots show that the values of γ and ω can have a substantial effect on detected communities, and it is not clear which partition offers a better summary of the underlying network's mesoscale structure.

maximise intralayer modularity contributions) and large coupling values (to maximise the interlayer term). To get around this issue, the authors of [15] chose γ and ω values that maximised the difference between the modularity Q and the modularity Q^{null} of a network without community structure. (For example, one can obtain the latter by permuting intralayer edges in the original network while keeping node degrees fixed.) This procedure revealed "cohesive regions" in the (γ, ω) plane where a given network is significantly more modular than a random network with similar characteristics. However, this method requires a large number of calculations, because it performs modularity maximisation several times for each choice of γ and ω. For large networks,

this type of grid search is prohibitive. In addition, different null models may identify different γ and ω values at which the difference $Q - Q^{\text{null}}$ is maximal, and knowing which of these is "correct" poses yet another challenge.

A different approach for choosing modularity parameters was recently proposed by Weir and collaborators in [158]. Their method takes as an input a set of partitions and then discards all "non-admissible" partitions, namely those that are dominated by others for any choice of resolution and interlayer coupling. The pruned set of partitions is several orders of magnitude smaller than the initial one. From among the remaining partitions, one can choose those that have the largest domains of optimality in (γ, ω) space to analyse further. The downside of this method is that it requires a large number of input partitions (the authors used between 10,000 and 240,000 runs for the three applications that they studied), which poses challenges for large networks. In situations with more than two resolution and coupling parameters, the number of runs needs to be even larger to adequately sample the space of multilayer partitions.

Another recent paper considered the problem of choosing parameter values for multilayer modularity maximisation [4]. The authors' goal was to propose formulas for γ and ω that depend on an input network and a multilayer partition of its set of nodes into communities. This is an ambitious objective; however, their work has several limitations. In writing down expressions for γ and ω, the authors gave little justification for how they obtained these formulas. For instance, the resolution γ depends on a "redundancy" measure that counts the fraction of edges that appear in more than one layer. In their definition, γ may only take values in $(0, 1] \cup \{2\}$; they provided no justification for excluding the interval $(1, 2)$ or values above 2. Similarly, their definition of the interlayer coupling ω does not allow values that are larger than 1, which is especially inadequate in light of the fact that the $\omega \to \infty$ limit is known to be important [149]. Another shortcoming is that the authors did not use their method for choosing γ and ω for community detection. Instead, they showed that their proposed parameter choices lead to larger values of the modularity Q compared to other choices when the multilayer partition is fixed; they provided no evidence that their methodology leads to "better" detected community structures in either synthetic or real networks.

To summarise, there is currently no systematic methodology to determine appropriate values for resolution and interlayer coupling parameters that appear in formulations of multilayer modularity. This is a significant drawback of community-detection methods that are based on modularity maximisation.

5.2.2 Statistical Models

Stochastic block models and other generative models of networks with planted mesoscale structures have also been extended to multilayer settings. However, one shortcoming of existing SBMs is that they restrict how mesoscale structure can vary across layers, and many multilayer SBMs require that communities be identical across all layers [116, 121, 150, 168].

Several authors have relaxed this assumption in various ways. Focusing on temporal multilayer networks, Ghasemian et al. [51] described a Markov model in which nodes move to new communities between successive layers with probabilities that are given by a transition matrix. Because their goal was to study detectability thresholds [97] in these networks, they made several simplifications that reduce the utility of their model for inferring mesoscale structures in real-world networks. For example, their model produces networks in which all communities have the same expected size and all nodes have the same expected degree.

The degree-corrected, mixed-membership generative model in De Bacco et al. [34] assumed that block structure is identical across layers, but these blocks can induce different types of structures in different layers (e.g., assortative, disassortative, core–periphery, and so on). Stanley et al. [145] proposed a model that allows different layers to have different communities. However, their algorithm involves a clustering step that assigns layers to "strata", such that each stratum is described by a single SBM.[2] Consequently, their model does not include an explicit mechanism for generating different communities in different layers. Vallès-Català et al. [153] proposed a model in which different layers can have different communities, but they assumed that one observes only an aggregate monolayer network. Consequently, it is not possible to fit this model to multilayer network data.

In a recent paper, Bazzi et al. [17] proposed a general probabilistic model of multilayer networks that allows arbitrary community structures, although they did not develop their model as a way to undertake community detection. Their work yields families of networks on which to test algorithm performance, as it provides a "forward model" with extremely flexible planted mesoscale structures. The "inverse problem" of finding communities that are likely to have generated an observed multilayer network remains unsolved in this general setting. Nevertheless, the ideas of Bazzi et al. play

[2] A similar type of approach appeared in [67], where the authors aimed to simultaneously partition a temporal network into a series of contiguous time segments and to detect a "consensus" community structure within each segment. Their algorithm is mechanistic (based on optimising quality functions) rather than statistical.

an important role in the derivation of our results in Section 5.3 and in some of our numerical experiments in Section 5.4.3.

In summary, these existing statistical methods cannot be used to infer multilayer mesoscale structure that varies in arbitrary ways between layers. In situations that require this level of generality, modularity-maximisation methods are more appropriate.

5.3 Relating Modularity and SBMs

In this section, we present new results that relate multilayer modularity maximisation to inference in suitably-chosen multilayer SBMs. We start by reviewing Newman's work in [105] that first established such a connection for monolayer networks. In Section 5.3.2, we describe a generative model whose posterior probability recovers the multilayer modularity function (5.1). We then particularise this result to temporal, multilevel, and multiplex networks in Sections 5.3.3, 5.3.4, and 5.3.5, respectively. Throughout this section, we work primarily with unweighted networks, although we briefly discuss possible extensions of our results to weighted networks in Section 5.3.6.

Although we primarily study applications to bipartite networks in this thesis, our results in this section apply more broadly. For simplicity, we work with undirected, unipartite networks throughout Sections 5.3 and 5.4. One can easily derive analogous results for directed or bipartite networks by making the necessary changes to the modularity function (see Section 2.4.1) and to the SBM in each layer (see Section 2.4.2). In fact, we apply our method to directed networks in Section 5.4.4 and to bipartite consumer–product networks in Section 5.5.

5.3.1 Monolayer Networks

Modularity maximisation and inference in SBMs are two of the most widely used methods for identifying mesoscale structure in networks [48]. In [105], Newman uncovered a surprising connection between these two methods. We briefly summarise his argument before considering the more general multilayer setting in the rest of this section.

Recall that modularity is a function that measures the quality of a partition $\boldsymbol{g}$, which assigns nodes to communities. Larger values of modularity correspond to partitions in which more edges fall within groups than would be expected by chance, as quantified by a null model. For undirected monolayer networks, the version of modularity that is used most often employs the Newman–Girvan null

model [110] (which is closely related to configuration models [49]). In this case, the modularity function compares the entries of a given network's adjacency matrix $\boldsymbol{A}$ to the expectation of the entries after uniformly randomly rewiring the edges such that the expected degrees of the nodes are preserved:

$$Q(\boldsymbol{g}) = \sum_{i,j=1}^{N} \left(A_{ij} - \gamma \frac{d_i d_j}{2m} \right) \delta(g_i, g_j) \,, \tag{5.2}$$

where N is the number of nodes in the network, d_i is the degree of node i (and d_j is the degree of node j), m is the number of edges, and γ is the resolution parameter (see Section 2.4.1). By tuning γ, one can detect coarser or finer community structure in networks.

It is this modularity that Newman compared with likelihood functions that arise from statistical network models [105]. Following his derivation, consider a degree-corrected SBM that is parametrised by the matrix $\boldsymbol{\Theta} = (\theta_{rs})_{1 \leq r,s \leq K}$, where K is the number of communities in a network. Although in SBMs one typically needs to assume that the number of communities is given (or learned through some other means, such as Bayesian inference [129]), knowing the correct value of K is not necessary for establishing the results of this section. The parameters θ_{rs} indicate the propensities with which nodes in community r connect to nodes in community s. Suppose that the number of edges between two nodes, i and j, follows a Poisson distribution with mean $d_i d_j \theta_{g_i g_j}/(2m)$. The observed adjacency matrix $\boldsymbol{A}$ (i.e., the data), given parameters $\boldsymbol{\Theta}$ and $\boldsymbol{g}$, then has likelihood

$$\mathbb{P}(\boldsymbol{A}|\boldsymbol{\Theta}, \boldsymbol{g}) = \prod_{i,j=1}^{N} \frac{(d_i d_j \theta_{g_i g_j}/2m)^{A_{ij}}}{A_{ij}!} e^{-d_i d_j \theta_{g_i g_j}/(2m)} \,. \tag{5.3}$$

Taking logarithms of both sides of equation (5.3) yields

$$\log \mathbb{P}(\boldsymbol{A}|\boldsymbol{\Theta}, \boldsymbol{g}) = \sum_{i,j=1}^{N} \left[A_{ij} \log \frac{d_i d_j}{2m} + A_{ij} \log \theta_{g_i g_j} - \log(A_{ij}!) - \frac{d_i d_j}{2m} \theta_{g_i g_j} \right] . \tag{5.4}$$

If we know the parameters $\boldsymbol{\Theta}$, our goal is to maximise the log-likelihood (5.4) with respect to the partition vector $\boldsymbol{g}$. We can ignore any terms in the right-hand side that do not depend on $\boldsymbol{g}$ and write

$$\log \mathbb{P}(\boldsymbol{A}|\boldsymbol{\Theta}, \boldsymbol{g}) = \sum_{i,j=1}^{N} \left(A_{ij} \log \theta_{g_i g_j} - \frac{d_i d_j}{2m} \theta_{g_i g_j} \right) + (\text{const.}) \,. \tag{5.5}$$

Following [105], we consider a particular type of SBM known as a "planted-partition model" (PPM) [31]. In this model, the parameters θ_{rs} take one of two values: θ_{in} for

$r = s$ and θ_{out} for $r \neq s$. When $\theta_{\text{in}} > \theta_{\text{out}}$, the PPM generates networks with assortative communities, for which the density of edges is higher within blocks of nodes than it is between blocks. We may now write

$$\begin{aligned} \theta_{g_i g_j} &= (\theta_{\text{in}} - \theta_{\text{out}})\delta(g_i, g_j) + \theta_{\text{out}} \,, \\ \log \theta_{g_i g_j} &= (\log \theta_{\text{in}} - \log \theta_{\text{out}})\delta(g_i, g_j) + \log \theta_{\text{out}} \,. \end{aligned} \tag{5.6}$$

Substituting equations (5.6) into equation (5.5) yields

$$\begin{aligned} \log \mathbb{P}(\boldsymbol{A}|\boldsymbol{\Theta}, \boldsymbol{g}) = (\log \theta_{\text{in}} - \log \theta_{\text{out}}) \sum_{i,j=1}^{N} \left(A_{ij} - \frac{\theta_{\text{in}} - \theta_{\text{out}}}{\log \theta_{\text{in}} - \log \theta_{\text{out}}} \frac{d_i d_j}{2m} \right) \delta(g_i, g_j) \\ + (\text{const.}) \,. \end{aligned} \tag{5.7}$$

The connection between the modularity in equation (5.2) and the log-likelihood in equation (5.7) now becomes clear. Both of them are functions of $\boldsymbol{g}$ that take the same form (up to additive and multiplicative constants), provided we set

$$\gamma = \frac{\theta_{\text{in}} - \theta_{\text{out}}}{\log \theta_{\text{in}} - \log \theta_{\text{out}}} \,. \tag{5.8}$$

In particular, the optimal community assignments $\boldsymbol{g}$ are the same for both of these functions; maximising one is the same as maximising the other.[3]

The above comparison between modularity maximisation and maximum-likelihood estimation for the PPM is useful for two reasons. First, it reveals some features of modularity that otherwise are not clear, such as the fact that it detects communities that are "statistically equivalent" in the sense that they are characterised by the same parameters θ_{in} and θ_{out}. Thus, modularity maximisation may not perform as well on networks with communities that have very different internal edge densities, even if these communities are assortative. Second, this argument suggests that choosing the resolution γ as in equation (5.8) is a reasonable default when detecting communities using modularity maximisation. (One can estimate the unknown values θ_{in} and θ_{out} that appear in equation (5.8) using iterative approaches [105].) This is especially relevant when one is interested in choosing a single resolution value for a network rather than exploring multiscale community structure [138].

We remark in passing that a similar equivalence holds between modularity with a uniform null model and a non-degree-corrected PPM. More importantly for us, analogous results hold for directed and bipartite networks (both with and without

[3] Of course, because the underlying community-detection problem is NP-hard [24], the two methods can perform very differently in practice, because they converge to different local optima.

degree correction), and the associated resolution value is the same as the one in (5.8). We expect that establishing similar results for weighted networks is considerably more difficult. (We also briefly discuss the weighted case in Section 5.3.6.)

5.3.2 Multilayer Modularity and Multilayer SBMs

We now seek to describe a generative model whose posterior probability recovers the modularity function in equation (5.1). Many different types of multilayer SBMs have been studied [17, 34, 51, 116, 121, 145, 150, 153, 168], and there are many more that one could propose, so we start by pointing out some requirements for such models to be suited for our purpose.

The first key requirement is that the statistical model has a posterior probability over multilayer partitions $\boldsymbol{g}$ that one can write in closed form; otherwise, we are not able to make a meaningful comparison with multilayer modularity. This rules out the general benchmark model in [17], in which one uses Monte Carlo methods to sample from the distribution of multilayer partitions. The second requirement is that the model should allow arbitrary variations in community structure across layers. This is important, because the modularity in equation (5.1) also allows this type of generality. This requirement rules out many existing multilayer SBMs [116, 121, 150, 168]. Finally, we demand that any interdependencies between layers are induced solely through the multilayer partition $\boldsymbol{g}$ and the coupling between nodes in different layers, rather than through intralayer edges. This rules out, for example, the model proposed in [13], which includes an "edge persistence" parameter that favours intralayer edges to remain active over multiple temporal layers. Note that the modularity in equation (5.1) also encapsulates all interlayer dependencies in a separate term $Q_{\text{inter}}(\boldsymbol{g})$, which does not involve any of the intralayer adjacency matrices $\boldsymbol{A}^t$.

With these requirements in mind, we are ready to describe the building blocks of a multilayer SBM that recovers the modularity in equation (5.1). In the process, one has to place further restrictions on the generative model; this is unsurprising, because SBMs allow more general mesoscale structures than the type of communities that are detectable through modularity maximisation. (In the monolayer setting from Section 5.3.1, one such restriction is that the SBM only has two edge propensities, θ_{in} and θ_{out}.)

We assume that edges are placed independently in each layer, conditioned on the multilayer partition $\boldsymbol{g}$. In particular, any coupling between nodes in different layers (through interlayer edges or otherwise) has no direct influence on the placement of edges within each layer. This is in line with many existing multilayer SBMs, including

those in [17, 51], which are the existing models that most closely resemble the ones that we propose in the present work. Using a degree-corrected SBM with a set of parameters $\boldsymbol{\Theta}^t$ for each layer, the conditional log-likelihood of observing a network specified by its supra-adjacency matrix $\boldsymbol{\mathcal{A}}$ (see Section 2.3) is

$$\log \mathbb{P}(\boldsymbol{\mathcal{A}}|\boldsymbol{g}, \boldsymbol{\Theta}) = \sum_{t=1}^{T} \sum_{i,j=1}^{N} \left(A_{ij}^t \log \theta_{g_i^t g_j^t}^t - \frac{d_i^t d_j^t}{2m_t} \theta_{g_i^t g_j^t}^t \right) + (\text{const.}) \,. \tag{5.9}$$

By taking the same approach as in Section 5.3.1, we can relate each term in equation (5.9) to the corresponding intralayer modularity term from equation (5.1). Assuming that the SBM in each layer takes the form of a PPM with two parameters, θ_{in}^t and θ_{out}^t, the log-likelihood becomes

$$\log \mathbb{P}(\boldsymbol{\mathcal{A}}|\boldsymbol{g}, \boldsymbol{\theta}_{\text{in}}, \boldsymbol{\theta}_{\text{out}}) = \sum_{t=1}^{T} \left(\log \theta_{\text{in}}^t - \log \theta_{\text{out}}^t \right) \sum_{i,j=1}^{N} \left(A_{ij}^t - \gamma_t \frac{d_i^t d_j^t}{2m_t} \right) \delta(g_i^t, g_j^t) + (\text{const.}) \,, \tag{5.10}$$

where

$$\gamma_t = \frac{\theta_{\text{in}}^t - \theta_{\text{out}}^t}{\log \theta_{\text{in}}^t - \log \theta_{\text{out}}^t} \,. \tag{5.11}$$

For now, we assume that the SBM parameters θ_{in}^t and θ_{out}^t are the same for all layers. (We discuss the more general setting, in which these parameters can differ across layers, in Section 5.3.3.) In this case, the log-likelihood from equation (5.10) becomes

$$\log \mathbb{P}(\boldsymbol{\mathcal{A}}|\boldsymbol{g}, \theta_{\text{in}}, \theta_{\text{out}}) = (\log \theta_{\text{in}} - \log \theta_{\text{out}}) \sum_{t=1}^{T} \sum_{i,j=1}^{N} \left(A_{ij}^t - \gamma \frac{d_i^t d_j^t}{2m_t} \right) \delta(g_i^t, g_j^t) + (\text{const.}) \,, \tag{5.12}$$

where

$$\gamma = \frac{\theta_{\text{in}} - \theta_{\text{out}}}{\log \theta_{\text{in}} - \log \theta_{\text{out}}} \,. \tag{5.13}$$

The expression in (5.12) now matches the term $Q_{\text{intra}}(\boldsymbol{g})$ from equation (5.1) (up to scaling by an expression that does not depend on $\boldsymbol{g}$). The "optimal" resolution value from (5.13) is, perhaps not surprisingly, the same as the one in equation (5.8) that was derived for the monolayer setting [105]. Note additionally that this derivation holds irrespective of the interdependency structure between the layers and whether it is temporal, multiplex, or something else.

We have now accounted for the intralayer modularity terms in equation (5.1). These terms, as well as the log-likelihood in equation (5.12), lack any form of coupling

between the community assignments $\boldsymbol{g}^t$ in different layers. Without such coupling, maximising either expression with respect to $\boldsymbol{g}$ is no different from performing the optimisation separately for each layer. We introduce layer interdependencies in our generative model through a prior distribution on $\boldsymbol{g}$. This is a prior distribution, because it encodes our belief of what community assignments are probable before observing the network data specified by $\boldsymbol{\mathcal{A}}$. As one might expect, the form of this prior depends markedly on the specific coupling between layers. Therefore, in our subsequent discussions, we can no longer consider the most general representations of multilayer networks. We thus focus on specific types of multilayer structure: we consider temporal networks in Section 5.3.3, multilevel networks in Section 5.3.4, and multiplex networks in Section 5.3.5.

5.3.3 Temporal Networks

In a multilayer representation of a temporal network, each layer encodes interactions that occur in some period of time. Although there are exceptions (e.g., see [152]), it is common to assume that interlayer edges in these temporal networks are "diagonal" (they only connect copies of the same node in different layers), "layer-coupled" (their weight depends only on the layers rather than on the nodes), and "ordinal" (they only connect successive layers) [78]. We also take these interlayer edges to be directed, indicating that information flows along the arrow of time.

We represent these temporal networks using the supra-adjacency matrix from Section 2.3. Recall that diagonal blocks in this matrix correspond to intralayer edges and off-diagonal blocks correspond to interlayer edges. With the simplifications above, each of the off-diagonal blocks is a multiple of the $N \times N$ identity matrix. Let ω_t denote the weight of interlayer edges from layer $t-1$ to layer t, where $t \in \{2, \dots, T\}$. The supra-adjacency matrix $\boldsymbol{\mathcal{A}}$ then has the following structure:

$$\boldsymbol{\mathcal{A}} = \begin{bmatrix} \boldsymbol{A}^1 & \omega_2 \boldsymbol{I} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{A}^2 & \omega_3 \boldsymbol{I} & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \omega_T \boldsymbol{I} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{A}^T \end{bmatrix}.$$

Uniform coupling and resolution parameters

It is common to assume for simplicity that the parameters $\omega_2, \dots, \omega_T$ are all equal to some value ω; this is known as *uniform* coupling [18, 98]. In this case, the expression

(5.1) for multilayer modularity becomes

$$Q(\boldsymbol{g}) = \sum_{t=1}^{T} \sum_{i,j=1}^{N} \left(A_{ij}^{t} - \gamma \frac{d_i^t d_j^t}{2m_t} \right) \delta(g_i^t, g_j^t) + \omega \sum_{t=2}^{T} \sum_{i=1}^{N} \delta(g_i^{t-1}, g_i^t) \,. \tag{5.14}$$

We also assume for now that the resolution parameter γ is the same for all layers; we will relax this assumption later in this section. After specifying values for γ and ω, modularity is a function of the community labels $\boldsymbol{g}$, where g_i^t gives the community assignment of node i in layer t.

We relate the vectors $\boldsymbol{g}^t$ of community assignments for different layers using a prior probability $\mathbb{P}(\boldsymbol{g})$. Following previous approaches for sampling multilayer partitions [17, 51], assume that the community membership of node i in layer t is copied from the previous layer with probability p and is sampled from some specified null distribution $\mathbb{P}_0$ with probability $1-p$. Because of the temporal structure, labels in the first layer are sampled from the null distribution rather than copied from some other layer. The probability of generating a particular multilayer partition $\boldsymbol{g}$ is then

$$\mathbb{P}(\boldsymbol{g}) = \mathbb{P}(\boldsymbol{g}^1) \prod_{t=2}^{T} \mathbb{P}(\boldsymbol{g}^t | \boldsymbol{g}^{t-1}) = \prod_{i=1}^{N} \mathbb{P}_0(g_i^1) \prod_{t=2}^{T} \left\{ \prod_{i=1}^{N} \left[p\delta(g_i^{t-1}, g_i^t) + (1-p)\mathbb{P}_0(g_i^t) \right] \right\}.$$

Taking logarithms and rearranging yields

$$\begin{aligned}
\log \mathbb{P}(\boldsymbol{g}) &= \sum_{i=1}^{N} \log \mathbb{P}_0(g_i^1) + \sum_{t=2}^{T} \sum_{i=1}^{N} \log \left[1 + \frac{p\delta(g_i^{t-1}, g_i^t)}{(1-p)\mathbb{P}_0(g_i^t)} \right] + \sum_{t=2}^{T} \sum_{i=1}^{N} \log \left[(1-p)\mathbb{P}_0(g_i^t) \right] \\
&= \sum_{i=1}^{N} \log \mathbb{P}_0(g_i^1) + \sum_{t=2}^{T} \sum_{i=1}^{N} \log \left[1 + \frac{p}{(1-p)\mathbb{P}_0(g_i^t)} \right] \delta(g_i^{t-1}, g_i^t) \\
&\quad + \sum_{t=2}^{T} \sum_{i=1}^{N} \log \left[(1-p)\mathbb{P}_0(g_i^t) \right] .
\end{aligned} \tag{5.15}$$

The second equality follows from noting that any logarithmic term in the second sum is 0 whenever $\delta(g_i^{t-1}, g_i^t) = 0$. The ability to move the Kronecker delta outside of the logarithmic expression is the key technical trick that allows us to relate this prior probability to the interlayer modularity terms from equation (5.1). This is also the crucial step that makes certain cases, such as multiplex networks (see Section 5.3.5), more difficult to analyse.

Comparing the expression in equation (5.15) with the interlayer modularity terms in equation (5.14), we start to see a resemblance: both involve weighted sums over nodes i and layers t of Kronecker deltas of the form $\delta(g_i^{t-1}, g_i^t)$. To obtain an exact match, we have to make further assumptions. Suppose that the null distribution

$\mathbb{P}_0$ is uniform over community assignments. Specifically, if there are K communities (possibly including empty ones), let $\mathbb{P}_0(g_i^t) = 1/K$ for any node i and layer t, so that each node is equally likely to be assigned to each of the communities. The first and third terms in equation (5.15) become constants, and the only term that depends on $\boldsymbol{g}$ is the second one. From now on, we also write $\log \mathbb{P}(\boldsymbol{g}|p, K)$ for the log-prior to emphasise its dependence on the parameters p and K.

We can now use Bayes' rule to combine intralayer and interlayer terms that arise in our proposed multilayer SBM into a single posterior probability:

$$\begin{aligned}
\log \mathbb{P}(\boldsymbol{g}|\boldsymbol{\mathcal{A}}, \theta_{\text{in}}, \theta_{\text{out}}, p, K) &= \log \frac{\mathbb{P}(\boldsymbol{\mathcal{A}}, \boldsymbol{g}|\theta_{\text{in}}, \theta_{\text{out}}, p, K)}{\mathbb{P}(\boldsymbol{\mathcal{A}}|\theta_{\text{in}}, \theta_{\text{out}}, p, K)} \\
&= \log \frac{\mathbb{P}(\boldsymbol{\mathcal{A}}|\boldsymbol{g}, \theta_{\text{in}}, \theta_{\text{out}})\mathbb{P}(\boldsymbol{g}|p, K)}{\mathbb{P}(\boldsymbol{\mathcal{A}}|\theta_{\text{in}}, \theta_{\text{out}}, p, K)} \\
&= \log \mathbb{P}(\boldsymbol{\mathcal{A}}|\boldsymbol{g}, \theta_{\text{in}}, \theta_{\text{out}}) + \log \mathbb{P}(\boldsymbol{g}|p, K) \\
&\quad - \log \mathbb{P}(\boldsymbol{\mathcal{A}}|\theta_{\text{in}}, \theta_{\text{out}}, p, K)\,. \qquad (5.16)
\end{aligned}$$

The first two terms of (5.16) correspond, respectively, to the log-likelihood from equation (5.12) and the log-prior from equation (5.15). The third term is difficult to compute in practice, but it does not depend on $\boldsymbol{g}$, so we can treat it as a constant for our purposes. We thus obtain

$$\begin{aligned}
&\log \mathbb{P}(\boldsymbol{g}|\boldsymbol{\mathcal{A}}, \theta_{\text{in}}, \theta_{\text{out}}, p, K) \\
&\quad = (\log \theta_{\text{in}} - \log \theta_{\text{out}}) \sum_{t=1}^{T} \sum_{i,j=1}^{N} \left(A_{ij}^t - \gamma \frac{d_i^t d_j^t}{2m_t} \right) \delta(g_i^t, g_j^t) \\
&\qquad + \sum_{t=2}^{T} \sum_{i=1}^{N} \log \left(1 + \frac{p}{1-p} K \right) \delta(g_i^{t-1}, g_i^t) + (\text{const.}) \\
&\quad \propto \sum_{t=1}^{T} \sum_{i,j=1}^{N} \left(A_{ij}^t - \gamma \frac{d_i^t d_j^t}{2m_t} \right) \delta(g_i^t, g_j^t) + \sum_{t=2}^{T} \sum_{i=1}^{N} \frac{\log \left(1 + \frac{p}{1-p} K \right)}{\log \theta_{\text{in}} - \log \theta_{\text{out}}} \delta(g_i^{t-1}, g_i^t) + (\text{const.})\,. \qquad (5.17)
\end{aligned}$$

By maximising the right-hand side of (5.17) with respect to $\boldsymbol{g}$, we find the mode of the posterior distribution over multilayer partitions, given the multilayer network $\boldsymbol{\mathcal{A}}$ and model parameter values θ_{in}, θ_{out}, p, and K. We can now ensure that maximising the multilayer modularity (5.14) is equivalent to maximising the posterior probability (5.17) by choosing the following values for the parameters γ and ω:

$$\gamma = \frac{\theta_{\text{in}} - \theta_{\text{out}}}{\log \theta_{\text{in}} - \log \theta_{\text{out}}}\,, \qquad (5.18)$$

$$\omega = \frac{1}{\log\theta_{\text{in}} - \log\theta_{\text{out}}} \log\left(1 + \frac{p}{1-p}K\right) . \tag{5.19}$$

We show how γ and ω depend on the other parameters in Figure 5.2. It is useful to point out some limiting cases. In this discussion, we focus on the interlayer coupling parameter ω. When $p = 0$, which entails that there is no copying of community assignments from one layer to the next, equation (5.19) gives $\omega = 0$. This implies that community detection proceeds independently for each layer. This limiting case thus behaves as desired. When $p \to 1$, we obtain $\omega \to \infty$, which ensures that maximising modularity recovers the same communities in each layer, as it should.[4] Perhaps less obvious is the dependence on θ_{in} and θ_{out}. Consider the ratio $\epsilon = \theta_{\text{out}}/\theta_{\text{in}}$, which determines the strength of the planted community structure. According to equation (5.19), a weaker community structure in a network (i.e., a larger value of ϵ) should entail a larger value of ω. One explanation is that by over-emphasising persistence, we take advantage of correlations between layers to increase the "signal" of underlying community structure, while simultaneously dialing down layer-specific "noise". Finally, ω increases (weakly) with the number of communities K. For fixed θ_{in} and θ_{out}, it is typically more difficult to detect a larger number of communities than a smaller one (i.e., the detectability threshold moves up [51]). Increasing the amount of coupling improves community-detection results by taking advantage of correlations between layers.

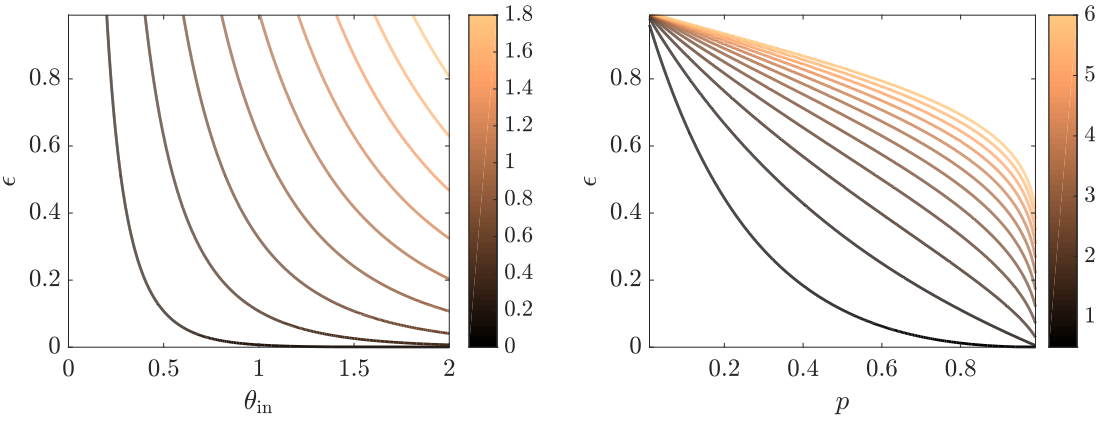
Non-uniform coupling and resolution parameters

Thus far, we have assumed that all layers have the same resolution parameter γ and are connected by interlayer edges with the same weight ω. This amounts to assuming that the layers are "statistically similar", in that they can be described by a common set of parameters θ_{in}, θ_{out}, p, and K. For many applications, this assumption is unrealistic, so it is important to investigate what happens when these parameters are layer-dependent.

Proceeding in analogy to the layer-independent case, the log-likelihood of observing a network $\boldsymbol{\mathcal{A}}$ conditioned on a multilayer partition $\boldsymbol{g}$ is

$$\begin{aligned}&\log\mathbb{P}(\boldsymbol{\mathcal{A}}|\boldsymbol{g},\boldsymbol{\theta}_{\text{in}},\boldsymbol{\theta}_{\text{out}})\\ &\quad= \sum_{t=1}^{T}\left[\left(\log\theta_{\text{in}}^t - \log\theta_{\text{out}}^t\right)\sum_{i,j=1}^{N}\left(A_{ij}^t - \frac{\theta_{\text{in}}^t - \theta_{\text{out}}^t}{\log\theta_{\text{in}}^t - \log\theta_{\text{out}}^t}\frac{d_i^t d_j^t}{2m_t}\right)\delta(g_i^t, g_j^t)\right] + (\text{const.}) .\end{aligned} \tag{5.20}$$

[4] In fact, an optimal multilayer partition has identical communities across layers for any value of ω larger than some critical (finite) value ω_∞ [18].



(a) Optimal γ (b) Optimal ω

Figure 5.2: Contour plots of (a) optimal γ value from equation (5.18) as a function of θ_{in} and $\epsilon = \theta_{\text{out}}/\theta_{\text{in}}$, and (b) optimal ω value from equation (5.19) as a function of copying probability p and ϵ (using $K = 2$; other values give similar results).

To impose a prior on the multilayer partition $\boldsymbol{g}$, let p_t be the probability that a node in layer t copies its community label from layer $t-1$, where $t \in \{2, \dots, T\}$. Recall that $\mathbb{P}_0$ is the null distribution for community labels when nodes do not copy their community assignments from the previous layer. We now make this distribution layer-dependent, so $\mathbb{P}_0^t$ is the null distribution for layer t, to allow layers with non-uniform characteristics. As before, suppose that each null distribution $\mathbb{P}_0^t$ is uniform, which entails that $\mathbb{P}_0^t(g_i^t) = 1/K_t$, where K_t is the number of communities (possibly including empty ones) in layer t. Ignoring any terms that no longer depend on $\boldsymbol{g}$, the log-prior is then

$$\begin{aligned}\log \mathbb{P}(\boldsymbol{g}) &= \sum_{i=1}^{N} \log\left[\mathbb{P}_0^t(g_i^1)\right] + \sum_{t=2}^{T}\sum_{i=1}^{N} \log\left[p_t \delta(g_i^{t-1}, g_i^t) + (1-p_t)\mathbb{P}_0^t(g_i^t)\right] \\ &= \sum_{t=2}^{T}\sum_{i=1}^{N} \log\left(1 + \frac{p_t}{1-p_t} K_t\right) \delta(g_i^{t-1}, g_i^t) + (\text{const.})\,. \end{aligned} \tag{5.21}$$

Combining equations (5.20) and (5.21), we obtain the following posterior probability distribution for $\boldsymbol{g}$:

$$\begin{aligned}&\log \mathbb{P}(\boldsymbol{g}|\boldsymbol{\mathcal{A}}, \boldsymbol{\theta}_{\text{in}}, \boldsymbol{\theta}_{\text{out}}, \boldsymbol{p}, \boldsymbol{K}) \\ &\quad\propto \sum_{t=1}^{T}\left[\frac{\log\theta_{\text{in}}^t - \log\theta_{\text{out}}^t}{\langle \log\theta_{\text{in}}^t - \log\theta_{\text{out}}^t\rangle_t} \sum_{i,j=1}^{N}\left(A_{ij}^t - \frac{\theta_{\text{in}}^t - \theta_{\text{out}}^t}{\log\theta_{\text{in}}^t - \log\theta_{\text{out}}^t}\frac{d_i^t d_j^t}{2m_t}\right)\delta(g_i^t, g_j^t)\right] \\ &\qquad + \sum_{t=2}^{T}\sum_{i=1}^{N} \frac{\log\left(1 + \frac{p_t}{1-p_t}K_t\right)}{\langle \log\theta_{\text{in}}^t - \log\theta_{\text{out}}^t\rangle_t}\delta(g_i^{t-1}, g_i^t) + (\text{const.})\,, \end{aligned} \tag{5.22}$$

where $\langle\cdot\rangle_t$ denotes a mean across all layers $t \in \{1, \ldots, T\}$.

Guided by (5.22), we define temporal modularity for the case with non-uniform parameters with the following formula:

$$Q(\boldsymbol{g}) = \sum_{t=1}^{T} \beta_t \sum_{i,j=1}^{N} \left(A_{ij}^t - \gamma_t \frac{d_i^t d_j^t}{2m_t} \right) \delta(g_i^t, g_j^t) + \sum_{t=2}^{T} \omega_t \sum_{i=1}^{N} \delta(g_i^{t-1}, g_i^t)\,. \tag{5.23}$$

In contrast to previous definitions of multilayer modularity [98], our definition includes a new set of layer weights β_t, in addition to the resolution parameters γ_t and interlayer couplings ω_t. Comparing (5.23) to the log-probability in equation (5.22), we see that these parameters should take the following values:

$$\gamma_t = \frac{\theta_{\text{in}}^t - \theta_{\text{out}}^t}{\log \theta_{\text{in}}^t - \log \theta_{\text{out}}^t}\,, \tag{5.24}$$

$$\omega_t = \frac{1}{\langle \log \theta_{\text{in}}^t - \log \theta_{\text{out}}^t \rangle_t} \log\left(1 + \frac{p_t}{1-p_t} K_t\right)\,, \tag{5.25}$$

$$\beta_t = \frac{\log \theta_{\text{in}}^t - \log \theta_{\text{out}}^t}{\langle \log \theta_{\text{in}}^t - \log \theta_{\text{out}}^t \rangle_t}\,. \tag{5.26}$$

The expressions in (5.24)–(5.26) are consistent with the ones that we obtained for the case of uniform coupling. If the parameters θ_{in}^t, θ_{out}^t, p_t, and K_t are layer-independent, the first two expressions reduce to equations (5.18) and (5.19); additionally, equation (5.26) gives $\beta_t = 1$, which is why these weights do not appear explicitly when one considers uniform coupling.

The role of the parameters β_t is to give different weightings to intralayer modularity terms that correspond to different layers. To our knowledge, this is the first definition of multilayer modularity that incorporates such layer weightings explicitly.[5] Their values depend on the strength of the community structure in their respective layers. Letting $\epsilon_t = \theta_{\text{out}}^t / \theta_{\text{in}}^t$, we see that $\beta_t \propto \log(1/\epsilon_t)$. This implies that layers with strong community structure (i.e., ϵ_t close to 0) get a larger weight when maximising multilayer modularity than layers in which the community structure is weak (i.e., ϵ_t close to 1). It is also worth noting what happens when some layers have disassortative structure (i.e., when $\epsilon_t > 1$). The formula (5.26) suggests taking $\beta_t < 0$, pushing the method towards finding multilayer partitions that *minimise* monolayer modularity for these layers. Indeed, it is known that minimising modularity for monolayer networks identifies divisions with many edges between groups and few edges within groups [104, 105].

[5] One can write the most general version of multilayer modularity in [98] in the form of equation (5.23) by rescaling the adjacency matrices $\boldsymbol{A}^t$ to incorporate the weights β_t. However, this is not a scenario that the authors of [98] (or other researchers) have considered explicitly.

The multilayer setting is especially interesting, because it allows (at least in principle) a combination of assortative and disassortative layers [34].

Although the proposed multilayer modularity from equation (5.25) takes a non-standard form because of the weights β_t, methods such as GenLouvain [73], which are written as maximisation problems of the form given in equation (2.9), will continue to work. We also note that one can explicitly consider situations in which the interlayer coupling parameters are uniform and the resolution parameters are layer-dependent, or the other way around.

A key facet of the results of this section is that they give a principled method for choosing heterogeneous interlayer coupling weights ω_t. Although the original derivation of multilayer modularity in [98] is general enough to include scenarios with non-uniform parameters, most computations that researchers have done in practice have used uniform coupling and resolution values [15, 98, 158]. For most applications, this is a simplifying assumption rather than a hypothesis that these parameters are in fact the same for all layers. Our work suggests ways to remove this assumption and perform computations when parameters are layer-dependent, which we expect to be relevant for almost all applications.

Before discussing other types of multilayer structure in Sections 5.3.4 and 5.3.5, we note that the derivation from this section holds whenever the graph of dependencies between layers is a tree. In that case, community labels propagate from the root of this tree to the leaves, yielding a similar prior distribution $\mathbb{P}(\boldsymbol{g})$ to the one from equation (5.15). One example of potential interest for applications is the star graph, in which one central layer influences community structure in all remaining layers.

5.3.4 Multilevel Networks

In a multilevel network [90], the layers comprise a hierarchy of monolayer networks, and interlayer edges between nodes indicate inclusion relationships. Like in temporal networks, the layers in multilevel networks have a natural ordering, which makes it straightforward to modify the approach from Section 5.3.3 and obtain a similar result.

Multilevel networks have been studied before, most notably by Snijders and collaborators, who used multilevel modelling for social-network analysis [142, 143, 154]. The use of multilevel networks for community-detection applications is considerably rarer. We note a recent study by Barbillon et al. [12], who investigated a data set of collaborations and exchanges of resources between what they term "elite" cancer researchers in France and their laboratories. Although Barbillon et al. modelled these interactions as a multiplex network of researchers, one can alternatively (and perhaps

more appropriately) use a multilevel representation with researchers in one layer and laboratories in the other layer. A related example is a multilayer collaboration network, in which coauthored papers induce connections between individuals, research groups, departments, and universities (or a subset of these). International trade networks, in which nodes at higher levels represent larger geographical areas, provide another potential application.

For shopping networks, we show an example of a multilevel network in Figure 5.3. The bottom level consists of a bipartite customer–product network of the kind that we have studied previously. The top level connects customers to product *categories*, effectively aggregating purchases within the same category. The customers are the same across both levels, whereas each product in the bottom level belongs to exactly one product category in the top level, defining a many-to-one inclusion relationship. Thus, a multilevel network provides a way to incorporate existing product hierarchies into a multilayer network structure.

There are several reasons to study community structure in such multilevel networks. For shopping networks, the level of aggregation when constructing networks from transaction data is not a priori obvious; should loose bananas be considered the same product as packed bananas, or should they be represented as separate nodes in a network? A multilevel network of the sort depicted in Figure 5.3 includes multiple levels of aggregation at once. Another reason is that modularity maximisation, when run multiple times, typically identifies several different partitions with similar modularity values [56]. Choosing one from among these to analyse further is often difficult. By detecting communities in a multilevel network like the one in Figure 5.3, we push the algorithm towards those solutions that tend to assign products in the same category to the same community, and this can reduce the number of partitions that we need to consider.

A key difference between multilevel networks and the other examples that we consider in this chapter (including temporal and multiplex networks) is that we need to consider non-diagonal interlayer edges. Additionally, nodes in different layers no longer necessarily correspond to the same set of entities. We therefore need to introduce some new notation. Let $\mathcal{N}^t$ be the set of nodes in layer t, which has $|\mathcal{N}^t| = N^t$ nodes. Assume that $\mathcal{N}^1$ corresponds to the top level of the hierarchy. For $t \in \{2, \ldots, T\}$, the functions $\pi^t : \mathcal{N}^t \to \mathcal{N}^{t-1}$ send nodes $i \in \mathcal{N}^t$ to their parents immediately higher up in the hierarchy. (In the case of multilevel consumer–product networks, such a function maps customer nodes to themselves and product nodes to their corresponding product

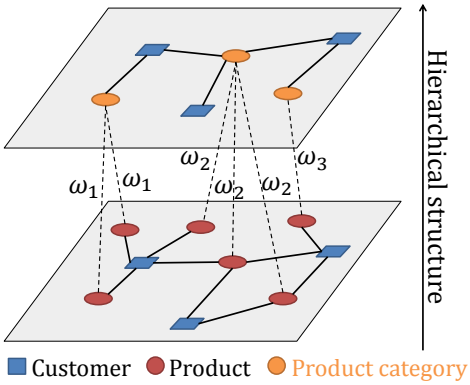


Figure 5.3: Example of a multilevel network of customers and products, with weighted interlayer edges indicating the inclusion of product nodes in their respective categories on the top level. Interlayer edges between customer nodes (not shown) connect copies of the same node in different layers.

categories.) For convenience, we use the notation π_i^t to denote $\pi^t(i)$; note the implicit assumption that node i is from layer t.

The log-likelihood for this multilevel network is similar to the one in equation (5.20) for non-uniform temporal networks:

$$\begin{aligned}&\log \mathbb{P}(\boldsymbol{\mathcal{A}}|\boldsymbol{g}, \boldsymbol{\theta}_{\text{in}}, \boldsymbol{\theta}_{\text{out}}) \qquad (5.27)\\ &= \sum_{t=1}^{T} \left[\left(\log \theta_{\text{in}}^t - \log \theta_{\text{out}}^t\right) \sum_{i,j \in \mathcal{N}^t} \left(A_{ij}^t - \frac{\theta_{\text{in}}^t - \theta_{\text{out}}^t}{\log \theta_{\text{in}}^t - \log \theta_{\text{out}}^t} \frac{d_i^t d_j^t}{2m_t} \right) \delta(g_i^t, g_j^t) \right] + (\text{const.})\,.\end{aligned}$$

The prior on $\boldsymbol{g}$ is also similar to that for temporal networks, although information flows from the top level of the hierarchy to the bottom level, rather than in the direction of time. First, we generate the vector $\boldsymbol{g}^1$ of community assignments in the top level by sampling from the null distribution $\mathbb{P}_0^1$. To sample community assignments $\boldsymbol{g}^t$ in subsequent layers, we then assume that each node copies its community label from its parent with probability p_t and otherwise randomly samples a label from a null distribution $\mathbb{P}_0^t$. This process yields the following log-prior:

$$\begin{aligned}\log \mathbb{P}(\boldsymbol{g}) &= \log \mathbb{P}(\boldsymbol{g}^1) + \sum_{t=2}^{T} \log \mathbb{P}(\boldsymbol{g}^t | \boldsymbol{g}^{t-1}) \\ &= \sum_{i \in \mathcal{N}^1} \log \mathbb{P}_0^1(g_i^1) + \sum_{t=2}^{T} \sum_{i \in \mathcal{N}^t} \log \left[(1-p_t) \mathbb{P}_0^t(g_i^t) \right] \\ &\quad + \sum_{t=2}^{T} \sum_{i \in \mathcal{N}^t} \log \left[1 + \frac{p_t}{(1-p_t)\mathbb{P}_0^t(g_i^t)} \right] \delta(g_{\pi_i^t}^{t-1}, g_i^t)\,. \qquad (5.28)\end{aligned}$$

As in Section 5.3.3, we assume that the null distributions $\mathbb{P}_0^t$ are uniform, so the first two terms in the right-hand side of (5.28) are independent of $\boldsymbol{g}$. In particular, we let $\mathbb{P}_0^t(g_i^t) = 1/K_t$. Ignoring constants, we then obtain the following expression for the log-prior:

$$\log \mathbb{P}(\boldsymbol{g}) = \sum_{t=2}^{T} \sum_{i \in \mathcal{N}^t} \log\left(1 + \frac{p_t}{1-p_t} K_t\right) \delta(g_{\pi_i^t}^{t-1}, g_i^t) + (\text{const.})\,. \tag{5.29}$$

From equations (5.27) and (5.29), it follows that the posterior distribution for $\boldsymbol{g}$ is

$$\begin{aligned}
&\log \mathbb{P}(\boldsymbol{g} | \boldsymbol{\mathcal{A}}, \boldsymbol{\theta}_{\text{in}}, \boldsymbol{\theta}_{\text{out}}, \boldsymbol{p}, \boldsymbol{K}) \\
&\quad = \sum_{t=1}^{T} \left[\left(\log \theta_{\text{in}}^t - \log \theta_{\text{out}}^t\right) \sum_{i,j \in \mathcal{N}^t} \left(A_{ij}^t - \frac{\theta_{\text{in}}^t - \theta_{\text{out}}^t}{\log \theta_{\text{in}}^t - \log \theta_{\text{out}}^t} \frac{d_i^t d_j^t}{2m_t} \right) \delta(g_i^t, g_j^t) \right] \\
&\qquad + \sum_{t=2}^{T} \sum_{i \in \mathcal{N}^t} \log\left(1 + \frac{p_t}{1-p_t} K_t\right) \delta(g_{\pi_i^t}^{t-1}, g_i^t) + (\text{const.})\,.
\end{aligned} \tag{5.30}$$

Meanwhile, the corresponding layer-weighted modularity for multilevel networks is

$$Q(\boldsymbol{g}) = \sum_{t=1}^{T} \beta_t \sum_{i,j \in \mathcal{N}^t} \left(A_{ij}^t - \gamma_t \frac{d_i^t d_j^t}{2m_t} \right) \delta(g_i^t, g_j^t) + \sum_{t=2}^{T} \sum_{i \in \mathcal{N}^t} \omega_t \delta(g_{\pi_i^t}^{t-1}, g_i^t)\,. \tag{5.31}$$

Comparing the expressions in (5.30) and (5.31), we arrive at the following values for the resolution parameters γ_t, coupling parameters ω_t, and layer weightings β_t:

$$\gamma_t = \frac{\theta_{\text{in}}^t - \theta_{\text{out}}^t}{\log \theta_{\text{in}}^t - \log \theta_{\text{out}}^t}\,, \tag{5.32}$$

$$\omega_t = \frac{1}{\langle \log \theta_{\text{in}}^t - \log \theta_{\text{out}}^t \rangle_t} \log\left(1 + \frac{p_t}{1-p_t} K_t\right)\,, \tag{5.33}$$

$$\beta_t = \frac{\log \theta_{\text{in}}^t - \log \theta_{\text{out}}^t}{\langle \log \theta_{\text{in}}^t - \log \theta_{\text{out}}^t \rangle_t}\,. \tag{5.34}$$

These are the same expressions as the ones for temporal networks in equations (5.24)–(5.26). Additionally, just as in the temporal case, one can also analyse the case of layer-independent parameters γ and ω. (In that case, the layer weights β_t are equal to 1.) Although this derivation is very similar to the one for temporal networks from Section 5.3.3, the application to multilevel networks is markedly different, due to the interpretation of interlayer edges as inclusion relationships. In Section 5.5, we apply this set of results to multilevel consumer–product networks.

5.3.5 Multiplex Networks

In a multiplex network, nodes are adjacent to each other via multiple types of edges; each edge type corresponds to a layer in the resulting multilayer network [78, 157]. In this section, we establish a relationship between modularity maximisation and certain multilayer SBMs for multiplex networks.

As in Section 5.3.3, we continue to assume that our multilayer networks are fully-interconnected, diagonal, and layer-coupled. To construct a multiplex SBM, we follow [17] and let $\boldsymbol{P}$ denote the matrix of interlayer dependencies, where p_{st} (with $s \neq t$) is the probability that a node in layer t copies its community label from layer s. (For $s = t$, we assume that $p_{st} = 0$.) The matrix $\boldsymbol{P}$ may or may not be symmetric. We say that the case in which all probabilities p_{st} are equal to the same value p has "uniform coupling".

Recall from Section 5.3.2 that the intralayer modularity terms $Q_{\text{intra}}(\boldsymbol{g})$ take the same functional form as the conditional log-likelihood $\mathbb{P}(\boldsymbol{\mathcal{A}}|\boldsymbol{g}, \boldsymbol{\theta}_{\text{in}}, \boldsymbol{\theta}_{\text{out}})$, regardless of the dependency structure between layers. We therefore focus our attention on devising a prior $\mathbb{P}(\boldsymbol{g})$ that recovers the interlayer modularity terms $Q_{\text{inter}}(\boldsymbol{g})$ for multiplex networks.

In a multiplex network, there is no natural ordering of the layers, in contrast to temporal and multilevel networks. Consider the vector $\boldsymbol{g}_i = (g_i^1, \ldots, g_i^T)$ of community labels for some node i. From the definition of conditional probability, it follows that

$$\mathbb{P}(\boldsymbol{g}_i) = \mathbb{P}(g_i^1)\mathbb{P}(g_i^2|g_i^1) \times \cdots \times \mathbb{P}(g_i^T|g_i^1, \ldots, g_i^{T-1})\,.$$

However, we can also write

$$\mathbb{P}(\boldsymbol{g}_i) = \mathbb{P}(g_i^T)\mathbb{P}(g_i^{T-1}|g_i^T) \times \cdots \times \mathbb{P}(g_i^1|g_i^T, \ldots, g_i^2)\,,$$

and we can write a similar expression for any other permutation of the T layers. Of course, all of these products have to evaluate to the same quantity (namely, $\mathbb{P}(\boldsymbol{g}_i)$). It is difficult to define a probabilistic model for $\boldsymbol{g}$ that is internally consistent in this way. In particular, a model similar to the one in [18] does not have these properties (in addition to not having a closed form).

We take a different approach. Considering each node i independently, we first sample a permutation σ of the layers from the symmetric group S_T with probability $\mathbb{P}(\sigma)$, and we then update community labels in the order indicated by this permutation: first, we sample a community label for i in what is now layer 1; we then sample a label

in layer 2, conditioned on the label from layer 1; and so on.[6] The permutation σ that indicates the update order can be different for different nodes. The resulting prior is

$$\mathbb{P}(\boldsymbol{g}) = \prod_{i=1}^{N} \sum_{\sigma \in S_T} \mathbb{P}(\sigma) \mathbb{P}\left(g_i^{\sigma^{-1}(1)}\right) \prod_{t=2}^{T} \mathbb{P}\left(g_i^{\sigma^{-1}(t)} | g_i^{\sigma^{-1}(t-1)}\right). \tag{5.35}$$

Note that we assume that the update process is memoryless, so the community label in layer t (after applying the permutation σ) depends only on the community label in layer $t-1$. This is the same assumption that we (and others [17, 51]) have made for temporal networks.

For simplicity, we write ς_t for $\sigma^{-1}(t)$. As in Section 5.3.3, we use a copying and resampling process to generate community labels. The conditional probabilities in equation (5.35) are then given by

$$\mathbb{P}(g_i^{\varsigma_t} | g_i^{\varsigma_{t-1}}) = p_{\varsigma_{t-1}\varsigma_t} \delta(g_i^{\varsigma_{t-1}}, g_i^{\varsigma_t}) + (1 - p_{\varsigma_{t-1}\varsigma_t}) \mathbb{P}_0^{\varsigma_t}(g_i^{\varsigma_t}) .$$

For the first layer,

$$\mathbb{P}(g_i^{\varsigma_1}) = \mathbb{P}_0^{\varsigma_1}(g_i^{\varsigma_1}) .$$

With a uniform prior $\mathbb{P}_0^{\varsigma_t}(g_i^{\varsigma_t}) = 1/K_{\varsigma_t}$, we obtain

$$\mathbb{P}(\boldsymbol{g}) = \prod_{i=1}^{N} \sum_{\sigma \in S_T} \left\{ \mathbb{P}(\sigma) \frac{\prod_{t=2}^{T} (1 - p_{\varsigma_{t-1}\varsigma_t})}{\prod_{t=1}^{T} K_{\varsigma_t}} \prod_{t=2}^{T} \left[1 + \frac{p_{\varsigma_{t-1}\varsigma_t}}{1 - p_{\varsigma_{t-1}\varsigma_t}} K_{\varsigma_t} \delta(g_i^{\varsigma_{t-1}}, g_i^{\varsigma_t}) \right] \right\} .$$

Taking logarithms and treating any terms that do not depend on $\boldsymbol{g}$ as constant yields

$$\log \mathbb{P}(\boldsymbol{g}) = \sum_{i=1}^{N} \log \left\{ \sum_{\sigma \in S_T} \mathbb{P}(\sigma) \prod_{t=2}^{T} (1 - p_{\varsigma_{t-1}\varsigma_t}) \left[1 + \frac{p_{\varsigma_{t-1}\varsigma_t}}{1 - p_{\varsigma_{t-1}\varsigma_t}} K_{\varsigma_t} \delta(g_i^{\varsigma_{t-1}}, g_i^{\varsigma_t}) \right] \right\} + (\text{const.}) . \tag{5.36}$$

As we mentioned in Section 5.3.3, being able to take the Kronecker delta outside of the logarithmic expression is the key step that allows us to relate this prior probability to the corresponding interlayer modularity terms. However, we cannot do this easily in the present case because of the sum over permutations $\sigma \in S_T$. Instead, we bound the expression from equation (5.36) using Jensen's inequality. Because the logarithm is a concave function,

$$\log \left(\sum_i x_i \right) \geq \sum_i q_i \log \left(\frac{x_i}{q_i} \right) \tag{5.37}$$

[6] One can alternatively consider sampling over all possible spanning trees on T nodes and assume that the update propagates from the root of each tree to the leaves.

for any $q_i \geq 0$ with $\sum_i q_i = 1$, and equality holds for $q_i = x_i / \sum_i x_i$. Applying the inequality (5.37) to equation (5.36) yields

$$\log \mathbb{P}(\boldsymbol{g}) \geq \sum_{i=1}^{N} \sum_{\sigma \in S_T} q_i^\sigma \log \left\{ \mathbb{P}(\sigma) \frac{\prod_{t=2}^{T} (1 - p_{\varsigma_{t-1}\varsigma_t})}{q_i^\sigma} \prod_{t=2}^{T} \left[1 + \frac{p_{\varsigma_{t-1}\varsigma_t}}{1 - p_{\varsigma_{t-1}\varsigma_t}} K_{\varsigma_t} \delta(g_i^{\varsigma_{t-1}}, g_i^{\varsigma_t}) \right] \right\} + (\text{const.}) \tag{5.38}$$

$$= \sum_{i=1}^{N} \sum_{\sigma \in S_T} q_i^\sigma \left[\log \mathbb{P}(\sigma) - \log q_i^\sigma + \sum_{t=2}^{T} \log(1 - p_{\varsigma_{t-1}\varsigma_t}) \right] + \sum_{i=1}^{N} \sum_{\sigma \in S_T} \left\{ q_i^\sigma \sum_{t=2}^{T} \left[\log \left(1 + \frac{p_{\varsigma_{t-1}\varsigma_t}}{1 - p_{\varsigma_{t-1}\varsigma_t}} K_{\varsigma_t} \right) \delta(g_i^{\varsigma_{t-1}}, g_i^{\varsigma_t}) \right] \right\} + (\text{const.}) \,. \tag{5.39}$$

The first term on the right-hand side of the inequality (5.39) does not depend on $\boldsymbol{g}$, so we can ignore it. Therefore,

$$\log \mathbb{P}(\boldsymbol{g}) \geq \sum_{i=1}^{N} \sum_{\sigma \in S_T} q_i^\sigma \sum_{t=2}^{T} \log \left(1 + \frac{p_{\varsigma_{t-1}\varsigma_t}}{1 - p_{\varsigma_{t-1}\varsigma_t}} K_{\varsigma_t} \right) \delta(g_i^{\varsigma_{t-1}}, g_i^{\varsigma_t}) + (\text{const.}) \,. \tag{5.40}$$

We want to rearrange the right-hand side of (5.40) into an expression that resembles the interlayer modularity terms in equation (5.1). To do this, we fix s and t and consider all of the terms in (5.40) that involve $\delta(g_i^s, g_i^t)$. There are $(T-1)!$ permutations σ with $\varsigma_{t'-1} = s$ and $\varsigma_{t'} = t$ for some $t' \in \{2, \ldots, T\}$. For convenience, let $S_T^{(s,t)}$ denote the subset of S_T that consists of permutations that map layers s and t to two successive layers. The lower bound of the log-prior then becomes

$$\log \mathbb{P}(\boldsymbol{g}) \geq \sum_{i=1}^{N} \sum_{s \neq t} \sum_{\sigma \in S_T^{(s,t)}} q_i^\sigma \log \left(1 + \frac{p_{st}}{1 - p_{st}} K_t \right) \delta(g_i^s, g_i^t) + (\text{const.}) \,. \tag{5.41}$$

The fact that we have a lower bound for this log-prior rather than an exact equality like in Section 5.3.3 may seem problematic. However, recall that we obtain an exact equality if the quantities q_i^σ take the specific values

$$q_i^\sigma = \frac{\mathbb{P}(\sigma) \prod_{t=2}^{T} (1 - p_{\varsigma_{t-1}\varsigma_t}) \left[1 + \frac{p_{\varsigma_{t-1}\varsigma_t}}{1 - p_{\varsigma_{t-1}\varsigma_t}} K_{\varsigma_t} \delta(g_i^{\varsigma_{t-1}}, g_i^{\varsigma_t}) \right]}{Z_i} \,. \tag{5.42}$$

For each node $i \in \{1, \ldots, N\}$, the quantity Z_i is a proportionality constant that ensures that $\sum_{\sigma \in S_T} q_i^\sigma = 1$. One can interpret each q_i^σ as the posterior probability $\mathbb{P}(\sigma | \boldsymbol{g}_i)$ that community labels for node i were sampled in the layer order indicated by

σ. This posterior probability is proportional to the product of the prior probability $\mathbb{P}(\sigma)$ and the likelihood $\mathbb{P}(\boldsymbol{g}_i|\sigma)$.

By comparing the right-hand side of (5.41) to the intralayer modularity terms from (5.1), we arrive at the following optimal values for the coupling parameters ω_{st}:

$$\omega_{st} = \frac{1}{\langle \log \theta^t_{\text{in}} - \log \theta^t_{\text{out}} \rangle_t} \log\left(1 + \frac{p_{st}}{1 - p_{st}} K_t\right) \sum_{\sigma \in S_T^{(s,t)}} q_i^\sigma \,. \tag{5.43}$$

Given a multilayer partition $\boldsymbol{g}$ and values for the other parameters of the generative model, one can — in principle — calculate values for the weights q_i^σ using equation (5.42). Substituting these values into equation (5.43) then gives estimates for the coupling weights ω_{st}. In practice, the sum over permutations $\sigma \in S_T^{(s,t)}$ from equation (5.43) has $(T-1)!$ terms. Although there are many empirical data sets with a sufficiently small number of layers (say, $T \lessapprox 6$) for which this calculation is feasible, this brute-force approach will not work in general.

We can make some simplifications in the case of uniform coupling, for which $p_{st} = p$ for $s, t \in \{1, \ldots, T\}$. Furthermore, we assume that the number of communities is the same for each layer ($K_t = K$) and that each permutation σ is sampled with equal probability (i.e., $\mathbb{P}(\sigma) = 1/T!$). In this case, the layers of our multiplex network are statistically equivalent, so on average any permutation of the layers should produce the same results. This implies that the quantities $q_i^\sigma = \mathbb{P}(\sigma|\boldsymbol{g}_i)$ ($\sigma \in S_T$) are equal in expectation, which yields

$$\sum_{\sigma \in S_T^{(s,t)}} q_i^\sigma \approx \frac{1}{T} \sum_{\sigma \in S_T} q_i^\sigma = \frac{1}{T} \,. \tag{5.44}$$

It is possible to make this argument mathematically rigorous in the limit $T \to \infty$ by approximating the quantities q_i^σ (for $\sigma \in S_T^{(s,t)}$) with quantities $q_i^{\sigma'}$ (for $\sigma' \in S_T$) of similar magnitude. In Table 5.1, we provide some numerical evidence that our approximation is reasonable even for small T. Indeed, we observe that the sample mean of the expression $\left(\sum_{\sigma \in S_T^{(s,t)}} q_i^\sigma\right) - 1/T$ is close to 0 for different values of the copying probability p and the number of layers T. (For these examples, we use $K = 5$.) From a practical standpoint, we expect that any errors in the approximation (5.44) will have a small impact on the results of modularity maximisation compared to other sources of error (e.g., the algorithm identifying different local optima of the modularity function across different runs).

The approximation (5.44) gives the following optimal coupling values for uniform multiplex networks:

$$\omega = \frac{1}{T \langle \log \theta_{\text{in}} - \log \theta_{\text{out}} \rangle_t} \log\left(1 + \frac{p}{1 - p} K\right) \,. \tag{5.45}$$

Table 5.1: Sample mean and standard deviation (in parentheses) of the expression $\left(\sum_{\sigma \in S_T^{(s,t)}} q_i^\sigma\right) - 1/T$ across 50 trials.

	$T = 3$	$T = 4$	$T = 5$	$T = 6$	$T = 7$	$T = 8$	$T = 9$	$T = 10$
$p = 0.5$	−0.013	0.027	0.009	0.008	−0.013	0.005	−0.004	−0.020
	(0.052)	(0.098)	(0.090)	(0.090)	(0.076)	(0.078)	(0.069)	(0.047)
$p = 0.7$	0.000	−0.001	0.016	0.003	−0.008	−0.003	0.025	0.010
	(0.068)	(0.073)	(0.082)	(0.096)	(0.071)	(0.086)	(0.100)	(0.082)
$p = 0.9$	−0.003	0.006	−0.004	0.007	0.004	0.010	0.008	−0.007
	(0.016)	(0.048)	(0.064)	(0.075)	(0.051)	(0.080)	(0.059)	(0.052)

In particular, we see that $\omega_{\text{multiplex}} = \omega_{\text{temporal}}/T$. This scaling makes intuitive sense, as the number of directed interlayer edges per physical node is equal to $T - 1$ for temporal networks and is equal to $T(T - 1)$ for multiplex networks.

5.3.6 Potential Extensions to Weighted Networks

The work that we have presented thus far in this section applies to unweighted networks; deriving analogous results for weighted networks (e.g., by relating existing weighted SBMs [3, 119] to weighted modularity) is considerably more difficult. We now discuss two ideas to address this problem, although we leave in-depth analysis to future work.

A key observation when generalising our approach to weighted networks is that weights only impact intralayer terms. In particular, the relationships between interlayer modularity terms $Q_{\text{inter}}(\boldsymbol{g})$ and log-priors $\log \mathbb{P}(\boldsymbol{g}|p, K)$ that we derived for temporal, multilevel, and multiplex networks continue to hold. (This is not surprising, as these quantities do not depend on the adjacency tensor $\boldsymbol{\mathcal{A}}$.) Thus, to study extensions of our analysis to weighted networks, it is sufficient to examine intralayer terms.

Let $\boldsymbol{W}$ be the weighted adjacency matrix of a monolayer network. Our first idea consists of "quantising" the weights in $\boldsymbol{W}$ and representing the resulting structure as an unweighted network with multiedges. We can model this situation using a Poisson SBM. More precisely, let $\widetilde{\boldsymbol{W}} \approx C\boldsymbol{W}$ be a rescaled and discretised approximation of $\boldsymbol{W}$ for some scalar constant C. (A discussion on how to choose C follows later.) Let $\widetilde{d}_i = \sum_{j=1}^N \widetilde{\boldsymbol{W}}_{ij}$ be the degree of node i in the new network $\widetilde{\boldsymbol{W}}$, and let $\widetilde{m} = \sum_{i,j=1}^N \widetilde{\boldsymbol{W}}_{ij}/2$ be the number of edges in this network. Note that $\widetilde{d}_i \approx Cs_i$ and $\widetilde{m} \approx Cm$, where $s_i = \sum_{j=1}^N W_{ij}$ is the strength of node i in the original weighted network $\boldsymbol{W}$ and $m = \sum_{i,j=1}^N W_{ij}/2$ is the total edge weight. Using a Poisson PPM (see Section 5.3.1),

the log-likelihood for $\widetilde{\boldsymbol{W}}$ is

$$\begin{aligned}\log \mathbb{P}(\widetilde{\boldsymbol{W}}|\boldsymbol{g}, \theta_{\text{in}}, \theta_{\text{out}}) &= (\log \theta_{\text{in}} - \log \theta_{\text{out}}) \sum_{i,j=1}^{N} \left(\widetilde{W}_{ij} - \gamma \frac{\widetilde{d}_i \widetilde{d}_j}{2\widetilde{m}} \right) \delta(g_i, g_j) + (\text{const.}) \\ &\approx C(\log \theta_{\text{in}} - \log \theta_{\text{out}}) \sum_{i,j=1}^{N} \left(W_{ij} - \gamma \frac{s_i s_j}{2m} \right) \delta(g_i, g_j) + (\text{const.}) \,,\end{aligned}$$

where

$$\gamma = \frac{\theta_{\text{in}} - \theta_{\text{out}}}{\log \theta_{\text{in}} - \log \theta_{\text{out}}} \,. \tag{5.46}$$

For a weighted multilayer network, this derivation implies that one needs to absorb the factor $C(\log \theta_{\text{in}} - \log \theta_{\text{out}})$, rather than just $(\log \theta_{\text{in}} - \log \theta_{\text{out}})$, into the expression for ω. For example, given a temporal network and assuming all parameters are layer-independent, we obtain

$$\omega \approx \frac{1}{C(\log \theta_{\text{in}} - \log \theta_{\text{out}})} \log \left(1 + \frac{p}{1-p} K \right) \,. \tag{5.47}$$

Thus, compared to the unweighted setting, there is an additional factor of C in the denominator. When $C = 1$ (and particularly when $\boldsymbol{\mathcal{W}}$ is an unweighted adjacency tensor, which requires no additional approximations), the formula (5.47) matches the expression (5.19) for unweighted temporal networks. One can now formulate an iterative algorithm that maximises multilayer modularity for weighted networks and that updates γ and ω according to the formulas (5.46) and (5.47).

How should one select C? In qualitative terms, we want the resulting matrix $\widetilde{\boldsymbol{W}}$ to be well-approximated by a Poisson PPM. Thus, one possibility is to select C by minimising the error (under a suitable norm) between entries of $\widetilde{\boldsymbol{W}}$ and expected values of the entries under the Poisson approximation. A key challenge is to perform this optimisation in an efficient manner, without considering all N^2 entries of the matrix $\widetilde{\boldsymbol{W}}$.

We now consider a second idea for handling weighted multilayer networks. Recall that the prior $\mathbb{P}(\boldsymbol{g}|p, K)$ does not involve any of the intralayer adjacency matrices $\boldsymbol{A}^t$ ($t \in \{1, \ldots, T\}$), so our derivations from Sections 5.3.3–5.3.5 that involve this prior do not change for weighted networks. Thus, in both unweighted and weighted settings, we expect that

$$\omega = C' \log \left(1 + \frac{p}{1-p} K \right) \,, \tag{5.48}$$

where the proportionality constant C' does not depend on p or K; rather, C' arises from matching single-layer modularity to the log-likelihood of a suitable single-layer

PPM. For instance, $C' = 1/(\log \theta_{\text{in}} - \log \theta_{\text{out}})$ for unweighted networks with temporal multilayer structure (see Section 5.3.3). The key question is how to estimate C' in the absence of an explicit equivalence between weighted single-layer modularity and a weighted single-layer SBM. One idea is to perform regression. More precisely, given a collection of different multilayer partitions, which one can obtain by running GenLouvain for different ω values, it is possible to estimate p and K for each of these partitions, as we will explain in Section 5.4.2. It is important that one uses the same γ value to obtain all of these partitions, so that C' remains approximately constant. By plotting ω against the logarithmic expression from (5.48), one can then determine C' using linear regression. (The challenge, as always, is that there is significant noise due to modularity maximisation returning different local optima across different runs.)

Although we do not explore these two ideas any further, we believe that they are reasonable starting points for investigating generalisations of our approach to weighted multilayer networks.

5.4 Parameter Estimation

In Section 5.3, we showed an equivalence between modularity maximisation and statistical inference in SBMs, given certain assumptions, for different types of multilayer networks. In this section, we use these results to determine appropriate values for the resolution and interlayer coupling parameters that arise in the formulation of multilayer modularity. For concreteness, we focus on the uniform setting, in which there are only two parameters (γ and ω). However, one can straightforwardly extend all of our results in this section to the layer-dependent case.

5.4.1 Iterative Algorithm

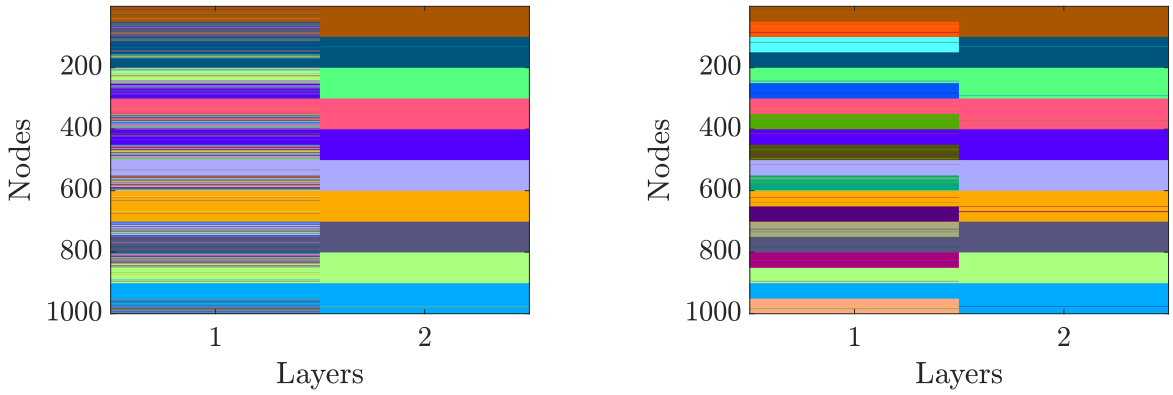
Throughout Section 5.3, we derived expressions for γ and ω that depend on the unknown quantities θ_{in}, θ_{out}, p, and K. We therefore have to estimate the values of all of these parameters; we use an iterative procedure similar to the one that was proposed in [105] for monolayer networks. Given some initial guesses $\gamma^{(0)}$ and $\omega^{(0)}$, the iterative process alternates between running modularity maximisation with the current estimates of these parameters and using the resulting community structure to estimate θ_{in}, θ_{out}, p, and K, thereby obtaining new values for γ and ω. The algorithm stops once γ and ω converge, up to some pre-specified numerical tolerance. We give the pseudocode for this iterative procedure in Algorithm 5.1. The iteration proceeds similarly when the parameters are layer-dependent. We note that it is also possible to

use the framework of Algorithm 5.1 to only update one of the two parameters, γ and ω, while keeping the other one fixed. In our applications to shopping networks from Section 5.5, we find it useful to use a version of our IterModMax algorithm that fixes γ and only iterates over ω.

function IterModMax($\boldsymbol{\mathcal{A}}$)
  initialize $\gamma = \gamma^{(0)}$ and $\omega = \omega^{(0)}$
  while not converged **do**
    $\boldsymbol{g} \leftarrow$ MaximiseModularity($\boldsymbol{\mathcal{A}}, \gamma, \omega$) ▷ Community detection (e.g., using GenLouvain [73])
    $\theta_{\text{in}}, \theta_{\text{out}}, p, K \leftarrow$ EstimateSBMPara($\boldsymbol{\mathcal{A}}, \boldsymbol{g}$) ▷ Use detected communities to estimate SBM parameters
    $\gamma \leftarrow$ UpdateGamma($\theta_{\text{in}}, \theta_{\text{out}}$) ▷ Update γ using equation (5.13)
    $\omega \leftarrow$ UpdateOmega($\theta_{\text{in}}, \theta_{\text{out}}, p, K$) ▷ Update ω using equation (5.19) or equation (5.45)
  end while
  return γ, ω ▷ Optimal modularity parameters
  return $\boldsymbol{g}$ ▷ Detected communities using optimal parameters
end function

Algorithm 5.1: Iterative algorithm for performing modularity maximisation and estimating resolution and interlayer-coupling parameters in a multilayer network.

A key step of Algorithm 5.1 is estimating θ_{in}, θ_{out}, p, and K, given the multilayer network's supra-adjacency matrix $\boldsymbol{\mathcal{A}}$ and the current multilayer partition $\boldsymbol{g}$. We address this issue in Section 5.4.2, but first we illustrate the performance of Algorithm 5.1 on a simple network. We construct a network with $N = 1000$ nodes, $T = 2$ layers, and $K_1 = 20$ communities in the first layer that merge pairwise into $K_2 = 10$ communities in the second layer. We place edges independently in the two layers with probabilities $p_{\text{in}} = 0.32$ (for nodes in the same community) and $p_{\text{out}} = 0.1$ (for nodes in different communities); we choose these values near the detectability threshold for a monolayer network to make the community-detection task sufficiently difficult. Additionally, note that the parameters p_{in} and p_{out} are not the same as the parameters θ_{in} and θ_{out} from our generative model, as the latter are also multiplied by node degrees to generate edge probabilities. In Figure 5.4, we show the detected communities for two parameter choices: (a) the parameter values $\gamma = 1$ (i.e., the "default" value [124]) and $\omega = 1$; and (b) the values $\gamma \approx 1.60$ and $\omega \approx 1.30$ to which the iteration from Algorithm 5.1 converges. The naive approach recovers the correct structure in the second layer but not the first; in particular, it identifies 10, rather than 20, communities in the first layer. In contrast, our iterative modularity-maximisation algorithm identifies a multilayer partition that is close to the planted one.



(a) $\gamma = 1$ and $\omega = 1$ (b) $\gamma \approx 1.60$ and $\omega \approx 1.30$

Figure 5.4: Detected multilayer partitions for a toy network with 20 communities in the first layer that merge into 10 communities in the second layer. We set the values of γ and ω to (a) $\gamma = 1$ (the "default" value) and $\omega = 1$ and (b) the optimal values $\gamma \approx 1.60$ and $\omega \approx 1.30$ that we infer using Algorithm 5.1. In both cases, the normalised mutual information (NMI) between the detected partitions and the planted one is close to 1 for the second layer. For the first layer, the NMI is approximately 0.5 using the naive approach, and it is about 0.95 for the optimised approach.

5.4.2 Estimating SBM Parameters

In this section, we explain how to estimate, given a multilayer partition $\boldsymbol{g}$, the parameters of our various SBMs from Section 5.3. We approximate θ_{in} and θ_{out} in a manner similar to that presented in [105] for monolayer networks. Let M_{in}^t denote the random variable for the number of intra-community edges in layer t under the degree-corrected SBM from Section 5.3.2. The expected number of such edges across all layers is

$$\mathbb{E}\left[\sum_{t=1}^{T} M_{\text{in}}^t\right] = \frac{1}{2}\sum_{t=1}^{T}\sum_{i,j=1}^{N} \theta_{\text{in}} \frac{d_i^t d_j^t}{2m_t}\delta(g_i^t, g_j^t) = \frac{1}{2}\theta_{\text{in}}\sum_{t=1}^{T}\frac{\sum_r (\kappa_r^t)^2}{2m_t}\,, \tag{5.49}$$

where κ_r^t denotes the sum of the degrees of nodes in community r in layer t. Replacing the expectation on the left-hand side of equation (5.49) with the observed number of intra-community edges ($\sum_{t=1}^{T} m_{\text{in}}^t$) yields the estimate

$$\theta_{\text{in}} \approx \frac{\sum_{t=1}^{T} 2m_{\text{in}}^t}{\sum_{t=1}^{T} \frac{1}{2m_t}\sum_r (\kappa_r^t)^2}\,. \tag{5.50}$$

Using a similar approach, we estimate that

$$\theta_{\text{out}} \approx \frac{\sum_{t=1}^{T} 2m_{\text{out}}^t}{\sum_{t=1}^{T}\left[2m_t - \frac{1}{2m_t}\sum_r (\kappa_r^t)^2\right]}\,. \tag{5.51}$$

In the layer-dependent case, one can determine each set of parameters, θ^t_{in} and θ^t_{out}, in a similar way by considering intra-community and inter-community edges separately for each layer.

We now consider the other two sets of parameters. We set the number K of communities equal to the observed number of distinct community labels in the current multilayer partition $\boldsymbol{g}$:

$$K = \left|\left\{g_i^t : i \in \{1, \dots, N\}, t \in \{1, \dots, T\}\right\}\right| . \tag{5.52}$$

In the layer-dependent case, we set

$$K_t = \left|\left\{g_i^t : i \in \{1, \dots, N\}\right\}\right| , \quad t \in \{1, \dots, T\} . \tag{5.53}$$

Estimating the copying probabilities p (or p_t, if they are layer-dependent) is more difficult. The details of the calculation also depend on the type of multilayer network that one considers. We discuss temporal, multilevel, and multiplex networks in turn.

Temporal and multilevel networks

To estimate the copying probability p for a temporal network, note that the probability that a node i is in the same community in two consecutive layers in the proposed model is

$$\mathbb{P}(g_i^{t-1} = g_i^t) = p + \frac{1-p}{K} . \tag{5.54}$$

The first term corresponds to the probability that node i copies its community assignment from the previous layer, and the second term is the probability that node i samples that community uniformly from the K available choices. Estimating the same probability empirically from network data amounts to counting the fraction of times that a node stays in the same community (i.e., "persists") across two consecutive layers:

$$\mathbb{P}(g_i^{t-1} = g_i^t) \approx \frac{1}{N(T-1)} \sum_{t'=2}^{T} \sum_{i'=1}^{N} \delta(g_{i'}^{t'-1}, g_{i'}^{t'}) = \frac{\text{Pers}(\boldsymbol{g})}{N(T-1)} , \tag{5.55}$$

where we follow the notation in [18] and let $\text{Pers}(\boldsymbol{g})$ denote the number of instances in which a node belongs to the same community in consecutive layers. Equating the empirical probability (5.55) to the theoretical one in (5.54) yields

$$p + \frac{1-p}{K} \approx \frac{\text{Pers}(\boldsymbol{g})}{N(T-1)} \quad \Rightarrow \quad p \approx \frac{\frac{\text{Pers}(\boldsymbol{g})}{N(T-1)} - \frac{1}{K}}{1 - \frac{1}{K}} . \tag{5.56}$$

In the layer-dependent case, a similar derivation yields

$$p_t \approx \frac{\frac{\sum_{i=1}^{N} \delta(g_i^{t-1}, g_i^t)}{N} - \frac{1}{K}}{1 - \frac{1}{K}}\,, \quad t \in \{2, \ldots, T\}\,. \tag{5.57}$$

We follow an almost identical approach for multilevel networks. In the uniform case, our estimate for p is the same as in equation (5.56), although the calculation of persistence $\mathrm{Pers}(\boldsymbol{g})$ is different:

$$\mathrm{Pers}(\boldsymbol{g}) = \sum_{t=2}^{T} \sum_{i \in \mathcal{N}^t} \delta(g_{\pi_i^t}^{t-1}, g_i^t) \quad \Rightarrow \quad p \approx \frac{\frac{1}{T-1} \sum_{t=2}^{T} \frac{1}{N^t} \sum_{i \in \mathcal{N}^t} \delta(g_{\pi_i^t}^{t-1}, g_i^t) - \frac{1}{K}}{1 - \frac{1}{K}}\,. \tag{5.58}$$

Similarly, for the layer-dependent case,

$$p_t \approx \frac{\frac{1}{N^t} \sum_{i \in \mathcal{N}^t} \delta(g_{\pi_i^t}^{t-1}, g_i^t) - \frac{1}{K}}{1 - \frac{1}{K}}\,, \quad t \in \{2, \ldots, T\}\,. \tag{5.59}$$

Multiplex networks

Estimating the probability p for the multiplex version of our SBM — or the probabilities p_{st}, if one is considering layer-dependent parameters — is significantly more difficult. For simplicity, we start with the uniform case.

Consider the probability $\mathbb{P}(g_i^s = g_i^t)$ for a randomly chosen node i and two arbitrary layers s and t (with $s \neq t$). For a given multilayer partition $\boldsymbol{g}$, we estimate $\mathbb{P}(g_i^s = g_i^t)$ by counting the number of instances in which two community labels are the same and dividing by the total number of instances:

$$\mathbb{P}(g_i^s = g_i^t) \approx \frac{1}{NT(T-1)} \sum_{i'=1}^{N} \sum_{t'=1}^{T} \sum_{s' \neq t'} \delta(g_{i'}^{s'}, g_{i'}^{t'})\,. \tag{5.60}$$

We now calculate $\mathbb{P}(g_i^s = g_i^t)$ for the generative model described in Section 5.3.5. To do that, we have to account for all of the possible ways in which a node i can have the same community label in layers s and t. Suppose that labels are updated in the order that is indicated by a permutation σ and that, in this permutation, there are $n-1$ intermediate layers between s and $t \equiv t_n$. We denote these layers by $t_1, \ldots, t_{n-1}$. By symmetry, we may assume that layer s comes before layer t in the permutation, and we introduce a factor of 2 when necessary to account for the reverse case. We can break down the probability $\mathbb{P}(g_i^s = g_i^t)$ into two components: either (1) node i in layer t copies the label from layer t_{n-1} and that label is identical to the one from layer s,

or (2) node i in layer t randomly samples the same label from layer s. Therefore, we obtain the recursive relationship

$$\mathbb{P}(g_i^s = g_i^{t_n}) = p\mathbb{P}(g_i^s = g_i^{t_{n-1}}) + \frac{1-p}{K}\,, \tag{5.61}$$

which holds for any n. We illustrate this process in Figure 5.5. The base case is

$$\mathbb{P}(g_i^s = g_i^{t_1}) = p + \frac{1-p}{K}\,. \tag{5.62}$$

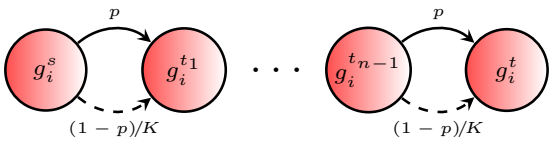


Figure 5.5: Illustration of the copying and sampling process that propagates community labels across layers. Assume that after applying a permutation, the layers are ordered $s, t_1, \ldots, t_{n-1}, t$. We are interested in the probability that a node i has the same community labels in layers s and t (i.e., that $g_i^s = g_i^t$). For each layer from $t_1, \ldots, t_{n-1}, t$, node i copies its community label from the previous layer with probability p (solid arrows), or it samples the same label uniformly at random with probability $(1-p)/K$ (dashed arrows).

Let $f(x) = px + (1-p)/K$. Thus, $f(1)$ is the probability that a node in one layer either copies the community label from the previous layer (after applying the permutation) or that it randomly samples the same label. We can then summarize the recursion from equations (5.61) and (5.62) with

$$\mathbb{P}(g_i^s = g_i^{t_n}) = f^{(n)}(1)\,, \tag{5.63}$$

where $f^{(n)}$ denotes that f is composed with itself n times.

The expression (5.63) corresponds to the probability that nodes s and t share the same label when they are n layers apart after applying a permutation σ. To get the overall probability, we first have to count, for each n, the number of permutations with this property. Specifically, we need to consider all $\sigma \in S_T$ with $\sigma^{-1}(u) = s$, $\sigma^{-1}(u+1) = t_1$, $\ldots$, $\sigma^{-1}(u+n) = t_n \equiv t$ for some u. Using straightforward combinatorics, one can show that there are $(T-2)!(T-n)$ such permutations. This yields

$$\mathbb{P}(g_i^s = g_i^t) = \sum_{n=1}^{T-1} \frac{2(T-2)!(T-n)}{T!} f^{(n)}(1) = \sum_{n=1}^{T-1} \frac{2(T-n)}{T(T-1)} f^{(n)}(1)\,. \tag{5.64}$$

For $p = 1$, note that $f(1) = 1$, and equation (5.64) gives $\mathbb{P}(g_i^s = g_i^t) = 1$, as expected. For $p < 1$, the function f has a unique fixed point at $1/K$. We can write

$$f^{(n)}(1) - \frac{1}{K} = p\left(f^{(n-1)}(1) - \frac{1}{K}\right) = \ldots = p^n\left(1 - \frac{1}{K}\right),$$

which implies that

$$f^{(n)}(1) = p^n \frac{K-1}{K} + \frac{1}{K}.$$

This yields

$$\mathbb{P}(g_i^s = g_i^t) = \frac{2(1-\frac{1}{K})}{T(T-1)} \sum_{n=1}^{T-1} p^n (T-n) + \frac{1}{K}. \tag{5.65}$$

We can therefore estimate p numerically by setting the right-hand side of equation (5.60) to be equal to that of equation (5.65). This leads to a polynomial root-finding problem that is easy to solve using readily available algorithms. It is also possible to rewrite (5.65) using formulas for the geometric series and its derivatives, but that complicates the numerics without giving additional insights into the estimated value of p.

The above calculation does not work in the layer-dependent case. Previously, all permutations that put $n-1$ layers between s and t gave the same value for the probability $\mathbb{P}(g_i^s = g_i^t)$, but this is no longer the case when there are different copying probabilities p_{st} for different layers. Instead, one needs to consider every permutation individually to obtain an exact expression for $\mathbb{P}(g_i^s = g_i^t)$. Such a brute-force approach is feasible for a small number of layers (say, $T \lessapprox 6$), but it will not work in general. One avenue for future research is exploring suitable approximations for estimating the parameters p_{st} in this more complicated setting.

5.4.3 Tests on Synthetic Networks

Before proceeding to applications, we test our parameter-estimation approach from Section 5.4.1 on synthetic networks. By running these numerical experiments on networks with known community structure, we are able to evaluate the level of success of our iterative modularity-maximisation algorithm. Another advantage of using synthetic network models is that, for many of them, one can easily tune the strength of planted community structure by changing one parameter. This flexibility allows us to study the community-detection problem across various levels of difficulty in a systematic way.

We consider two types of multilayer benchmarks for our numerical experiments. The first is a simple temporal network model, proposed in [51], in which nodes may

keep or change their community assignment via the same copying and resampling process that we used to establish a prior for $\boldsymbol{g}$. The second is a very general multilayer benchmark, proposed in [17], that allows arbitrary coupling structures between layers and arbitrary degree distributions.

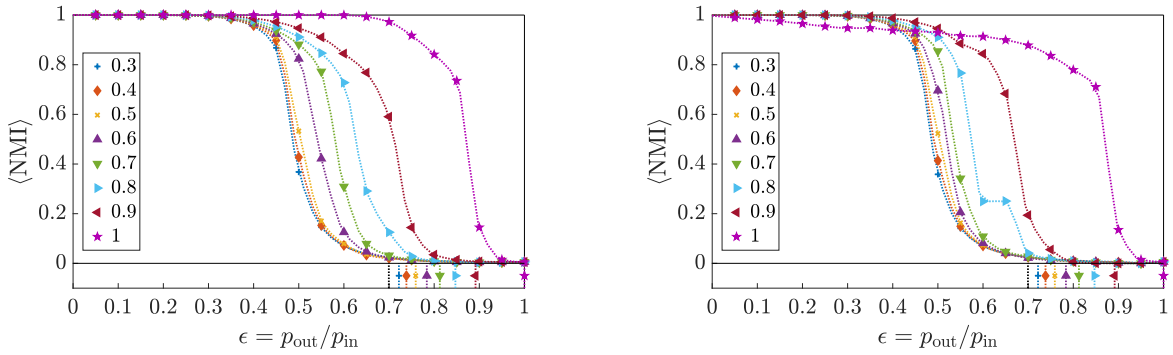
Temporal benchmark

We start with the generative temporal network model that was proposed by Ghasemian et al. [51]. The authors were interested in studying phase transitions in community detection in temporal networks, extending similar work from the study of monolayer networks [37, 38]. Apart from the lack of degree correction, this model is the same as the one that we used in Section 5.3.3 to study multilayer representations of temporal networks. In particular, it uses the same copying and resampling process to propagate community labels across temporal layers. In any layer other than the first, a node keeps its community label from the previous layer with probability η, and it samples a label uniformly at random (from K available labels) with probability $1 - \eta$. The parameter η is the same as our copying probability p, but we use different notation to emphasise that one is a tuning parameter for generating a network and the other is a parameter that we infer while detecting communities in that network. Note that the expected intralayer community sizes are equal to N/K for all communities and all layers.

After assigning nodes to communities in this way, we place edges independently in each layer with probabilities p_{in} (if the nodes are in the same community) and p_{out} (if the nodes are in different communities).[7] Because this generative model has no degree correction, it produces networks in which each node has the same expected degree in every layer. The ratio $\epsilon = p_{\text{out}}/p_{\text{in}}$ controls the strength of the planted community structure.

For their numerical experiments in [51], Ghasemian et al. limited themselves to the case with just $K = 2$ communities, and we do the same in our work. We construct networks with $T = 40$ layers and $N = 512$ nodes in each layer. Each node has a mean degree of $c = 32$, and we vary η and ϵ in our calculations. In qualitative terms, the ability of a community-detection algorithm to recover a planted partition should decrease as ϵ increases (because community structure is becoming weaker), and it should increase as η increases (because the multilayer partition is changing less

[7] Recall from Section 5.3.3 that these edge probabilities are not the same as the parameters θ_{in} and θ_{out} from our model, but they are related by $p_{\text{in}} \approx (c^2/2m)\theta_{\text{in}}$ and $p_{\text{out}} \approx (c^2/2m)\theta_{\text{out}}$, where c is the mean degree in the network.



(a) GenLouvain (b) GenLouvainRand

Figure 5.6: Results for the temporal multilayer benchmark network from [51]. The plots show layer-averaged NMI scores between the planted partition and the one detected by modularity maximisation as a function of community strength ϵ for two different versions of the GenLouvain algorithm [73]. Each line and set of markers corresponds to a different value of the copying probability η, and each data point is a mean over 100 trials. We use the theoretically-optimal γ and ω for each choice of parameters. The bottom parts of the plots indicate the detectability thresholds from [51], beyond which no algorithm is able to detect the planted structure. The plots differ in how the GenLouvain algorithm explores the space of multilayer partitions to increase modularity: (a) choose the move that yields the largest increase in modularity; (b) choose a move with a probability proportional to the resulting increase in modularity. The first algorithm seems to perform better, especially for large η, although we note that this version of the algorithm can lead to poor behavior in certain cases and for sufficiently large ω [18].

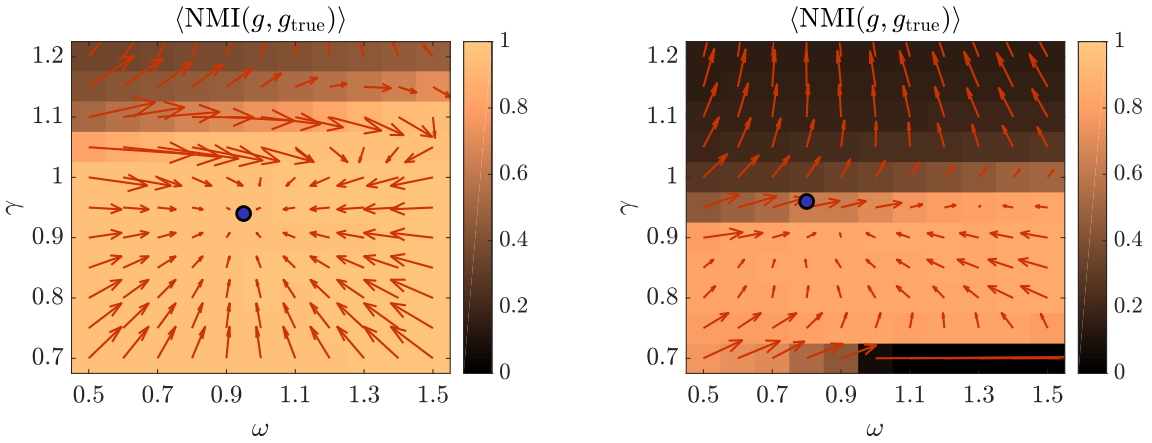
between successive layers). We show the results of these tests in Figure 5.6. Note that our plots use the theoretically-optimal values of γ and ω from equations (5.18) and (5.19), rather than iterating from some initial values. We choose to proceed in this manner because the authors of [51] also assumed that all parameters aside from the community assignments $\boldsymbol{g}$ are known. Towards the bottom of each plot, we also show the detectability thresholds derived in [51]; beyond these values, no algorithm can recover the planted communities. These thresholds depend on η and were derived in the limit in which the number T of layers tends to infinity. We measure the level of success of each run using the layer-averaged NMI (see Section 2.5.1).

The two different plots in Figure 5.6 correspond to two different implementations of the GenLouvain heuristic [73]. (See also our discussion in Section 2.4.1.) When the GenLouvain algorithms seek to increase the value of modularity by reassigning nodes to communities, one can either (1) select the move that leads to the largest increase

in modularity (GenLouvain) or (2) use a "weighted" random move, which is chosen with a probability proportional to the resulting increase in modularity (GenLouvainRand). Previous work [18] found that the original version of the GenLouvain algorithm can sometimes lead to undesirable behavior in which interlayer modularity terms are over-emphasised relative to intralayer terms for sufficiently large ω. The GenLouvainRand version of the algorithm was designed to avoid this problem. For our numerical experiments in Figure 5.6, we find that GenLouvain tends to perform better. This is due to two factors. First, at small values of η, the theoretically-optimal values of ω do not fall in the problematic regime that was described in [18]. Second, at large values of η, the fact that GenLouvain over-emphasises persistence tends to help rather than hinder the algorithm's performance. We also note, in particular, that the results in Figure 5.6 do not contradict the earlier findings from [18].

Our results in Figure 5.6 exhibit the same qualitative features as similar plots in the paper that introduced these benchmarks [51]. Specifically, for each value of η, the success rate of the algorithm (as measured by NMI) is close to 1 for small values of ϵ, and it then drops rapidly to 0 near some critical value of ϵ that depends on η. This degradation in performance occurs before the theoretically-predicted detectability thresholds derived in [51]. In contrast, the belief-propagation algorithm proposed by the authors of [51] has good performance closer to these thresholds. There are several reasons for this observation. The first is that belief-propagation methods are known to give asymptotically optimal accuracy, in that they are able to recover planted communities all the way down to the threshold as long as there is sufficient data (in terms of network size and number of layers) and as long as the relevant parameters are initialized sufficiently close to their optimal values [37, 38, 97]. In contrast, there are no such theoretical guarantees for modularity maximisation. Additionally, the belief-propagation algorithm in [51] only looks for multilayer partitions with two communities, whereas our modularity maximisation algorithm does not. Another possible reason for the suboptimal results in Figure 5.6 may be that the values of γ and ω that we use, which are the ones calculated from equations (5.18) and (5.19), are not the best possible values to use in practice. We investigate this next.

In Figure 5.7, we show heat maps of mean NMI values for two different choices of η and ϵ. Specifically, after generating a multilayer network for each of the two settings, we sample values of γ and ω from a two-dimensional grid and run GenLouvainRand 10 times for each pair of values. The mean NMI values in each grid cell indicate the level of agreement between the true partition and the one that we detect algorithmically. Figure 5.7(a) corresponds to an "easy" case, in which modularity



(a) "Easy" regime ($\eta = 0.7$, $\epsilon = 0.4$)

(b) "Hard" regime ($\eta = 0.5$, $\epsilon = 0.5$)

Figure 5.7: Illustration of the dynamics of the iteration in Algorithm 5.1. In each plot, the heat map shows NMI scores between the planted partition $\boldsymbol{g}_{\text{true}}$ and the algorithmically-detected partition $\boldsymbol{g}$, averaged over 10 runs for each combination of γ and ω values. The arrows centered at each (γ, ω) grid cell indicate the direction of the new γ and ω estimates (averaged over the 10 trials) that we obtain from equations (5.18)–(5.19) after performing modularity maximisation. For clarity, we have scaled down the arrow sizes. The blue disk in each plot indicates the location of the correct parameter values, γ_{opt} and ω_{opt}, which one can calculate from the planted partition. (a) Example with $\eta = 0.7$ and $\epsilon = 0.4$ (an "easy" case). As the arrows indicate, the iteration converges to the true parameter values, $\gamma_{\text{opt}} \approx 0.94$ and $\omega_{\text{opt}} \approx 0.95$, for many initial values $\gamma^{(0)}$ and $\omega^{(0)}$. (b) Example with $\eta = 0.5$ and $\epsilon = 0.5$ (a "hard" case). The region in the plane that results in the highest NMI values is given roughly by $0.8 \lessapprox \gamma \lessapprox 0.9$ and $0.7 \lessapprox \omega \lessapprox 1.5$. The optimal parameter values $\gamma_{\text{opt}} \approx 0.96$ and $\omega_{\text{opt}} \approx 0.80$ lie slightly outside of this regime. Although many of the arrows point in the direction of the blue disk, once the iteration continues from there, it results in increasingly larger resolution values and thus never converges. However, by returning the largest-modularity partition that it encounters during the iterative process, the algorithm identifies a solution that is close to the planted structure. (The NMI is about 0.86.)

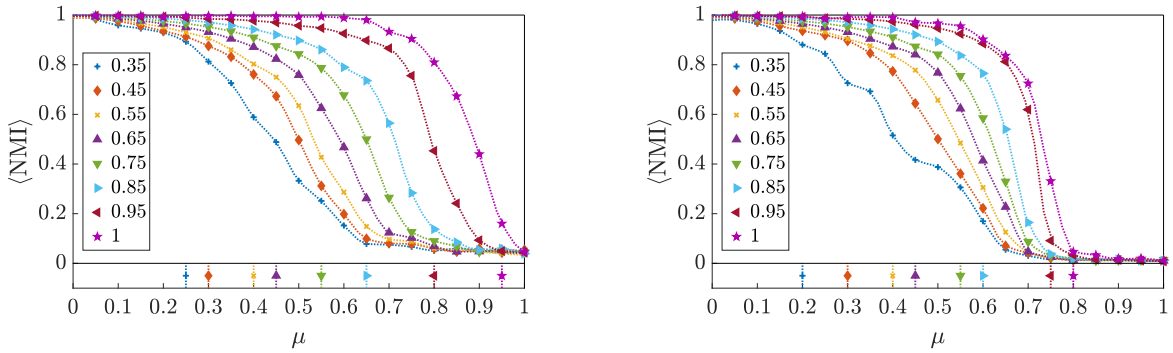
maximisation correctly recovers the planted partition at the optimal parameter values; and Figure 5.7(b) corresponds to a "hard" case, in which we are only partially able to recover the planted structure. (Situations in which there is no overlap between the detected partition and the planted one are not particularly interesting.) In each of these plots, we mark the optimal parameter values γ_{opt} and ω_{opt} with a blue disk. The arrows centered at each (γ, ω) grid point indicate the direction of the new γ and ω estimates (averaged over the 10 trials) that we obtain from equations (5.18)–(5.19) after performing modularity maximisation. (For clarity, we have scaled down

the arrow sizes.) This visualisation indicates how our iterative approach explores (on average) the two-dimensional parameter space from different starting positions. In Figure 5.7(a), most of the arrows point to the location of the optimal point $(\gamma_{\text{opt}}, \omega_{\text{opt}})$. We are thus in the "easy" regime, in which the iteration from Algorithm 5.1 converges to the correct parameter values and recovers the correct partition. The situation is more complicated in Figure 5.7(b). We observe that the optimal parameter values lie slightly outside the region in the (γ, ω) plane that have the largest mean NMI values. Although many of the arrows point in the direction of the optimal point $(\gamma_{\text{opt}}, \omega_{\text{opt}})$, GenLouvainRand detects suboptimal partitions into communities at these values. (We confirmed that the detected partitions are indeed suboptimal, as they give a smaller modularity value than the planted partition.) The arrows also illustrate why the iterative approach fails to converge in this case. From the optimal point, the resolution value becomes increasingly large, with no hope of returning to the large-NMI regime defined roughly by $0.7 \lessapprox \gamma \lessapprox 0.9$. This observation suggests that there is no stable fixed point of the iteration in Figure 5.7, although it is possible that there is an unstable fixed point.

To circumvent the issue of the resolution γ increasing indefinitely for some networks, we have found it useful to decrease γ in the iterations of Algorithm 5.1 whenever the number of communities exceeds some user-specified threshold. Specifically, we decrease γ by 20% (leaving ω unchanged) whenever K exceeds some given $K_{\max}$. This helps our algorithm explore the space of resolution and interlayer-coupling parameters in a useful way, even when it fails to converge to a fixed point. The algorithm can then return the largest-modularity partition that it encounters during this exploration. For the multilayer network in Figure 5.7(b), this approach finds a solution with an NMI score of 0.86. This example illustrates an important point: even when our proposed iterative algorithm does not converge to a fixed point, it can still identify meaningful community structure in multilayer networks. An important topic for future work is to devise better heuristics (or alternatives to the fixed-point iteration of Algorithm 5.1) for this parameter-estimation problem.

General multilayer benchmark

The examples that we have considered thus far consist of temporal networks in which each node has the same expected degree and each community has the same expected size. To test our method on networks with heterogeneous degree distributions and heterogeneous community sizes, both of which are common features of empirical



(a) GenLouvain (b) GenLouvainRand

Figure 5.8: Results for the temporal multilayer benchmark network from Bazzi et al. [17] using GenLouvain and GenLouvainRand to perform modularity maximisation. We use the same parameter values that the authors used in their paper. The resulting networks have $T = 100$ layers and $N = 150$ nodes in each layer. There are $K = 5$ communities, whose expected sizes follow a symmetric Dirichlet distribution with concentration parameter $\theta = 1$; and node degrees in each layer follow a truncated power-law distribution. (See [17] for details.) The plots show layer-averaged NMI scores between the planted partition and the one that we detect by modularity maximisation as a function of the mixing parameter μ. Each line and set of markers corresponds to a different value of the copying probability p_b, and each data point is a mean over 100 trials. We use Algorithm 5.1 to update the values of γ and ω from the initial values $\gamma^{(0)} = 1$ and $\omega^{(0)} = 1$. The lines at the bottom of each plot indicate the values of μ up to which at least 10% of the 100 runs converge to a fixed point.

networks, we turn to the general multilayer benchmark model in [17]. We use the code provided in [72] to generate networks using this generative model.

Similar to the model of Ghasemian et al. [51], the simplest models from [17] have two parameters. A mixing parameter μ controls the strength of the planted community structure; when $\mu = 0$, all edges lie within communities; and when $\mu = 1$, edges are equally likely to lie within communities and between communities. The second parameter of the benchmark from Bazzi et al. [17] is the probability with which a node in one layer copies its community label from a different layer; we denote this probability by p_b to distinguish it from our own parameter p. Importantly, the parameters p and p_b are the same for temporal networks, but they are different for multiplex networks. The discrepancy in the latter case arises because our generative model from Section 5.3.5 is different from the one from [17]. For multiplex networks, Bazzi et al. also introduced the quantity $\widehat{p}_b = (T-1)p_b$ for the probability that a

node copies its community label from one of the other $T-1$ layers. We refer readers to [17] for additional details about these benchmarks.

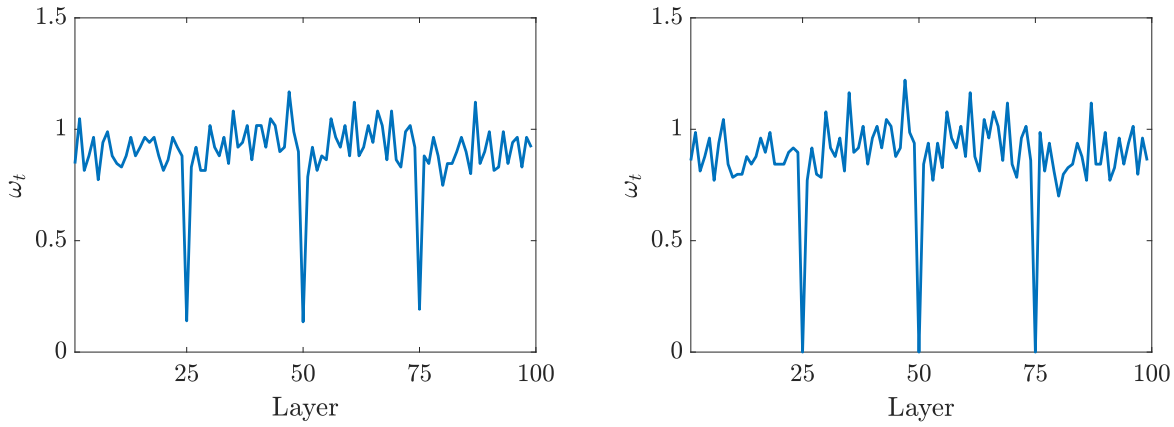
In Figure 5.8, we show the performance of our iterative modularity-maximisation algorithm on the uniform temporal benchmark from [17]. We use the same parameter values to generate networks that Bazzi et al. [17] used for their calculations. The iteration starts from $\gamma^{(0)} = 1$ and $\omega^{(0)} = 1$. The two plots use GenLouvain and GenLouvainRand to maximise modularity; GenLouvain tends to perform better, especially for larger values of the copying probability p_b. The plots from Figure 5.8 have the same qualitative features as those from Figure 5.6, with NMI scores close to 1 for μ close to 0 and rapid degradation in performance for critical values of μ that depend on the value of the copying probability p_b. We also observe a clear separation of the curves that correspond to different values of p_b, with improved performance as p_b increases. In particular, when $p_b = 1$, we see some level of recovery of the planted community structure all the way to $\mu = 1$. Towards the bottom of both plots, we show for each value of p_b the value of μ up to which at least 10% of the 100 runs of IterModMax converge to a fixed point in a maximum of 30 iterations. (Our choice of threshold is arbitrary; the results are similar for values other than 10%.) For smaller values of p_b, the mean NMI between the algorithmically-detected and planted communities remains large beyond these critical values of μ, suggesting the utility of our approach even when the iteration fails to converge. For $p_b = 1$, the critical value of μ appears to coincide with the value of μ beyond which the calculated NMI is close to 0.

In [17], Bazzi et al. fixed $\gamma = 1$ and computed NMI between algorithmically-detected and planted communities for different values of ω. In particular, they did not attempt to select an "optimal" set of parameter values, as this was not a goal of their paper. Using Algorithm 5.1 to select values of γ and ω in a principled way, we obtain NMI values that are comparable to or larger than those that were observed for the analogous calculations in [17] using the best possible ω value (i.e., the one which yields the largest NMI between detected and planted partitions).[8]

Bazzi et al. also examined temporal networks with non-uniform interlayer dependencies, with a focus on networks with change points [17]. We now apply our iterative algorithm with layer-dependent γ and ω values to one such network with $T = 100$

[8]Bazzi et al. used a different version of NMI (one that normalises the mutual information by the joint entropy of the input partitions). Thus, it is not possible to systematically compare our results from Figure 5.8 to theirs. However, we have recalculated some NMI values for specific choices of p_b and μ to enable a direct comparison to [17]. To give an example, for $p_b = 0.85$ and $\mu = 0.6$, we obtain $\langle \text{NMI} \rangle \approx 0.70$ using GenLouvain and $\langle \text{NMI} \rangle \approx 0.65$ using GenLouvainRand. By comparison, Bazzi et al. obtained $\langle \text{NMI} \rangle \approx 0.55$ and $\langle \text{NMI} \rangle \approx 0.40$ for the two versions of the algorithm.

layers using $\mu = 0.4$. We set the copying probability to be $p_t = 0$ for $t \in \{25, 50, 75\}$ (the change-point layers), and we set $p_t = 0.9$ for all other layers. This choice of parameters induces a reorganisation of mesoscale structure in layers 25, 50, and 75 of a sampled network. In Figure 5.9, we show the inferred values of the coupling parameters ω_t using our layer-dependent IterModMax algorithm both (a) after one iteration of the algorithm and (b) upon convergence (i.e., in the final iteration). Our algorithm correctly infers $\omega_t = 0$ at the three change points. The layer-averaged NMI (using the same version of NMI as in [17]) between the output partition and the planted partition is approximately 0.92, compared to a best possible value of 0.8 for the analogous calculation in [17].

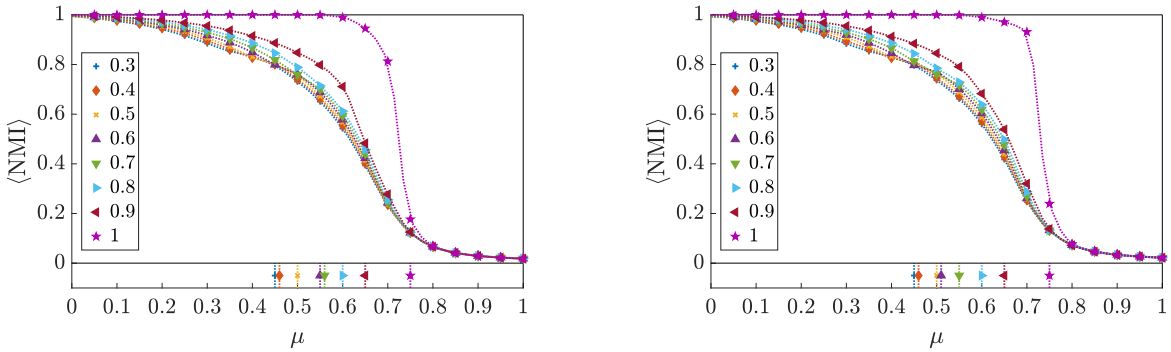


(a) First iteration (b) Final iteration

Figure 5.9: Inferred values of the coupling parameters ω_t for a temporal network with change points after (a) one iteration of the IterModMax algorithm and (b) the final iteration (i.e., upon convergence). We use $p_t = 0$ for $t \in \{25, 50, 75\}$, which induces an abrupt change in community structure at these layers. For the remaining layers, we set $p_t = 0.9$. We let $\mu = 0.4$ for the mixing parameter; other choices give similar results. For the remaining benchmark parameters, we use the same values as in [17]. Specifically, we generate a network with $T = 100$ layers and $N = 150$ nodes in each layer. There are $K = 5$ communities, whose expected sizes follow a symmetric Dirichlet distribution with concentration parameter $\theta = 1$; and node degrees follow a truncated power-law distribution. (See [17] for details.)

We make a few comments about Figure 5.9. First, these results use a version of the IterModMax algorithm that fixes $\gamma = 1$ throughout the iterations. (Due to the large number of parameters in the layer-dependent case, Algorithm 5.1 does not converge when both γ and ω are allowed to vary.) We initialise all coupling parameters ω_t to 1. Second, we use a version of the GenLouvain algorithm without the post-processing step that increases multilayer modularity by relabelling communities to

increase persistence (see [18] and the implementation of GenLouvain in [73] for details). With post-processing, the algorithm is unable to determine that layers 24 and 25 have unrelated community structures (and likewise for the other two change points). Third, although it can take the IterModMax algorithm up to 20–30 iterations to converge to a solution like the one from Figure 5.9(b), it only takes one iteration to pick out the change-point layers, like in Figure 5.9(a). This suggests that waiting for the iterative algorithm to converge may not be necessary if one is interested only in the location of any change points (and not in the extent to which communities reorganise at these change points.)



(a) GenLouvain (b) GenLouvainRand

Figure 5.10: Results for the multiplex benchmark from Bazzi et al. [17] using GenLouvain and GenLouvainRand to perform modularity maximisation. We use the same parameter values that the authors used in their paper. The resulting networks have $T = 15$ layers and $N = 1000$ nodes in each layer. There are $K = 10$ communities, whose expected sizes follow a symmetric Dirichlet distribution with concentration parameter $\theta = 1$; and node degrees in each layer follow a truncated power-law distribution. (See [17] for details.) The plots show layer-averaged NMI scores between the planted partition and the one that we detect by modularity maximisation as a function of the mixing parameter μ. Each line and set of markers corresponds to a different value of the copying probability $\widehat{p}_b = (T-1)p_b$, and each data point is a mean over 100 trials. We use Algorithm 5.1 to update the values of γ and ω from the initial values $\gamma^{(0)} = 1$ and $\omega^{(0)} = 1/T$. The lines at the bottom of each plot indicate the values of μ up to which at least 10% of the 100 runs converge to a fixed point.

In Figure 5.10, we show the performance of our method on the uniform multiplex benchmark network from [17]. As with the temporal benchmark, we use the same parameter values to generate networks that Bazzi et al. used in their numerical experiments. We initialise our iterative algorithm with $\gamma^{(0)} = 1$ and $\omega^{(0)} = 1/T$, where T is the number of layers. (We justify our choice of $\omega^{(0)}$ using the scaling

argument from Section 5.3.5.) For both GenLouvain and GenLouvainRand, and with the exception of the case with $\widehat{p}_b = 1$, we observe only marginal improvement in performance for larger values of the copying probability $\widehat{p}_b$. Bazzi et al. made a similar observation in [17]. For $\widehat{p}_b = 1$, the abrupt drop in performance coincides with the point at which fewer than 10% of runs converge to a fixed point. Our method of estimating the parameters γ and ω gives comparable NMI values to the best values that Bazzi et al. obtained in [17] for the analogous calculations.[9]

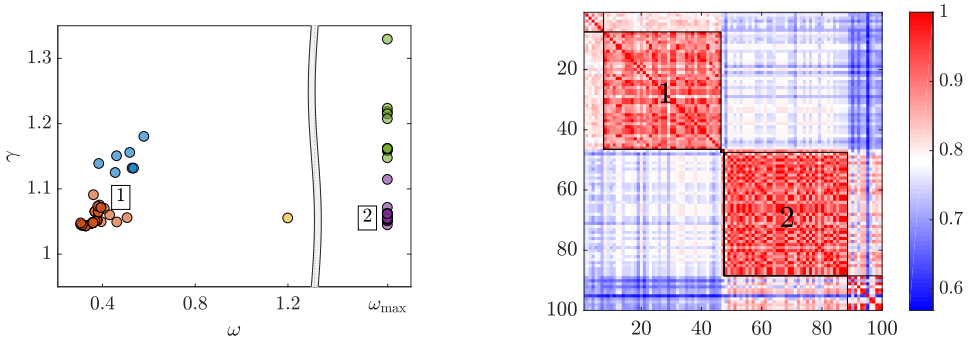
The numerical experiments from this section demonstrate the promise of our approach for estimating resolution and interlayer-coupling parameters, as it yields NMI scores that are comparable to or better than others that have been reported for the same community-detection methods [17]. Even when the iteration process fails to converge, we can use Algorithm 5.1 to explore different parts of parameter space and return the largest-modularity partition that was encountered during that exploration. As we saw in Figure 5.8 and Figure 5.10, our approach often detects communities that are similar to the planted ones. In cases in which our iterative modularity-maximisation algorithm underperforms other community-detection methods (e.g., the belief-propagation algorithm from [51]), it is difficult to say whether this is because of failures of the modularity-maximisation algorithm or shortcomings of our parameter-estimation method.

5.4.4 Application to a Multiplex Social Network

We illustrate our approach on the Lazega Law Firm network [89]. This small multiplex network encompasses interactions between $N = 71$ partners and associates who work at the same law firm. The network has $T = 3$ layers, each with directed edges, which encode *co-work*, *friendship*, and *advice* relationships. The data set also includes 7 pieces of metadata, which we can use to analyse detected communities. These are as follows: (1) status (partner or associate); (2) gender; (3) office (Boston, Hartford, or Providence); (4) seniority (years with the firm); (5) age; (6) practice (litigation or corporate); (7) law school (Harvard, Yale, University of Connecticut, or other). We group both the seniority and age metadata into 5-year bins.

We apply a version of Algorithm 5.1 for directed networks to the Lazega Law Firm data set, and we use GenLouvain for community detection. (Using GenLouvainRand gives very similar results.) Specifically, we ran 100 trials of our iterative

[9] To be consistent with [17], we calculated a second set of NMI values that normalises the mutual information by the joint entropy of the input partitions, rather than by their mean entropy (see Section 2.5.1).



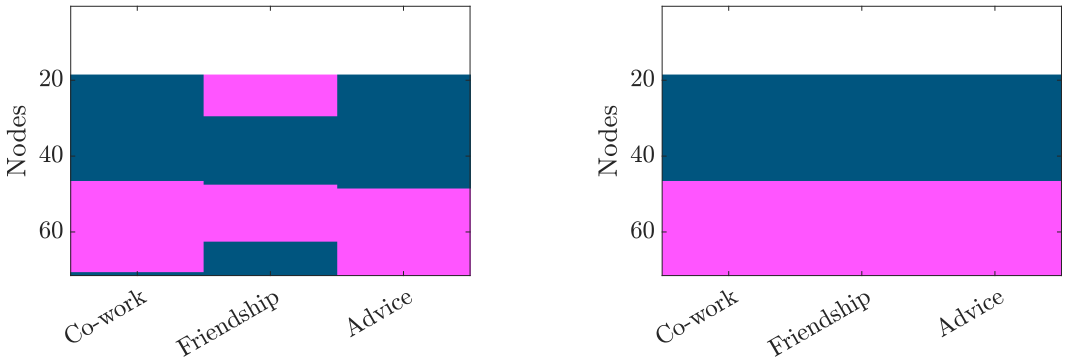
(a) Scatter plot of fixed points

(b) Matrix of NMI scores

Figure 5.11: (a) Scatter plot of (γ, ω) fixed points that our iterative modularity-maximisation algorithm identified when we applied it 100 times to the Lazega Law Firm network. The colors indicate membership to one of 5 clusters that we identified using the K-means algorithm [69]. (b) Matrix of layer-averaged NMI values between all pairs of partitions that we identified across 100 trials of our iterated modularity-maximisation algorithm. The 5 diagonal blocks correspond to the 5 clusters from panel (a).

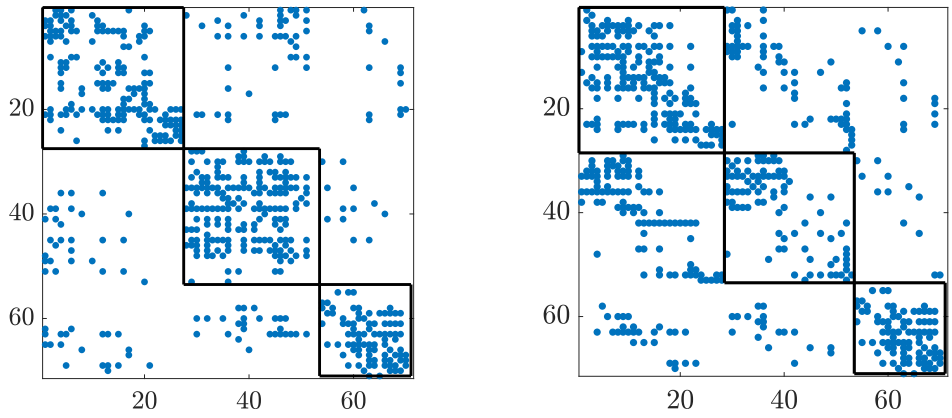
algorithm using initial values of γ and ω that were distributed uniformly in the intervals $[0, 5]$ and $[0, 1]$, respectively. Each trial converged within 30 iterations (which was our specified maximum), and typically our procedure reached a fixed point in a much smaller number of iterations. In Figure 5.11(a), we show a scatter plot of the resulting fixed points $(\gamma_{\text{opt}}, \omega_{\text{opt}})$. This plot has two distinct regions of interest, depending on whether the copying probability p learned by the algorithm is strictly smaller than 1 or equal to 1. In the former case, there is some variation in community structure across layers. In the latter case, communities are identical across all 3 layers, and the theoretically-optimal coupling value is $\omega_{\text{opt}} = \infty$. In the algorithm, we set ω equal to some large value ω_{max} to introduce a very large penalty to nodes that switch communities across layers. For the calculations in this section, we set $\omega_{\text{max}} = 1000$. We apply the K-means algorithm to the (suitably normalised) points in Figure 5.11(a) to assign them to 5 clusters, of which 3 correspond to the runs with $p < 1$ and the other 2 correspond to the runs with $p = 1$. Of the 100 runs, about 80 converge to one of the clusters labelled 1 and 2 in Figure 5.11(a).

A natural question is whether points that are closer together in the scatter plot from Figure 5.11(a) correspond to more similar multilayer community structures than points that are farther apart. We calculate the layer-averaged NMI between all pairs of partitions that we obtained across our 100 trials, and we show the resulting similarity matrix in Figure 5.11(b). We sort the rows and columns so that trials corresponding to



(a) Consensus partition for cluster 1 (b) Consensus partition for cluster 2

Figure 5.12: Visualisations of the consensus partitions that correspond to the two largest clusters from Figure 5.11(a). We use the same ordering of the rows in both panels. The partition in (a) reveals a set of nodes that are in a different community in the friendship layer than they are in the other two layers. The partition in (b) has identical communities in the three layers.



(a) Cluster 1 (b) Cluster 2

Figure 5.13: Sparsity plots of the adjacency matrix for the friendship layer, with rows and columns ordered according to their community assignments in the consensus partitions for (a) cluster 1 and (b) cluster 2.

the same K-means cluster from Figure 5.11(a) are close together. The black borders around the diagonal blocks of the similarity matrix in Figure 5.11(b) correspond to the 5 clusters. We indeed observe larger similarity scores among trials in the same cluster, including for the two dominant clusters labelled 1 and 2. (One of them has $p < 1$, and the other has $p = 1$.)

We use consensus clustering [87] with the null model from [70] to extract a representative partition for the two labelled clusters, and we visualise these consensus partitions in Figure 5.12. As expected from Figure 5.11(a), the partition for cluster 2 has identical communities across the three layers. The structure of the consensus

partition for cluster 1 is more interesting, and it reveals a set of lawyers who are in a different community in the friendship layer than in the co-work and advice layers. In fact, the friendship layer is what distinguishes the two partitions from Figure 5.12(a) and Figure 5.12(b). In Figure 5.13, we show sparsity plots of the adjacency matrix for the friendship layer, ordering the rows and columns of this matrix according to the two consensus partitions from Figure 5.12. The three diagonal blocks in each panel correspond to the three communities. There is a larger number of within-community entries in Figure 5.13(a) than in Figure 5.13(b). This suggests that the consensus partition for cluster 1 identifies stronger assortative structure in the friendship layer than the consensus partition for cluster 2. This simple example perfectly illustrates a key trade-off in multilayer community detection between detecting "optimal" communities in each layer (as the partition for cluster 1 tends to do) and detecting communities that are stable across layers (as is the case for the consensus partition for cluster 2).

Table 5.2: NMI scores between metadata and consensus partitions for the Lazega Law Firm data set. We highlight the largest value in each column.

	Office	Practice	Age	Seniority	Status	Gender	Law school
Partition 1	0.587	0.334	0.146	0.147	0.150	0.035	0.024
Partition 2	0.610	0.469	0.098	0.052	0.040	0.026	0.007
All 100 runs	0.577	0.406	0.125	0.106	0.093	0.037	0.022

We can gain further insight into the two consensus partitions by computing NMI scores with each type of metadata. We give our results in Table 5.2. For comparison, the bottom row of this table gives the mean NMI score across all 100 runs of our iterative algorithm. Our results are in line with a previous study of this data set, which found that the office and practice metadata are more strongly related to network community structure than the law-school or gender metadata [113]. We also observe that the two consensus partitions tend to correlate with different pieces of metadata. Specifically, for cluster 1, we see larger-than-expected NMI scores using age, seniority, and status metadata. For cluster 2, we see larger-than-expected NMI values using office and practice metadata. Recall that differences in the two consensus partitions are driven by variations in the friendship layer. This suggests that age, seniority, and status are important determinants of who is friends with whom. In contrast, office location and type of practice (litigation or corporate) are drivers of co-work and advice relationships.

As this example shows, there is no guarantee that the IterModMax algorithm identifies a single set of optimal parameter values and thus a single multilayer as-

signment of a network's nodes to communities. However, we feel that this is a strength, rather than a shortcoming, of our method. Many empirical networks may have multiple meaningful partitions into communities, which, for example, may align best with different pieces of metadata [113]. Even without the guarantee of a unique output, our method gives a systematic and principled way to reduce the number of cases to investigate using a much smaller number of computations than other similar methods [158].

5.5 Applications to Shopping Data

We now apply our iterative modularity-maximisation algorithms to temporal and multilevel shopping networks. Our goal in this section is twofold. First, we illustrate how our approach works on more realistic (and more challenging) data sets than the Lazega Law Firm network that we examined in Section 5.4.4. Second, we use these methods to derive insights about the multilayer structure of our shopping networks.

5.5.1 Temporal Networks

There are several reasons for studying community structure in temporal shopping networks, rather than examining different time periods independently. In a multilayer setting, community assignments of individual nodes are more consistent over time, which is an advantage over both monolayer community detection and other techniques (e.g., K-means clustering) for customer segmentation that do not take temporal trajectories into account. A second motivation for studying community structure in temporal shopping networks is that it reveals how communities evolve through time, and that, in turn, can uncover meaningful shifts in consumer behaviour. This second goal requires the use of community-detection techniques in which communities are not restricted to be identical across layers. Thus, our key question as we examine the structure of temporal shopping networks is the following: Do communities in these networks change significantly over time, and if so, how?

We use transactions from store `S2` (see Section 3.2.3) to construct a temporal consumer–product network that covers the three-year period from 1 March 2013 to 29 February 2016. We split this data into $T = 12$ non-overlapping 3-month time windows. We use either item-penetration or basket-penetration weighting on the edges (see Section 3.2.2). Given that our methods from this chapter work only for unweighted networks, we then apply a percentile threshold to an initially weighted network. All

results from this section use a 50^{th}-percentile threshold,[10] and we present results for other choices of threshold in Appendix B.1.

In our investigation of these shopping networks, we fix γ and use the IterModMax algorithm to update ω in an iterative fashion. (In these examples, we find that iterating over both γ and ω results in the resolution value increasing indefinitely.) We initially consider $\gamma \in \{0.8, 0.9, \ldots, 1.2\}$ and run 10 trials of IterModMax for each γ value, which stays fixed throughout the iterative process. In Figure 5.21, we show the NMI between each pair of the 50 partitions that we obtain in this way. (We also tried values of γ larger than 1.2, but we found that the NMI values drop abruptly at these resolutions.) In this section, we consider results for $\gamma = 1$ and $\gamma = 1.1$, as we believe that these values strike a balance between identifying partitions that are somewhat consistent across runs (see Figure 5.14) and which contain more than 2 communities (as is the case for $\gamma = 0.8$).

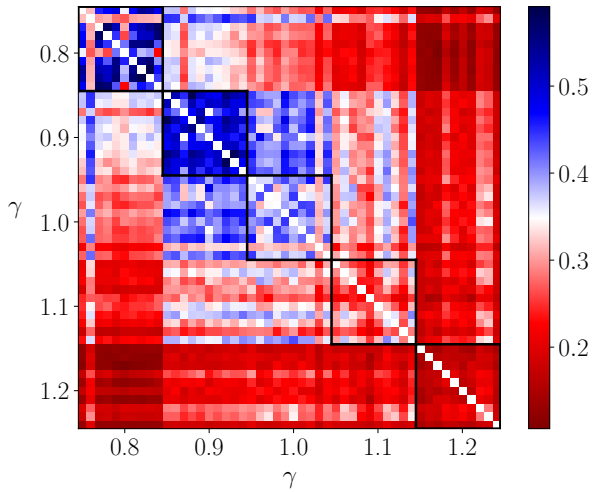
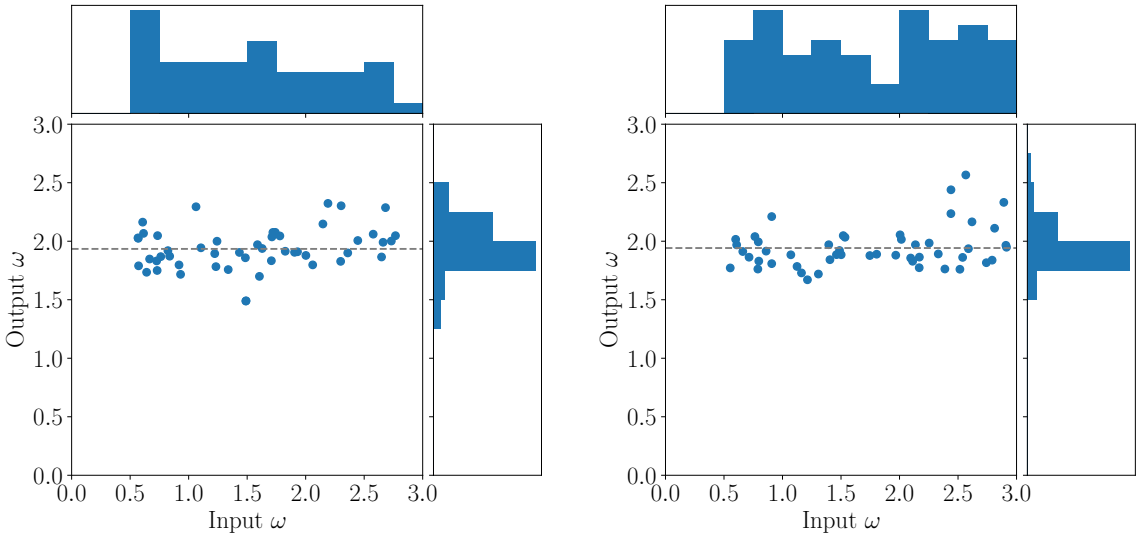


Figure 5.14: NMI values between partitions returned by our iterative modularity-maximisation algorithm for resolution values $\gamma \in \{0.8, 0.9, \ldots, 1.2\}$ and 10 trials for each γ. We observe a significant drop in NMI values for $\gamma = 1.2$.

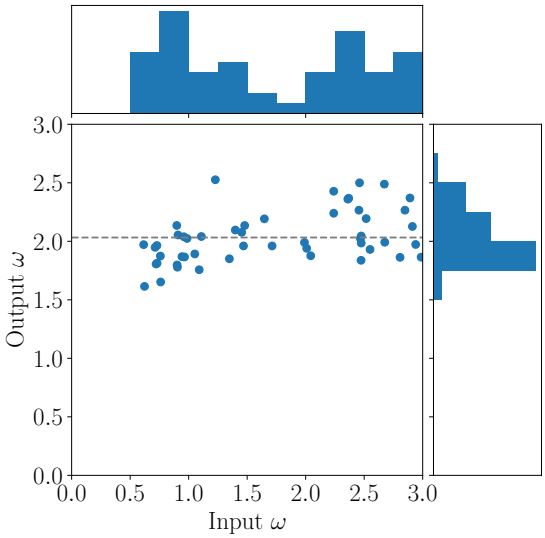
In Figure 5.15, we show the ω values that we obtain with our IterModMax algorithm for these two choices of γ on networks weighted using either item penetration or basket penetration before applying a 50^{th}-percentile threshold. Each panel has results for 50 trials; and in each trial, we sample the initial value of ω uniformly at random in the interval $[0.5, 3]$. Compared to the roughly uniform distribution of the 50 input ω values, the output distribution is much more narrowly peaked in all four

[10]We determine this percentile threshold based on the edge weights from all layers (and not independently for each layer). Because of this, from some layers, we remove more than half of the edges; from other layers, we remove fewer than half of the edges.

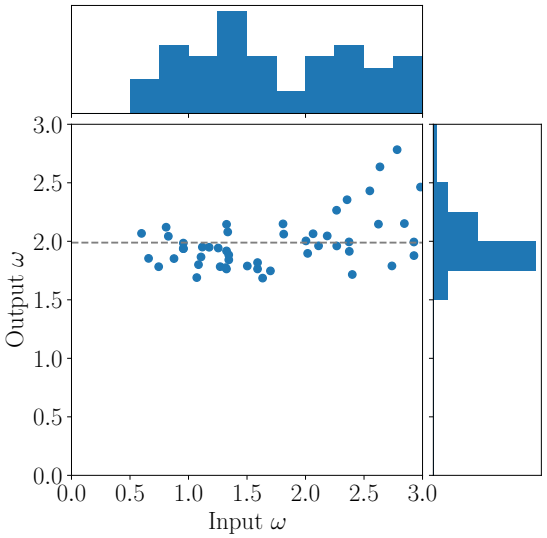


(a) $\gamma = 1.0$, item penetration

(b) $\gamma = 1.0$, basket penetration



(c) $\gamma = 1.1$, item penetration



(d) $\gamma = 1.1$, basket penetration

Figure 5.15: Output values of the interlayer coupling parameter ω across 50 IterModMax trials for two choices of γ and two types of weights.

panels, with a mean output value $\langle \omega_{\text{out}} \rangle \approx 2.0$. (This mean value is slightly larger for $\gamma = 1.1$ than for $\gamma = 1.0$.) Importantly, the panels from Figure 5.15 show little correlation between initial and final ω values. This suggests that the relatively narrow band around $\omega \approx 2.0$ identified by our IterModMax algorithm is meaningful.

Table 5.3: Summary of results for temporal shopping networks.

	Item penetration		*Basket penetration*	
	$\gamma = 1.0$	$\gamma = 1.1$	$\gamma = 1.0$	$\gamma = 1.1$
Trials converged (out of 50)	48	50	49	50
Mean NMI across trials	0.41	0.34	0.38	0.32
$\langle p \rangle_t$ across trials	0.84	0.84	0.83	0.82
$\langle p \rangle_t$ for consensus partition	0.90	0.91	0.90	0.91

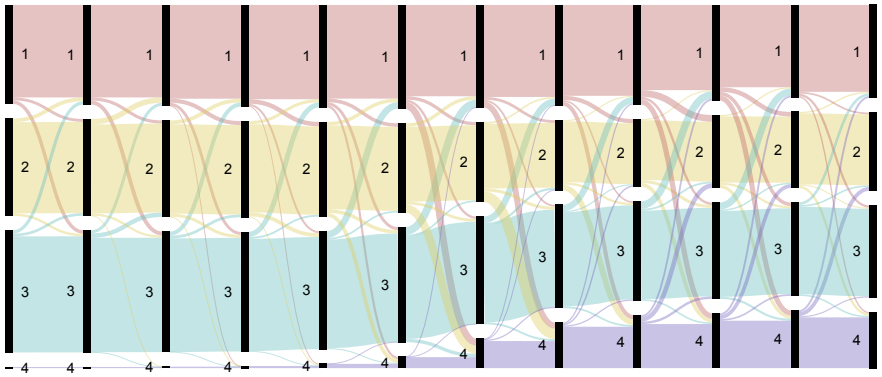
Consensus partitions

Although our iterative approach converges to similar values of ω across different initialisations, the resulting multilayer partitions still show considerable variation across trials. The mean layer-averaged NMI (see Section 2.5.1) varies between approximately 0.32 and 0.41 (see Table 5.3), depending on the weighting and on the γ value. Indeed, the visualisations from Figure 5.16, which depict the output partitions for two (arbitrary) IterModMax trials, show fairly different evolutions of 4 communities through time. It is not clear that one of these solutions is a more accurate representation of the data than the other.

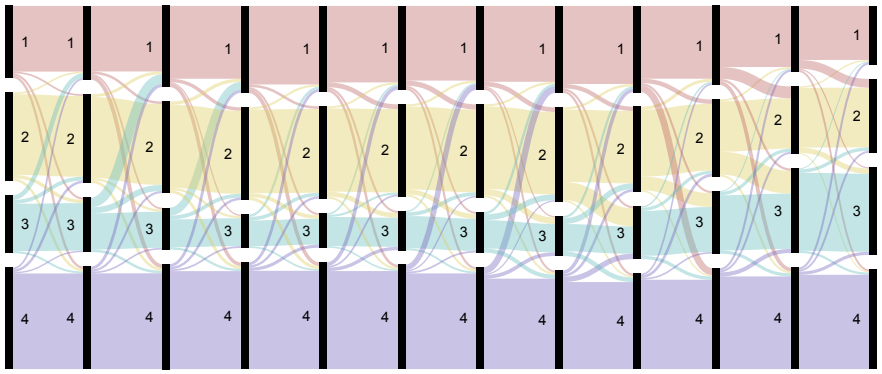
For this reason, we use consensus clustering (see Section 2.4.1) to get a representative partition for each weighting and γ value. As indicated in Table 5.3, the resulting consensus partitions exhibit more persistence than individual trials. Specifically, the estimated value of the copying probability p, averaged across all layers t, is $\langle p \rangle_t \approx 0.90$ for the consensus partitions, compared to about 0.83–0.84 on average across individual trials. This difference is also clear from Figure 5.17, where we plot normalised persistence over time for the consensus partitions against the mean persistence across trials. For the curves in Figure 5.17 that correspond to the consensus partitions, there appear to be no significant increases or decreases in persistence across the 3-year time period; the fraction of nodes that change communities between consecutive layers stays approximately constant throughout the sample.[11]

In Figure 5.18, we show visualisations of the consensus community structure for the two different weightings. Unlike similar visualisations for individual trials, in which some communities appear to grow or shrink over time (see Figure 5.16), the 3 large communities of these consensus partitions have roughly constant sizes. About 8% of the nodes change community membership between consecutive time periods. The

[11]Individual trials can have much more variability in persistence over time, but these fluctuations largely disappear when taking an average across trials or when combining the partitions from all trials into a consensus partition.



(a) Run 1



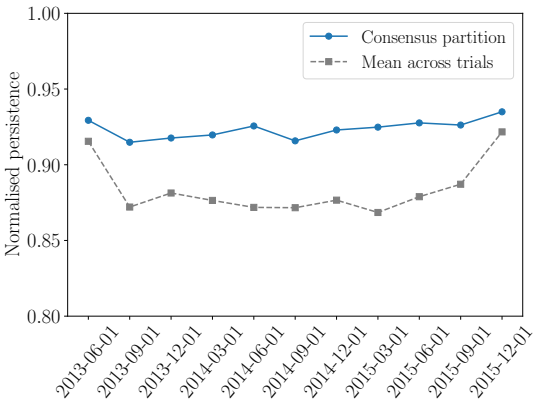
(b) Run 2

Figure 5.16: "Alluvial" plots for two different IterModMax runs using item-penetration weighting (with a 50^{th}-percentile threshold) and $\gamma = 1$. Black vertical bars correspond to the communities in each temporal layer, and coloured areas indicate "flow" of nodes between communities across consecutive layers. Both partitions have 4 communities, but the evolution of these communities through time is very different in the two cases.

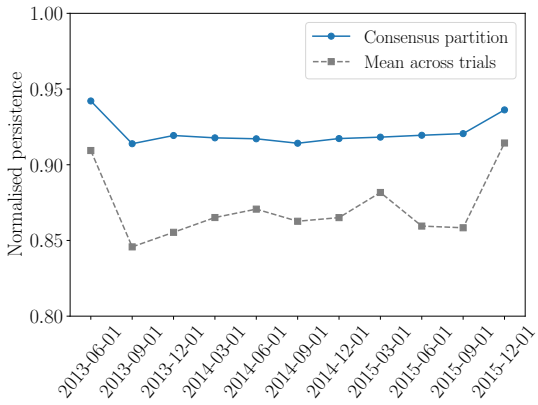
two consensus partitions, which correspond to networks with either item-penetration or basket-penetration weightings, are fairly similar to each other. The NMI between them is approximately 0.56, a value significantly larger than the mean NMI values from Table 5.3 for the similarities across trials.

Comparison to metadata

We characterise the composition of communities in the two consensus partitions by comparing them to customer and product groupings that are based on metadata. We focus on the outputs for $\gamma = 1$ using either of the two weightings. (See Appendix B.1 for analogous results using $\gamma = 1.1$.)



(a) Item penetration, $\gamma = 1$

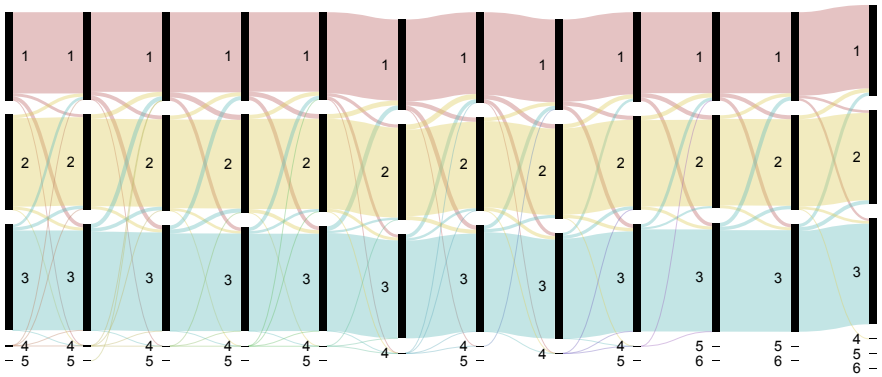


(b) Basket penetration, $\gamma = 1$

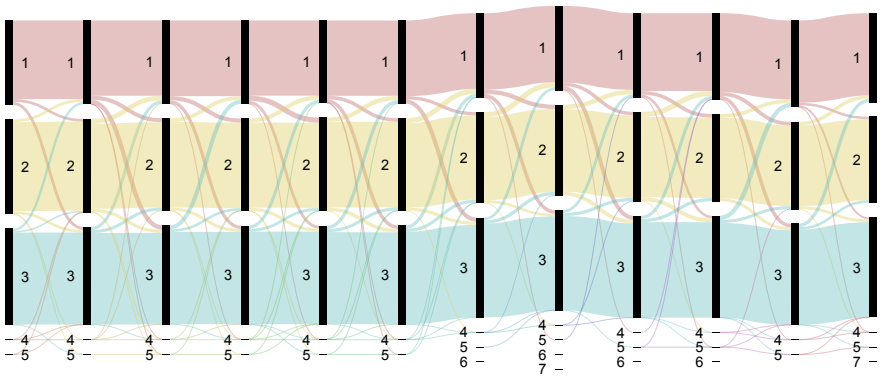
Figure 5.17: Normalised persistence over time for the consensus partitions versus normalised persistence over time averaged across all trials for $\gamma = 1$ and (a) item penetration weighting; (b) basket penetration weighting. For the consensus partitions, there appear to be no large increases or decreases in persistence across the 3-year time period. (We have not investigated whether the small upticks in persistence at the beginning and at the end of the sample are significant.)

For customers, we use the `purchasing-medium` segmentation that we described in Section 3.1.2. This is a classification of consumers into 6 segments based on their recent purchasing habits. The alluvial plots from Figure 5.19 indicate how the algorithmically-detected communities map onto these 6 segments. (In the first two layers of each alluvial diagram, we also indicate the "flow" of customers between communities in the first and last time windows of our temporal network.) The proportion of customers from each of the 3 main algorithmically-detected communities varies considerably across segments. For example, the segment labelled 2, which consists of upscale consumers, contains almost entirely customers from the third (blue) community. This particular segment also emerged as one that correlates strongly with algorithmically-detected communities in our investigation of annotated shopping networks (see Section 4.4). Segment 5 consists of price-sensitive consumers, and the majority of customers in this segment come from the second (yellow) community.

One question is whether the composition of the three large communities in the multilayer consensus partitions changes between the first and the last temporal layers. For example, does the fraction of price-sensitive customers in the second community increase or decrease by a significant amount? We find that community composition is very stable between the first and the last time windows (see Figure B.1 in Appendix B). It is possible that by enlarging the customer sample to include not just those who shop



(a) Item penetration, $\gamma = 1.0$

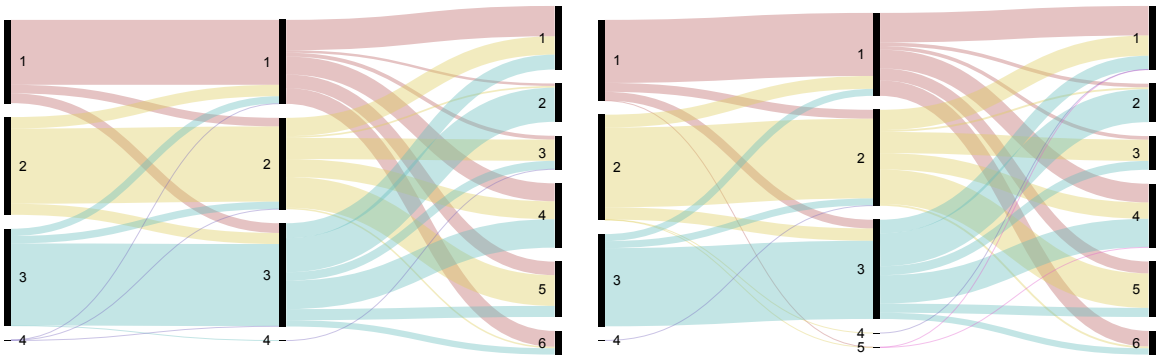


(b) Basket penetration, $\gamma = 1.0$

Figure 5.18: Alluvial plots of the consensus partitions obtained for $\gamma = 1.0$ and either (a) item-penetration or (b) basket-penetration weighting (with a subsequent 50^{th}-percentile threshold applied to both). In each case, there are 3 large communities. Their sizes remain roughly constant through time, but on average about 8% of the nodes change communities between consecutive layers.

very frequently (recall our discussion from Section 3.2.3 about how we constructed this data set), one may observe more variability over time in the composition of these communities.

In Figure 5.20, we show similar visualisations for product nodes, comparing detected communities to `L3` product categories (see Section 3.1.2). (As in the analogous plot for customer nodes, the first two layers of these alluvial diagrams show changes in product communities from the first to the last time windows in our sample.) The consensus partitions corresponding to the two different weightings show similar features; we highlight a few of these. Product categories 5, 6, and 7 correspond to deli counters, meat, and produce; and they include very few products from the second (i.e., yellow) community. (Recall that the second community contains price-sensitive

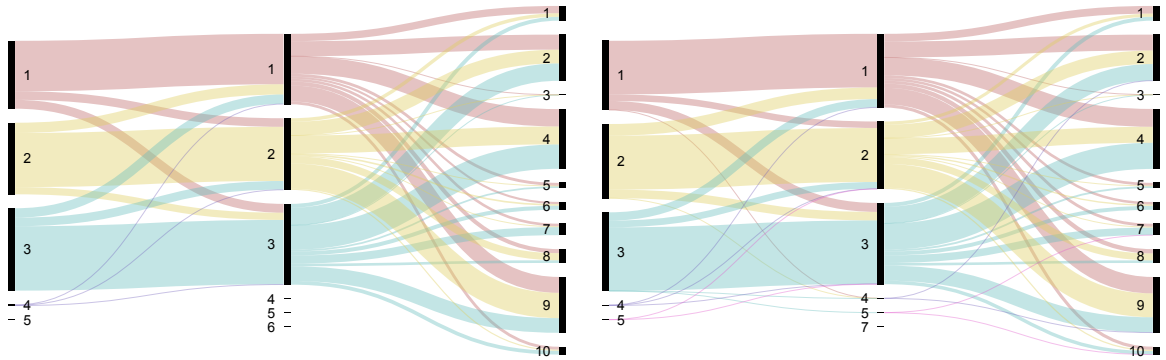


(a) Item penetration, $\gamma = 1.0$ (b) Basket penetration, $\gamma = 1.0$

Figure 5.19: Customer communities in the first and last time windows and their mapping to the 6 `purchasing-medium` segments for $\gamma = 1.0$ with (a) item-penetration weighting and (b) basket-penetration weighting. We observe similar patterns for the two different weightings. The proportion of customers from each of the 3 main algorithmically-detected communities varies considerably across segments. For example, segment 2 contains almost exclusively customers from the blue community, and segments 3 and 5 contain primarily customers from the yellow community.

customers.) In contrast, product category 8, which corresponds to frozen items, has its majority of nodes from the yellow community. This suggests that price-sensitive customers are more likely to buy frozen food compared to other shoppers. These findings are also consistent with our earlier results for annotated shopping networks (see Section 4.4) and single-layer shopping networks (see Section 3.3). Comparing the composition of product communities in the first and last temporal layers of our multilayer network (see Figure B.2 in Appendix B) reveals no significant differences between them.

Returning to our original question, there appear to be no substantial changes in the community structure of the temporal networks that we have analysed, at least at relatively small resolution values. For example, we have not found communities that grow or shrink over time, or communities in which a particular customer type (e.g., upscale) increase in dominance. However, because there is a small amount of flow of nodes between communities across consecutive time periods, it is possible that there are interesting patterns at the level of individual customers or products. For instance, it would be interesting to check what changes in purchasing habits result in one customer node moving from one community to another, and whether or not those moves are long-lived.



(a) Item penetration, $\gamma = 1.0$ (b) Basket penetration, $\gamma = 1.0$

Figure 5.20: Product communities in the first and last time windows and their mapping to the 10 L3 product categories for $\gamma = 1.0$ with either (a) item-penetration weighting or (b) basket-penetration weighting.

5.5.2 Multilevel Networks

We now apply the methods presented in this chapter to a multilevel consumer–product network (see the toy diagram in Figure 5.3). We use transactions from store S2 (see Section 3.2.3) that cover the 3-month time period from 1 March to 31 May 2016. The resulting network has 1462 customers (who are present in both layers of the multilevel network), 4950 products, and 744 product categories. We refer to the layer of the multilevel network connecting customers to product categories as the "category level" or as the "top level"; and we refer to the layer connecting customers to products as the "product level" or as the "bottom level". We generate a weighted network using item-penetration weighting (see Section 3.2.2), and we then apply a 50^{th}-percentile threshold independently in each layer to obtain an unweighted network. In the product level, 14 product nodes become disconnected as a result of applying this threshold; however, for convenience, we keep these nodes in the network.

Similarly to our analysis from Section 5.5.1, we fix γ and use our iterative modularity-maximisation algorithm to optimise ω. We initially consider $\gamma \in \{0.8, 0.9, \ldots, 1.7\}$ and run 5 trials of IterModMax for each γ value, which stays fixed throughout the iterative process. In Figure 5.21, we show the NMI between each pair of the 50 partitions that we obtain in this way. These pairwise NMI values drop abruptly for $\gamma \geq 1.2$, although they start increasing again at about $\gamma \approx 1.6$.[12] In light of these

[12] A reasonable concern is that the uptick in NMI at $\gamma \approx 1.6$ is due to the fact that partitions at these resolution values include a large number of communities, which is known to inflate NMI values. However, computing the adjusted mutual information (AMI), which corrects for such effects, at these resolution values yields the same patterns.

observations, we examine the values $\gamma \in \{0.9, 1.0, 1.1, 1.6\}$ in detail.

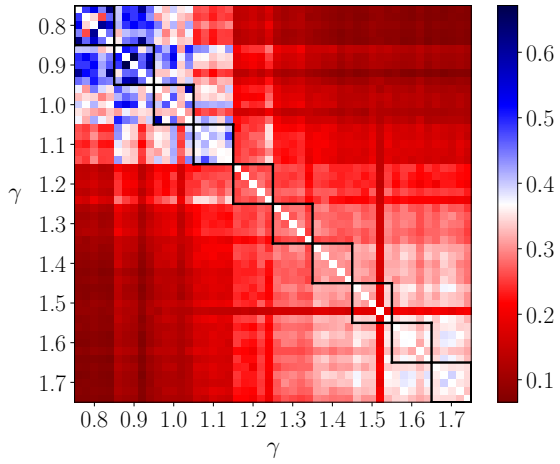
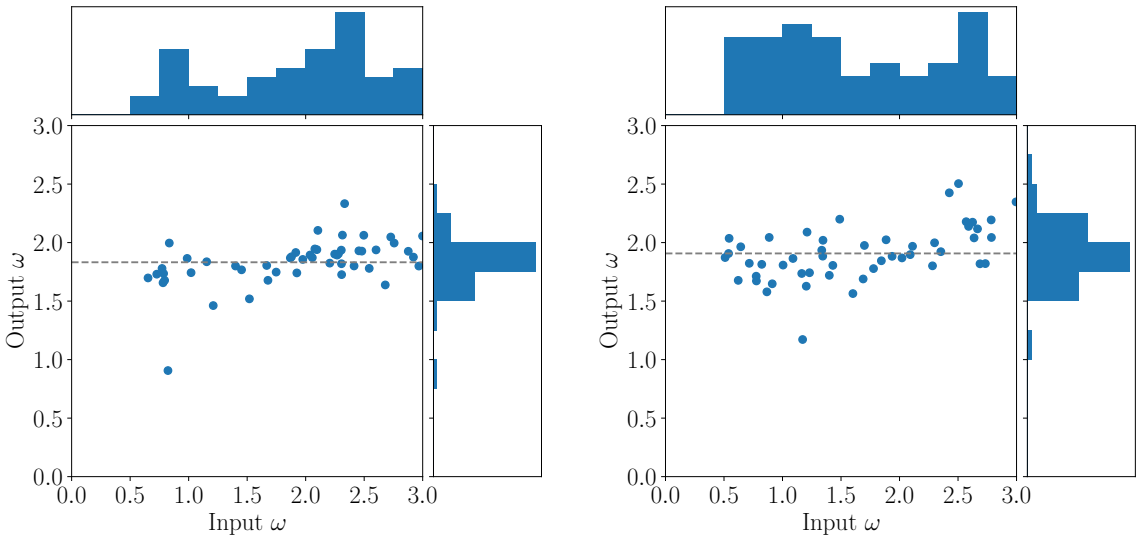


Figure 5.21: NMI values between partitions from our iterative modularity-maximisation algorithm for $\gamma \in \{0.8, 0.9, \ldots, 1.7\}$. We observe a significant drop in NMI values near $\gamma \approx 1.2$, and there is then a slight increase at about $\gamma \approx 1.6$.

In Figure 5.22, we show the ω values that we obtain with our IterModMax algorithm for these four choices of γ. Each panel has results for 50 trials; in each trial, we sample the initial value of ω uniformly at random from the interval $[0.5, 3]$. Compared to the roughly uniform distribution of the 50 input ω values, the output distribution is much more narrowly peaked in all four panels, with a mean output value ranging from $\langle \omega_{\text{out}} \rangle \approx 1.8$ for $\gamma = 0.9$ to $\langle \omega_{\text{out}} \rangle \approx 2.35$ for $\gamma = 1.1$. There appears to be a small positive correlation between input and output ω values. Overall, however, these plots indicate that the IterModMax algorithm converges to a relatively narrow band of ω values, despite the variability among partitions from different trials. (See the NMI matrix in Figure 5.21.)

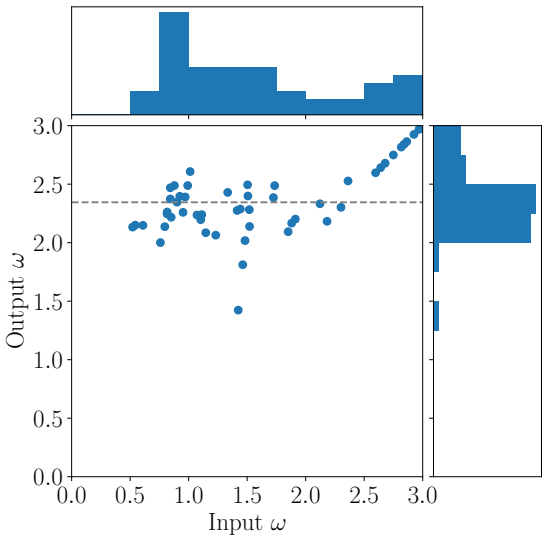
For each value of γ, we use consensus clustering [87] with the multilayer null model from [70] to extract a representative partition. In addition, to assess the benefits of a multilevel representation of our shopping data, we also run single-layer modularity maximisation and determine consensus partitions separately for the product level and the category level.

Our analysis of multilevel consumer–product networks reveals three key findings. First, NMI values between partitions obtained for different runs are consistently larger for multilevel networks than for single-layer networks. Thus, a multilevel representation of transaction data appears to reduce the number of markedly distinct solutions detected by modularity maximisation. This is beneficial, as the modularity objective function typically has many local optima [56]. Second, our IterModMax algorithm

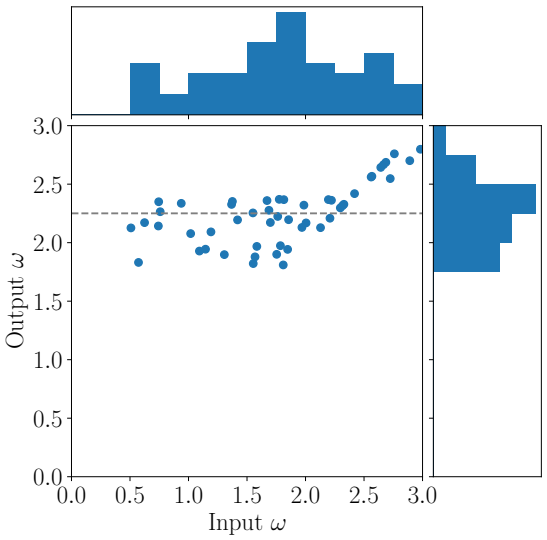


(a) $\gamma = 0.9$

(b) $\gamma = 1.0$



(c) $\gamma = 1.1$



(d) $\gamma = 1.6$

Figure 5.22: Output values of the interlayer coupling parameter ω across 50 IterModMax trials for four choices of γ. The dashed horizontal line in each panel indicates the mean output ω value. For each trial, we sample the initial value of ω uniformly at random from the interval $[0.5, 3]$. Compared to the roughly uniform distribution of the 50 input ω values, the output distribution is much more narrowly peaked in each of the four panels. There appears to be a small correlation between input and output ω values, especially for $\gamma \in \{1.1, 1.6\}$ and at larger input ω values. (Trials in which the iterative approach converges after one step show up as points with $\omega_{\text{in}} = \omega_{\text{out}}$.)

converges to multilayer partitions that have significant levels of persistence: about 85%–92% of nodes (depending on the γ value) have identical community assignments in the two layers. Partitions obtained by maximising single-layer modularity separately in each layer and then matching community labels to maximise contiguity across the two layers yield lower persistence. Third, our analysis corroborates some of our findings from Chapter 4. In particular, we find that certain types of products (such as produce, meat, and frozen items) tend to align with some communities and not others, and we also observe that customer communities split broadly based on price sensitivity. We now discuss these three sets of findings in turn, focusing primarily on the case $\gamma = 1$. We report additional results in Appendix B.2.

Multilevel versus monolayer representations

In Table 5.4, we report pairwise NMI values between the partitions that we obtain for the four choices of γ across 50 trials of either multilevel modularity maximisation (via the IterModMax algorithm) or single-layer modularity maximisation (via the GenLouvain algorithm). For both multilevel and single-layer partitions, we report layer-averaged NMI, as defined in Section 2.5.1. These NMI values are larger for multilevel partitions than for monolayer partitions, suggesting that there is more consistency across trials in the former case than in the latter. Additionally, we find that the monolayer approach results in many small communities, especially for larger γ (and especially for the product level). We obtain fewer small communities with a multilevel approach.

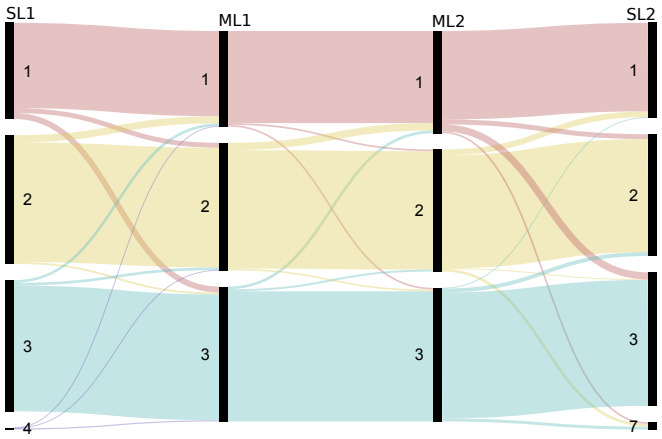
Table 5.4: Mean pairwise NMI between partitions that we obtain for 50 trials of IterModMax (for multilevel networks) and GenLouvain (for single-layer networks).

γ	Multilevel partitions	Single-layer partitions
0.9	0.47	0.45
1.0	0.46	0.36
1.1	0.38	0.27
1.6	0.36	0.24

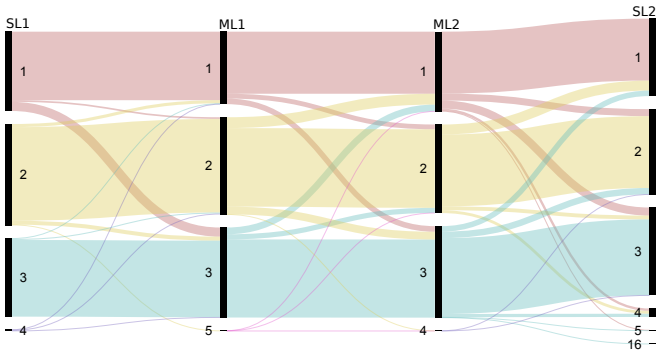
Focusing now on the multilevel (ML) and single-layer (SL) consensus partitions that we obtained for each value of γ, one question is how much they differ from each other. In other words, does employing community detection on a multilevel network yield a vastly different partition from analysing each layer separately? In Table 5.5,

Table 5.5: NMI between multilevel (ML) and single-layer (SL) partitions.

γ	*ML versus SL (category level)*			*ML versus SL (product level)*		
	Customers	Products	Both	Customers	Products	Both
0.9	0.77	0.70	0.74	0.52	0.36	0.39
1.0	0.71	0.73	0.72	0.71	0.45	0.50
1.1	0.62	0.67	0.63	0.57	0.32	0.36
1.6	0.34	0.39	0.33	0.38	0.20	0.22



(a) Single-layer and multilevel customer communities



(b) Single-layer and multilevel product communities

Figure 5.23: Alluvial diagrams of consensus communities for $\gamma = 1$, separately for (a) customer nodes and (b) product nodes. (See Appendix B for analogous visualisations for other values of γ.) In both panels, the two middle layers correspond to the multilevel consensus partition, and the outer layers correspond to the single-layer consensus partitions. (The category level is on the far left, and the product level is on the far right.) For clarity, we do not show communities with fewer than 5 nodes. There appears to be more "flow" between communities for product nodes than for customer nodes, especially for the product-level partitions.

we report NMI scores[13] between ML and SL consensus partitions separately for the category-level and the product-level networks. We also show visualisations of ML and SL consensus partitions for $\gamma = 1$ in Figure 5.23. The largest difference between these partitions occurs among product nodes in the product-level network. Even in this case, however, community labels for the ML and SL partitions overlap for a majority of the nodes. It follows that partitions detected using a multilevel network structure are similar to those found using single-layer networks (especially at lower resolution values), while at the same time bringing other benefits such as higher consistency across trials.

Persistence in multilevel versus single-layer partitions

By construction, we expect there to be more persistence in multilayer partitions detected directly from a multilevel network structure than in partitions determined separately for the two layers, and then post-processed (i.e., via a reassignment of community labels) to maximise overlap. The results from Table 5.6 indicate that this is indeed the case. Customer nodes have normalised persistence values that are consistently larger than the values for product nodes. However, even for product nodes, normalised persistence values of approximately 0.85 suggest that the community labels of 85% of nodes are inherited from the category level. (These persistence values are for the consensus partitions, rather than for individual trials.) Post-processed single-layer partitions have much smaller persistence values, especially for $\gamma = 1.6$.

Table 5.6: Normalised persistence across layers for a consensus multilevel partition and for post-processed single-layer consensus partitions.

γ	*Multilevel partition*			*Single-layer partitions*		
	Customers	Products	Both	Customers	Products	Both
0.9	0.96	0.87	0.89	0.71	0.57	0.60
1.0	0.95	0.84	0.87	0.82	0.64	0.68
1.1	0.96	0.86	0.88	0.76	0.58	0.62
1.6	0.98	0.89	0.91	0.52	0.36	0.40

Recall that one of our motivations for performing community detection on multilevel shopping networks is that the right level of granularity for aggregating transac-

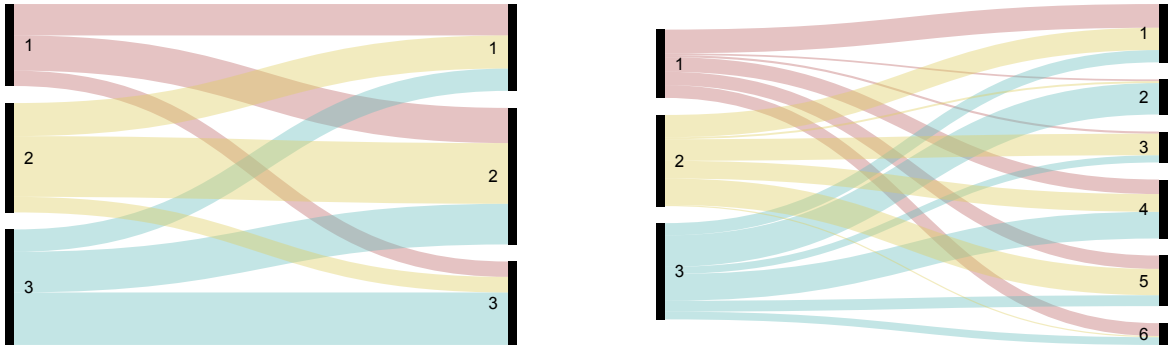
[13]One needs to be cautious about comparing NMI scores when the number of nodes is variable. In particular, our category-level network has fewer nodes (1462 customers and 744 product categories) than our product-level network (1462 customers and 4950 products). Nevertheless, the same patterns that we report in Table 5.5 also hold if we look instead at normalised persistence, which is equal to the fraction of nodes that have identical community assignments in two given partitions.

tions is not a priori clear. For example, we asked whether loose bananas should be considered as a different product from packed bananas. Our investigation suggests an affirmative answer to this question, as loose bananas end up in a different community from packed bananas in the product layer of our multilevel network, even though they map to the same node in the category layer. (This occurs at all four resolution values that we examined.) By contrast, red grapes and green grapes always appear in the same community, which suggests that it may be appropriate to combine these products into a single node. Taking this further, one can potentially use detected multilevel communities to construct a single-layer network that uses an "optimal" level of aggregation for each product (e.g., different nodes for loose bananas and packed bananas, but a single node for both red grapes and green grapes). This type of network compression may also be useful for other multilevel networks (e.g., trade networks, where nodes are increasingly large agglomerations of countries as one moves up a given hierarchy). However, we do not investigate this issue further in our work.

Comparison to metadata

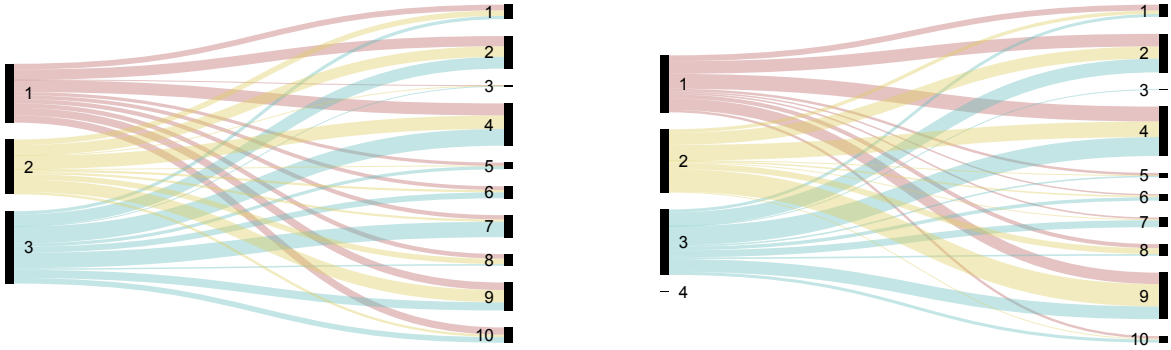
We now compare our multilevel consensus partition for $\gamma = 1$ to existing customer and product metadata. (Appendix B has similar visualisations for other values of γ.)

For customers, we use the purchasing segments that we described in Section 3.1.2. Recall that this customer segmentation comes in three hierarchical levels, which we called `purchasing-low` (13 possible labels), `purchasing-medium` (6 possible labels), and `purchasing-high` (3 possible levels, which correspond roughly to price-sensitive, mid-range, and high-end consumers). The two panels of Figure 5.24 show how customer nodes in the restriction of the multilevel consensus partition to the product level of the network map onto `purchasing-high` and `purchasing-medium` customer segments. In Figure 5.24(a), we highlight that the segment of high-end consumers, labelled 3 in the diagram, consists primarily of customers from the third community (shown in blue). By contrast, the segment of price-sensitive consumers, labelled 1 in the diagram, consists primarily of customers from the first two communities (in red and yellow). Looking at the finer segmentation in Figure 5.24(b), we observe that segment 2, which is regarded as the most upscale of the 6 categories in the `purchasing-medium` segmentation, consists almost exclusively of customers from the third (blue) community. In previous analyses, including our work with annotated networks from Chapter 4 and our study of temporal communities in Section 5.5.1, this particular segment has repeatedly emerged as one that correlates strongly with algorithmically-detected communities.



(a) Comparison to `purchasing-high` (b) Comparison to `purchasing-medium`

Figure 5.24: Alluvial diagrams of consensus customer communities for (a) `purchasing-high` and (b) `purchasing-medium` customer segments for $\gamma = 1$. (See Appendix B for analogous visualisations for other values of γ.)



(a) Category level (b) Product level

Figure 5.25: Comparison of (a) category-level and (b) product-level consensus partition to `L3` product categories for $\gamma = 1$. (See Appendix B for analogous visualisations for other values of γ.)

In Figure 5.25, we give similar visualisations for product nodes. Specifically, we restrict the consensus multilevel partition to product nodes and subsequently to each of the two network levels. We then compare the two resulting monolayer partitions to `L3` product categories (see Section 3.1.2). We highlight a few patterns in these visualisations. The `L3` produce category, which has the label 7 in Figure 5.25, consists almost exclusively of product nodes from the third (blue) community. The meat category, labelled 6 in the figure, is also primarily composed of nodes from the blue community. By contrast, the frozen-foods category, which has the label 8, has very few nodes from this community. These patterns agree with our findings for annotated

networks from Section 4.4.

5.6 Summary and Discussion

Many recent developments in community detection have focused on multilayer networks, which one can use to encode time-dependent interactions, multiple types of interactions, and other complications that arise in complex systems. There exist a variety of methods for community detection in multilayer networks. In this chapter, we focused on two approaches: modularity maximisation and statistical inference in stochastic block models. Our main results connect these two types of methods. By considering various types of multilayer structure, we showed that the multilayer modularity objective function is, under specific circumstances, the same as the posterior log-probability that corresponds to certain multilayer SBMs. Therefore, in these circumstances, maximising one expression with respect to community assignments is equivalent to maximising the other.

This link between the two approaches highlights some implicit assumptions of classical multilayer modularity that previously were not clear. These include the assumption that layers are "statistically equivalent" (which means that they are described by the same sets of parameters, θ_{in} and θ_{out}), unless one uses our proposed layer-weighted version of modularity, and that all communities in a layer have the same expected sizes (because the null distributions $\mathbb{P}_0^t$ are uniform and the copying probabilities p_t are the same for each node in a layer). Our result also suggests a principled method for determining appropriate values for the resolution and interlayer-coupling parameters that arise in the formulation of multilayer modularity. Compared to previous approaches for parameter selection [15, 158], our method is more theoretically grounded, and it also requires fewer computations, thereby improving the practicality of the method of multilayer modularity maximisation for large networks. We note that one cannot use existing approaches based on SBMs to infer communities that vary in arbitrary ways across layers, making multilayer modularity maximisation one of the few methods available for when one needs such a level of generality. (Another appropriate method is multilayer Infomap [36].)

We tested our iterative modularity-maximisation algorithm on two families of multilayer benchmark networks. Our results showed the expected qualitative features, with better performance in networks with stronger community structure and more persistent communities across layers. For the temporal and multiplex benchmarks from [17], the NMI scores that we obtained using our parameter-estimation algorithm

are generally larger than those from the original paper. We also tested our method on the Lazega Law Firm data set, which is a multiplex network with three layers. By running Algorithm 5.1 using different initialisations, we identified two fixed points to which the majority of the runs converged. One of these fixed points corresponds to community structure that is identical across layers. The other reveals a set of lawyers who belong to a community in the friendship layer that is different from their community in the co-work and advice layers.

One shortcoming of Algorithm 5.1 is that it fails to converge to a fixed point in networks whose community structure is not sufficiently strong. In such cases, one of our suggestions was to use the partition with the largest modularity value out of all partitions encountered during the iterative process. Because (based on our observations) failure to converge is almost always due to the resolution γ increasing indefinitely, an alternative solution is to fix γ throughout the iterations and only optimise ω.[14] We chose this approach for our investigation of temporal and multilevel consumer–product networks. Our main findings for these networks were that (1) communities in temporal shopping networks are relatively persistent over time, with only about 8% of nodes changing communities across consecutive time periods; and (2) the level of aggregation for products (barcode level versus "Level 1" categories) has a significant but relatively small effect on detected communities, especially when considering a multilevel network rather than the two corresponding monolayer networks.

The three types of multilayer structure that we examined — temporal, multilevel, and multiplex — encompass extremely common examples that arise in community-detection applications. We examined both the situation in which the resolution and interlayer-coupling parameters are uniform across all layers and the one in which they are layer-dependent. In the process of deriving our results in the latter case, we proposed a novel layer-weighted version of modularity that is appropriate for situations in which communities in different layers have different statistical properties. We also briefly explored the possibility of using the layer-dependent models to detect some types of (structural) change points [112], which correspond to layers in which a network's mesoscale structure undergoes a major reorganisation. For multiplex networks, we proposed a new way of generating multilayer partitions by permuting the layers and then updating community labels in the order indicated by the permutation.

[14] One promising research direction is to combine a version of the IterModMax algorithm in which γ is kept constant with the multiresolution consensus-clustering approach from [75]. Specifically, one would use our iterative algorithm to obtain a partition (i.e., an optimal ω) at each value of γ, and then one would apply the method from [75] to obtain a hierarchy of consensus partitions spanning multiple scales.

In contrast to other generative models of multiplex networks [17], our approach makes it possible to write down the probability of generating a particular multilayer partition in closed form, and we expect that this will be useful for future efforts (e.g., for performing inference in the resulting SBMs).

There are many possible extensions of this work. One can derive similar results for multilayer networks with different forms of interlayer coupling from those that we discussed in this chapter. Examples include temporal networks with memory, in which a layer depends not just on its predecessor, but on all other layers that came before it; networks with community-dependent coupling parameters; interconnected networks, in which interlayer connections are no longer diagonal; and multilayer networks with more than one aspect (e.g., ones that are both temporal and multiplex [78]). More broadly, our work (as well as Newman's previous work [105]) provides a recipe for deriving new versions of modularity for both monolayer and multilayer networks by starting from an SBM and making appropriate simplifications. Previous authors have used such a procedure to propose a version of modularity that incorporates information from metadata [161] by simplifying the SBM from [109]. We expect that one can similarly derive modularity functions from other SBMs that incorporate node annotations [65, 169] in both monolayer and multilayer settings. Another idea is to incorporate parameters that control expected community sizes in the definition of modularity, effectively relaxing the assumption that the null distributions $\mathbb{P}_0^t$ are uniform [167]. One could also replace the planted-partition model (in which there are just two parameters, θ_{in}^t and θ_{out}^t, in each layer) with a multiscale model with three parameters (in which θ_1^t gives the probability of an edge at the smallest scale, θ_2^t gives the probability at an intermediate scale, and θ_3^t gives the probability at the largest scale). By following a similar process to the one that we presented in this chapter, one can derive a new type of modularity for detecting such hierarchical structure.

It would be interesting to study the dynamics of the iteration that forms the core of Algorithm 5.1 (and possibly design better algorithms based on those insights). One can use ideas from continuous optimisation to develop an algorithm that explores the parameter space in a more robust way. For example, rather than using a single run of modularity maximisation to estimate new resolution and interlayer-coupling values, one can combine information from multiple runs to choose a direction to move and a step size to use in a move in that direction. One such approach is to use consensus clustering to obtain a representative partition at each step of an iteration; however, that would increase computational time by a significant amount.

Another challenging, but important, task is to extend the results of this chapter to weighted networks. It is not obvious how one can relate existing weighted SBMs, such as those in [3, 119], to weighted modularity. Instead, we proposed two approaches for how one might determine ω in weighted networks. Developing these ideas further would be extremely useful for many applications.

In summary, we showed that two common and independently-developed methods for community detection in multilayer networks — modularity maximisation and SBM-based statistical inference — are closely related. This connection goes far beyond being only an interesting theoretical finding, as the techniques that we developed in this chapter can provide practical insights into devising more robust community-detection algorithms that do not require ad-hoc experimentation with parameter values.

Chapter 6

Introducing Edge Correlations into Multilayer Networks

6.1 Motivation

A key assumption of nearly all existing models of multilayer networks is that edges are generated independently conditioned on a multilayer partition [17, 34, 51, 116, 121, 145, 150, 153]. This independence condition applies both within each layer (thus ignoring the tendency of real networks to include 3-cliques and other small-scale structures) and across layers (thus ignoring scenarios in which the same nodes share connections in multiple layers). This assumption is explicit in the case of generative models, such as standard SBMs, that sample edges independently. We saw in Chapter 5 that maximising modularity is equivalent to maximising the posterior probability of a special case of a multilayer SBM, which implies that modularity shares the same limitations. When a set of nodes persists in the same community across multiple layers, this indicates only that they tend to share many edges between them, but not necessarily that the edges themselves are predominantly the same.

In this chapter, we focus on relaxing the edge-independence assumption that applies to edges between the same two physical nodes in different layers. (We still consider each pair of nodes independently.) In Figure 6.1, we show an example of a two-layer network with strong edge correlation (either positive or negative). Incorporating such correlations into a network model is beneficial for many applications. For example, a multiplex network of air routes, where each layer corresponds to one airline, is likely to include some popular routes that appear in multiple layers [26]. In a temporal social network, we expect people to have repeated interactions with other people [8]; this is a stronger statement than just saying that they tend to interact within the same community over time.

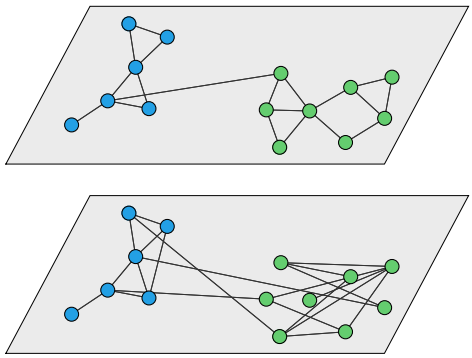


Figure 6.1: Toy example of a correlated network with two layers and with two blocks of nodes in each layer. Edges among blue nodes are positively correlated, edges among green nodes are negatively correlated, and edges between a blue node and a green node are uncorrelated.

For shopping networks, we expect significant "edge persistence": people tend to buy not just the same types of products over time (e.g., cereal), but specific brands within those wider categories (e.g., Cheerios). Of course, the level of loyalty is likely to vary across customers, with some more consistent in their shopping and others more adventurous. We also expect the level of persistence to vary across different product types. This complexity is not captured adequately by existing models.

Before introducing edge-correlated models in the next few sections, we mention four areas of application for our work: graph matching, edge prediction, computation of new network diagnostics in the form of layer correlations, and detection of correlated communities.

In the context of multilayer community detection, assuming edge independence is common, because it makes the resulting models and algorithms significantly more tractable. In community detection, the emphasis is on nodes rather than on edges (because communities are sets of nodes), so the assumption is acceptable. For other applications, such assumptions are no longer reasonable. Consider the *graph matching* problem [32], where one seeks to infer a latent correspondence between nodes in two different networks when the identities of the nodes are unknown. For example, one may wish to match common users between anonymised Twitter and Facebook networks.[1] Two people who are connected on Facebook are more likely to also be connected on Twitter, and vice-versa, so taking advantage of edge correlations makes the graph-matching problem feasible. In a series of papers [94–96], Lyzinski and collaborators

[1]This problem was made famous (or infamous) in the context of the Netflix Prize, where some researchers were able to deanonymise a Netflix dataset of users with movie reviews by combining it with public data from the Internet Movie Database (IMDb) [100].

established conditions under which graph matching is successful, including situations where some nodes are "seeded" (their identities are known) or "junk" (they cannot be matched because they only appear in one of the networks). In their work, they considered correlated Erdős–Rényi (ER) networks and correlated SBMs, which we also investigate in this chapter (albeit for a different purpose). We take these models further by incorporating degree correction and relaxing the assumption that blocks (i.e., communities) are the same in different layers. These extensions may allow one to study the graph-matching problem on more realistic network models. Degree correction is especially important, as real networks exhibit highly heterogeneous degree distributions [76].

Modelling edge correlations is also useful for the task of *edge prediction*,[2] where one seeks to assign probabilities of occurrence to unobserved or missing edges (see also Section 2.5.2). In bipartite shopping networks, this process results in a set of new product recommendations to customers. SBMs have long been used for edge prediction, both for single-layer networks [3, 58, 118] and for multilayer networks [34, 148, 153]. The models that we propose, which take advantage of cross-layer edge correlations in data, should perform even better on this task. We discuss this in more detail in Section 6.5.

A third application of our work is the computation of novel network diagnostics in the form of correlations between pairs of layers of a multilayer network. One can then use these correlation estimates for several purposes. For example, they can be used to compress multilayer network data, by discarding (or merging) layers that are strongly positively correlated with an existing layer, as these additional layers provide little new information. In feature-based machine learning, correlation values can be additional features that supplement more standard network diagnostics such as clustering and assortativity coefficients, mean geodesic distance, modularity, and various quantities determined from degree distributions. Exploring either of these two applications is beyond the scope of this thesis; instead, we focus on deriving estimates of these correlations in different scenarios (see Sections 6.2–6.4).

Our proposed edge-correlated network models can also be used for community detection. That is, given a multilayer network, one can design a full inference algorithm that determines both the parameters that describe edge probabilities (and correlations) and a multilayer community structure that underlies these probabilities. Solving this inference problem would enable the detection of "correlated communities". Because

[2]This task is commonly called "link prediction" in the literature, but we tend to use the phrase "edge prediction" in our work.

of the additional complexity in the model, this is bound to be more difficult than standard multilayer community detection. We leave this inference problem for future work. Instead, for the rest of the chapter, we assume that the block structure of a network is known and we infer the remaining parameters, including the correlations that are the core element of our model.[3]

The models that we describe in this chapter are similar to those introduced in [94–96] for graph-matching purposes. We also mention a paper by Barucca and collaborators [13] that described a generalised version of the temporal model by Ghasemian et al. [51] that we discussed in Section 5.4.3. This generalisation includes an "edge persistence" parameter ξ, which gives the probability that an edge from one layer also occurs in the next temporal layer. For several reasons, we take a different approach. First, the model of [13] is specific to temporal networks, whereas we are also interested in other types of multilayer networks. Second, their model does not easily incorporate degree correction. Third, we want to include correlations explicitly in the model, rather than implicitly using the edge persistence parameter ξ.

The rest of this chapter is organised as follows. We start with a simple example of correlated ER graphs in Section 6.2, to make our exposition easier to follow for more complicated models. In these ER graphs, we derive maximum likelihood (ML) estimates both of the marginal edge-existence probabilities in each layer and of the cross-layer correlation. In Section 6.3, we integrate mesoscale structures into the network model by incorporating correlations in an SBM-like model. At this point, we (temporarily) assume that each layer has the same block structure. We consider models without degree correction to start before analysing degree-corrected models. In Section 6.4, we remove the assumption that all layers have the same mesoscale structure. We summarise the results from all of the models that we have considered in Sections 6.2–6.4 in Table 6.1. In Section 6.5, we describe how to use these models for edge prediction, and we provide some results for synthetic networks. We then proceed with applications to consumer–product networks in Section 6.6. We find significant correlation (up to about 0.45) between different layers of a temporal shopping network, although some communities exhibit much larger correlations than others. We then apply our edge-prediction approach from Section 6.5 to this temporal network. We summarise our work in Section 6.7 and discuss a few ideas for future work.

[3]In fact, our work from Chapter 4 and previous algorithms for SBM inference [3, 34, 109, 145] suggest that a full inference algorithm can be cast as an iterative approach that alternates between estimating a multilayer network's mesoscale structure and estimating its edge probabilities and correlations. Thus, the latter task — which is the focus of this chapter — constitutes one half of an algorithm for correlated-community detection.

6.2 Correlated Erdős–Rényi Layers

Although our ultimate goal is to propose a correlated multilayer model in which each layer exhibits some form of mesoscale structure, we start from the simpler case in which the network in each layer resembles an ER graph from the $\mathcal{G}(n, p)$ ensemble [107]. In single-layer SBMs, it is common to use either Bernoulli or Poisson random variables to generate edges between nodes. The former is generally more accurate, because it does not yield multiedges, but the latter is used more commonly in practice, as it often simplifies calculations considerably [122, 171]. We consider both types of model in this section.

The derivations in this section and throughout the rest of this chapter consider just two network layers at a time. Although this may seem limiting, we can apply this framework to generate correlated networks with more than two layers in a sequential manner (see the discussion in Section 6.2.1), and we can determine pairwise layer correlations for a network with arbitrarily many layers (see the application to shopping networks in Section 6.6). Modelling all layers simultaneously rather than in a pairwise fashion is another topic for future work. We note that the derivations are nearly identical whether the underlying networks are bipartite or not. For the bipartite case, we explicitly assume that edges only connect nodes of different types (in our case, customers and products), so that the total number of node pairs that can form edges is $N_C N_P$ (rather than $N(N-1)/2$, where N is the total number of nodes).

Let $\boldsymbol{A}^1$ and $\boldsymbol{A}^2$ denote the adjacency matrices of our two network layers. As in classical SBMs, we assume that edges in these two layers are generated by some random process, so the entries A^1_{ij} and A^2_{ij} are random variables. Imposing some statistical correlation between these two sets of random variables then introduces cross-layer correlations in the resulting multilayer network structure. In Sections 6.2.1 and 6.2.2, we use bivariate Bernoulli and bivariate Poisson random variables, respectively, to model the entries A^1_{ij} and A^2_{ij}.

6.2.1 Bernoulli Model

The model that we present in this section is the same as the one proposed by Lyzinski et al. in [94], although the focus of that paper was on the graph-matching problem (see Section 6.1). Additionally, our derivation of the ML parameter estimates is novel, as is our formulation of this model in terms of a correlation parameter.

Assume that our two intra-layer networks are ER graphs with edge probabilities p_1 and p_2. (In the bipartite case, assume that p_1 and p_2 denote the edge probabilities

between nodes of different types, resulting in an ER-like ensemble of bipartite graphs.) For each pair of nodes (i, j), we have

$$\mathbb{P}(A^1_{ij} = 1) = p_1\,, \tag{6.1}$$
$$\mathbb{P}(A^2_{ij} = 1) = p_2\,. \tag{6.2}$$

To couple edges that connect the same pair of nodes in different layers, let

$$\mathbb{P}(A^1_{ij} = 1, A^2_{ij} = 1) = q \tag{6.3}$$

denote the joint probability for an edge to occur in both layers. Unless $q = p_1 p_2$, this construction means that the random variables A^1_{ij} and A^2_{ij} are not independent.

The parameters p_1, p_2, and q (which lie in the interval $[0, 1]$) fully specify our first forward model of networks with correlated ER layers. To generate a network from this model, one considers each pair of nodes (i, j) and, independently of all other node pairs, assigns values to A^1_{ij} and A^2_{ij} according to the following probabilities:

$$\begin{aligned}
&\mathbb{P}(A^1_{ij} = 1, A^2_{ij} = 1) = q\,,\\
&\mathbb{P}(A^1_{ij} = 1, A^2_{ij} = 0) = p_1 - q\,,\\
&\mathbb{P}(A^1_{ij} = 0, A^2_{ij} = 1) = p_2 - q\,,\\
&\mathbb{P}(A^1_{ij} = 0, A^2_{ij} = 0) = 1 - p_1 - p_2 + q\,.
\end{aligned} \tag{6.4}$$

Note that these expressions follow from the laws of probability and from the definitions of p_1, p_2, and q. For these probabilities to be well-defined, the necessary and sufficient conditions are $0 \leq q \leq \min(p_1, p_2)$ and $p_1 + p_2 \leq 1 + q$. Figure 6.2 illustrates the feasible region for p_1 and p_2 given a value for q.

It is also possible to generate a correlated ER network in a sequential manner. First, one generates the adjacency matrix $\boldsymbol{A}^1$ by placing edges with probability p_1. The probabilities of edges in the second layer are then determined by conditioning on the first layer:

$$\begin{aligned}
\mathbb{P}(A^2_{ij} = 1 | A^1_{ij} = 1) &= \frac{\mathbb{P}(A^1_{ij} = 1, A^2_{ij} = 1)}{\mathbb{P}(A^1_{ij} = 1)} = \frac{q}{p_1}\,,\\
\mathbb{P}(A^2_{ij} = 1 | A^1_{ij} = 0) &= \frac{\mathbb{P}(A^1_{ij} = 0, A^2_{ij} = 1)}{\mathbb{P}(A^1_{ij} = 0)} = \frac{p_2 - q}{1 - p_1}\,,\\
\mathbb{P}(A^2_{ij} = 0 | A^1_{ij} = 1) &= \frac{\mathbb{P}(A^1_{ij} = 1, A^2_{ij} = 0)}{\mathbb{P}(A^1_{ij} = 1)} = \frac{p_1 - q}{p_1}\,,\\
\mathbb{P}(A^2_{ij} = 0 | A^1_{ij} = 0) &= \frac{\mathbb{P}(A^1_{ij} = 0, A^2_{ij} = 0)}{\mathbb{P}(A^1_{ij} = 0)} = \frac{1 - p_1 - p_2 + q}{1 - p_1}\,.
\end{aligned} \tag{6.5}$$

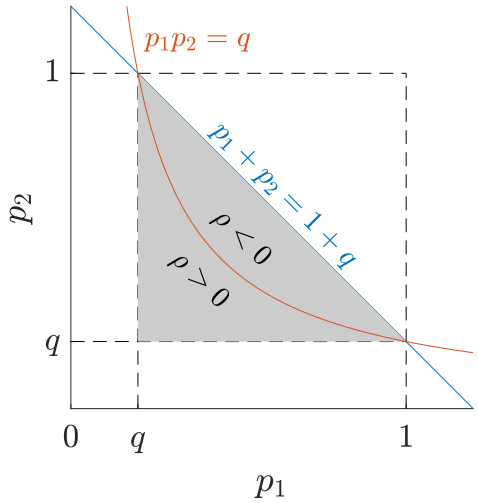


Figure 6.2: Visualisation of the feasible region (grey area) for p_1 and p_2 given a value of q. The boundaries of this region are defined by the inequalities $q \leq p_1 \leq 1$, $q \leq p_2 \leq 1$, and $p_1 + p_2 \leq 1 + q$. The parabola $p_1 p_2 = q$ specifies the boundary between regimes with positive layer correlation and regimes with negative layer correlation.

With this approach, it is possible to generate networks with arbitrarily many layers by first sampling edges in the first layer, and then sampling edges in each subsequent layer by conditioning on the previous one. This kind of process is especially well-suited for temporal networks, in which layers have a natural ordering. For multiplex networks, an extension of equations (6.4) to more than two layers is more appropriate.

It is also possible to parametrise this model in terms of the marginal Bernoulli probabilities p_1 and p_2 and the Pearson correlation

$$\rho = \frac{\mathbb{E}\left[A_{ij}^1 A_{ij}^2\right] - \mathbb{E}\left[A_{ij}^1\right]\mathbb{E}\left[A_{ij}^2\right]}{\sigma[A_{ij}^1]\sigma[A_{ij}^2]} = \frac{q - p_1 p_2}{\sqrt{p_1(1-p_1)p_2(1-p_2)}}\,, \tag{6.6}$$

where $\mathbb{E}[\cdot]$ and $\sigma[\cdot]$ give the mean and standard deviation of a random variable. One benefit of using ρ, rather than q, as the third model parameter is that its value is easier to interpret. A value of ρ close to 0 indicates weak correlation between layers, whereas values close to the extremes of $+1$ and -1 indicate strong positive correlation and strong negative correlation, respectively.

We can gain some further intuition by considering the cases $\rho = 0$, $\rho = 1$, and $\rho = -1$ in detail for our forward model. First, we have

$$\rho = 0 \quad \Leftrightarrow \quad q = p_1 p_2 \quad \Leftrightarrow \quad \mathbb{P}(A_{ij}^1 = 1, A_{ij}^2 = 1) = \mathbb{P}(A_{ij}^1 = 1)\mathbb{P}(A_{ij}^2 = 1)\,.$$

This says that the correlation is 0 if and only if edges are generated independently in the two layers, according to the marginal probabilities p_1 and p_2.

For the case $\rho = 1$, assume without loss of generality that $p_1 \leq p_2$. By construction, $q \leq \min(p_1, p_2)$. We then have

$$\rho = \frac{q - p_1 p_2}{\sqrt{p_1(1-p_1)p_2(1-p_2)}} \leq \frac{p_1 - p_1 p_2}{\sqrt{p_1(1-p_1)p_2(1-p_2)}} = \sqrt{\frac{p_1 - p_1 p_2}{p_2 - p_1 p_2}} \leq 1\,,$$

with equality if and only if $p_1 = p_2 = q$. In this case, the only nonzero probabilities in (6.4) are those that correspond to nodes (i, j) either being adjacent in both layers or adjacent in neither layer. Equivalently, this corresponds to the two layers exhibiting identical network structure, as we would expect for $\rho = 1$.

To analyse the case $\rho = -1$, assume that $q - p_1 p_2 \leq 0$. (Otherwise, equation (6.6) cannot yield negative correlation values.) Additionally, $1 - p_1 - p_2 + q \geq 0 \Leftrightarrow 1 - p_1 - p_2 \geq -q$, which must hold to ensure that the probability $\mathbb{P}(A^1_{ij} = 0, A^2_{ij} = 0)$ is well-defined. We have

$$\begin{aligned}\rho &= \frac{q - p_1 p_2}{\sqrt{p_1(1-p_1)p_2(1-p_2)}} = -\frac{(\sqrt{p_1 p_2 - q})^2}{\sqrt{p_1(1-p_1)p_2(1-p_2)}} \\ &= -\sqrt{\frac{p_1 p_2 - q}{p_1 p_2}}\sqrt{\frac{p_1 p_2 - q}{1 - p_1 - p_2 + p_1 p_2}} \geq -\sqrt{\frac{p_1 p_2 - q}{p_1 p_2}}\sqrt{\frac{p_1 p_2 - q}{p_1 p_2 - q}} \geq -1\,,\end{aligned}$$

with equality if and only if $q = 0$ and $p_1 = 1 - p_2$.

To summarise this set of calculations, we have: (1) $\rho = 0 \Leftrightarrow q = p_1 p_2$; (2) $\rho = 1 \Leftrightarrow q = p_1 = p_2$; and (3) $\rho = -1 \Leftrightarrow q = 0$ and $p_1 = 1 - p_2$.

Maximum-likelihood parameter estimates

We now derive ML estimates of the parameters p_1, p_2, and q. Let $\mathcal{E}$ denote the set of node pairs that can form edges. (We use this general notation to encompass both bipartite and non-bipartite settings.) We independently consider each pair of nodes $(i, j) \in \mathcal{E}$ when generating edges, so the likelihood of observing adjacency matrices $\boldsymbol{A}^1$ and $\boldsymbol{A}^2$ is

$$\begin{aligned}&\mathbb{P}(\boldsymbol{A}^1, \boldsymbol{A}^2 | p_1, p_2, q) \\ &\quad = \prod_{(i,j)\in\mathcal{E}} q^{A^1_{ij}A^2_{ij}}(p_1 - q)^{A^1_{ij}(1-A^2_{ij})}(p_2 - q)^{(1-A^1_{ij})A^2_{ij}}(1 - p_1 - p_2 + q)^{(1-A^2_{ij})(1-A^2_{ij})}\,.\end{aligned} \tag{6.7}$$

It is helpful to introduce the following notation:

$$\begin{aligned}e_{11} &:= |\{(i,j) \in \mathcal{E} : A^1_{ij} = 1, A^2_{ij} = 1\}|\,, \\ e_{10} &:= |\{(i,j) \in \mathcal{E} : A^1_{ij} = 1, A^2_{ij} = 0\}|\,, \\ e_{01} &:= |\{(i,j) \in \mathcal{E} : A^1_{ij} = 0, A^2_{ij} = 1\}|\,,\end{aligned}$$

$$e_{00} \;:=\; \left|\{(i,j) \in \mathcal{E} : A^1_{ij} = 0, A^2_{ij} = 0\}\right| .$$

These quantities correspond, respectively, to the number of node pairs that are adjacent in both layers, that are adjacent in the first layer but not the second, that are adjacent in the second layer but not the first, and that are not adjacent in either layer. Using this notation and taking the logarithm of (6.7), we arrive at the following expression for the log-likelihood:

$$\mathcal{L} = e_{11} \log q + e_{10} \log(p_1 - q) + e_{01} \log(p_2 - q) + e_{00} \log(1 - p_1 - p_2 + q) . \tag{6.8}$$

When fitting this model to network data, the quantities e_{11}, e_{10}, e_{01}, e_{00} are all known, and we are interested in determining the values of p_1, p_2, and q that are best explained by the data. To do so, we maximise[4] the likelihood (6.8) by setting its partial derivatives to 0:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial p_1} = 0 \quad &\Leftrightarrow \quad \frac{e_{10}}{p_1 - q} - \frac{e_{00}}{1 - p_1 - p_2 + q} = 0 , \\
\frac{\partial \mathcal{L}}{\partial p_2} = 0 \quad &\Leftrightarrow \quad \frac{e_{01}}{p_2 - q} - \frac{e_{00}}{1 - p_1 - p_2 + q} = 0 , \\
\frac{\partial \mathcal{L}}{\partial q} = 0 \quad &\Leftrightarrow \quad \frac{e_{11}}{q} - \frac{e_{10}}{p_1 - q} - \frac{e_{01}}{p_2 - q} + \frac{e_{00}}{1 - p_1 - p_2 + q} = 0 .
\end{aligned} \tag{6.9}$$

From these equations, we obtain

$$\widehat{p}_1 = \frac{e_{11} + e_{10}}{e_{11} + e_{10} + e_{01} + e_{00}} , \tag{6.10}$$

$$\widehat{p}_2 = \frac{e_{11} + e_{01}}{e_{11} + e_{10} + e_{01} + e_{00}} , \tag{6.11}$$

$$\widehat{q} = \frac{e_{11}}{e_{11} + e_{10} + e_{01} + e_{00}} . \tag{6.12}$$

Note that the denominator in all three expressions is equal to the number of potential edges (i.e., the cardinality of $\mathcal{E}$). This quantity is known for a given network, but its value depends on whether the network is bipartite, whether self-edges are allowed, etc. Additionally, let $m_1 = e_{11} + e_{10}$ and $m_2 = e_{11} + e_{01}$, respectively, denote the number of observed edges in the first and the second layers. It follows that the maximum-likelihood (ML) estimate $\widehat{p}_1$ is equal to the number of observed edges in layer 1 divided by the number of potential edges, and an analogous relation holds for $\widehat{p}_2$. The estimate $\widehat{q}$ is equal to the number of node pairs that are adjacent in both layers divided by the total number of node pairs. These results therefore match our intuitive expectations.

[4]The Hessian is negative definite at the critical point that one obtains by setting these derivatives to 0, implying that this point is indeed a local maximum (as opposed to a local minimum or saddle point) of the log-likelihood function.

We obtain an estimate of the Pearson correlation ρ between the two layers by substituting the ML estimates $\widehat{p}_1$, $\widehat{p}_2$, and $\widehat{q}$ into equation (6.6):

$$\widehat{\rho} = \frac{e_{00}e_{11} - e_{10}e_{01}}{\sqrt{(e_{11}+e_{10})(e_{11}+e_{01})(e_{10}+e_{00})(e_{01}+e_{00})}}\,. \tag{6.13}$$

It is not obvious a priori that the expression in (6.13) is an ML estimate of ρ. In other words, it is not clear that maximising the log-likelihood (6.8) with respect to p_1, p_2, and ρ (rather than p_1, p_2, and q), gives the same results. We confirmed that the results are indeed the same by substituting the expressions (6.10), (6.11), and (6.13) into the equations (6.9). Although the two formulations yield equivalent results, it is generally much easier to maximise the log-likelihood with respect to p_1, p_2, and q (or analogous parameters for more complicated models), so we use this parametrisation throughout the rest of the chapter.

We make a few comments about the expression of the correlation from (6.13). One can show the following: (1) $\widehat{\rho} = 0 \Leftrightarrow \widehat{q} = \widehat{p}_1\widehat{p}_2$; (2) $\widehat{\rho} = 1 \Leftrightarrow e_{10} = e_{01} = 0$; and (3) $\widehat{\rho} = -1 \Leftrightarrow e_{11} = e_{00} = 0$. These mirror similar results that we derived earlier in Section 6.2.1 for the forward model and its correlation parameter ρ. Note additionally that $\widehat{\rho}$ is not defined when either layer is an empty or a complete graph. In these cases, the denominator of (6.13) is 0, which is itself a consequence of the 0 values of the standard deviations of the Bernoulli random variables that govern edge existence.

6.2.2 Poisson Model

By definition, an ER random graph from the $\mathcal{G}(n,p)$ ensemble uses Bernoulli trials to generate edges. We consider a modification of this standard model that instead uses Poisson random variables.

As in the networks with correlated ER layers from Section 6.2.1, we assume that edges between different node pairs (i,j) are sampled independently from the same distribution. We now aim to model the random variables A^1_{ij} and A^2_{ij} using a bivariate Poisson distribution with some prescribed correlation ρ. The standard construction [62] is

$$\begin{aligned} A^1_{ij} &= Y^1_{ij} + Y^0_{ij}\,, \\ A^2_{ij} &= Y^2_{ij} + Y^0_{ij}\,, \end{aligned}$$

where Y^0_{ij}, Y^1_{ij}, and Y^2_{ij} are independent Poisson random variables with non-negative rate parameters μ_0, μ_1, and μ_2, respectively. It follows that $A^1_{ij} \sim \text{Pois}(\mu_1 + \mu_0)$ and

$A^2_{ij} \sim \text{Pois}(\mu_2 + \mu_0)$. Let $\lambda_1 = \mu_1 + \mu_0$ and $\lambda_2 = \mu_2 + \mu_0$. The Pearson correlation is

$$\rho = \frac{\mathbb{E}[A^1_{ij}A^2_{ij}] - E[A^1_{ij}]\mathbb{E}[A^2_{ij}]}{\sigma[A^1_{ij}]\sigma[A^2_{ij}]} = \frac{\mu_0}{\sqrt{(\mu_1+\mu_0)(\mu_2+\mu_0)}} = \frac{\mu_0}{\sqrt{\lambda_1\lambda_2}}\,. \tag{6.14}$$

We have

$$0 \leq \rho \leq \min\left\{\sqrt{\frac{\lambda_1}{\lambda_2}}, \sqrt{\frac{\lambda_2}{\lambda_1}}\right\},$$

with ρ equal to the upper bound if $\mu_1 = 0$ (which implies that $\rho = \sqrt{\lambda_1/\lambda_2}$) or $\mu_2 = 0$ (which implies that $\rho = \sqrt{\lambda_2/\lambda_1}$). In particular, $\rho = 1$ if and only if $\mu_0 = \lambda_1 = \lambda_2$, so A^1_{ij} and A^2_{ij} are modelled by the single random variable Y^0_{ij}. Note additionally that $\rho = 0$ if and only if $\mu_0 = 0$, so A^1_{ij} and A^2_{ij} are independent Poisson random variables. From [62], the joint probability distribution function is

$$\mathbb{P}(A^1_{ij} = k_1, A^2_{ij} = k_2) = e^{-(\mu_0+\mu_1+\mu_2)} \sum_{k=0}^{\min\{k_1,k_2\}} \frac{\mu_0^k \mu_1^{k_1-k} \mu_2^{k_2-k}}{k!(k_1-k)!(k_2-k)!}\,. \tag{6.15}$$

This construction cannot encompass negative correlation values. One way to handle such values is by letting

$$A^1_{ij} = Y^1_{ij} + Y^0_{ij}\,,$$
$$A^2_{ij} = Y^2_{ij} - Y^0_{ij}\,,$$

with $Y^0_{ij} \sim \text{Pois}(\mu_0)$, $Y^1_{ij} \sim \text{Pois}(\mu_1)$, and $Y^2_{ij} \sim \text{Pois}(\mu_2)$, as in the positive-correlation case. We then obtain $A^1_{ij} \sim \text{Pois}(\mu_1 + \mu_0)$, as before. However, A^2_{ij} is no longer a Poisson-distributed random variable, and it no longer has support on the non-negative integers. Instead, A^2_{ij} has a Skellam distribution [141] with parameters μ_2 and μ_0. Because this construction departs from standard network models, we only investigate the case of positive correlation values. In applications where negative correlations are likely to arise, we suggest using Bernoulli models.[5]

Maximum-likelihood parameter estimates

Equation (6.15) gives the probability of a single pair of observations (A^1_{ij}, A^2_{ij}). Given adjacency matrices $\boldsymbol{A}^1$ and $\boldsymbol{A}^2$, the likelihood is a product over all (i,j):

$$\mathbb{P}(\boldsymbol{A}^1, \boldsymbol{A}^2 | \mu_0, \mu_1, \mu_2) = \prod_{(i,j)\in\mathcal{E}} \left[e^{-(\mu_0+\mu_1+\mu_2)} \sum_{k=0}^{\min\{A^1_{ij}, A^2_{ij}\}} \frac{\mu_0^k \mu_1^{A^1_{ij}-k} \mu_2^{A^2_{ij}-k}}{k!(A^1_{ij}-k)!(A^2_{ij}-k)!} \right].$$

[5]There is another reason against using Poisson models for networks with strong negative correlations between layers. In such cases, one layer is necessarily much denser than the other, as it has $\mathcal{O}(N^2)$ edges. Poisson models are inaccurate for dense networks, because they generate multiedges with high probability. Bernoulli models remain accurate all the way to the limit $\rho \to -1$.

Taking the logarithm of both sides, the log-likelihood is

$$\mathcal{L} = -|\mathcal{E}|(\mu_0 + \mu_1 + \mu_2) + \sum_{(i,j)\in\mathcal{E}} \log \left(\sum_{k=0}^{\min\{A_{ij}^1, A_{ij}^2\}} \frac{\mu_0^k \mu_1^{A_{ij}^1 - k} \mu_2^{A_{ij}^2 - k}}{k!(A_{ij}^1 - k)!(A_{ij}^2 - k)!} \right), \tag{6.16}$$

where $|\mathcal{E}|$ denotes the cardinality of $\mathcal{E}$ and is equal to the number of potential edges.

Assume that we only apply this correlated Poisson model to unweighted networks without multiedges. (In the sparse-network regime that interests us, the expected number of multiedges in the forward model vanishes as the number of nodes tends to infinity.) The factorials that appear in equation (6.16) are then always equal to 1, and the sum inside the logarithm only has more than one term when $A_{ij}^1 = A_{ij}^2 = 1$. As in Section 6.2.1, let e_{11}, e_{10}, e_{01}, and e_{00} count the number of node pairs that are adjacent in both layers, adjacent in the first layer only, adjacent in the second layer only, and adjacent in neither layer, respectively. The log-likelihood then simplifies to

$$\mathcal{L} = -|\mathcal{E}|(\mu_0 + \mu_1 + \mu_2) + e_{11} \log(\mu_0 + \mu_1\mu_2) + e_{10} \log \mu_1 + e_{01} \log \mu_2 \,. \tag{6.17}$$

Fitting this model to data amounts to finding the values of μ_0, μ_1, and μ_2 that maximise $\mathcal{L}$. As before, we determine these values by setting the corresponding partial derivatives of $\mathcal{L}$ to 0:

$$\frac{\partial \mathcal{L}}{\partial \mu_0} = 0 \quad \Leftrightarrow \quad -|\mathcal{E}| + \frac{e_{11}}{\mu_0 + \mu_1\mu_2} = 0 \,, \tag{6.18}$$

$$\frac{\partial \mathcal{L}}{\partial \mu_1} = 0 \quad \Leftrightarrow \quad -|\mathcal{E}| + \frac{e_{11}\mu_2}{\mu_0 + \mu_1\mu_2} + \frac{e_{10}}{\mu_1} = 0 \,, \tag{6.19}$$

$$\frac{\partial \mathcal{L}}{\partial \mu_2} = 0 \quad \Leftrightarrow \quad -|\mathcal{E}| + \frac{e_{11}\mu_1}{\mu_0 + \mu_1\mu_2} + \frac{e_{01}}{\mu_2} = 0 \,. \tag{6.20}$$

From equation (6.18), we have

$$\frac{e_{11}}{\mu_0 + \mu_1\mu_2} = |\mathcal{E}| \,.$$

Plugging this expression into equation (6.19), we find

$$-|\mathcal{E}|(1 - \mu_2) + \frac{e_{10}}{\mu_1} = 0 \quad \Rightarrow \quad \mu_1 = \frac{e_{10}}{|\mathcal{E}|(1 - \mu_2)} \,,$$

which we substitute in (6.20) to get an equation for μ_2:

$$\begin{aligned} -|\mathcal{E}| \left[1 - \frac{e_{10}}{|\mathcal{E}|(1 - \mu_2)} \right] + \frac{e_{01}}{\mu_2} &= 0 \\ \Leftrightarrow \quad |\mathcal{E}|\mu_2^2 - (|\mathcal{E}| - e_{10} + e_{01})\,\mu_2 + e_{01} &= 0 \,. \end{aligned}$$

The solutions of the resulting quadratic equation are

$$\mu_2 = \frac{(|\mathcal{E}| - e_{10} + e_{01}) \pm \sqrt{(|\mathcal{E}| - e_{10} + e_{01})^2 - 4|\mathcal{E}|e_{01}}}{2|\mathcal{E}|} \,.$$

We can simplify the expression inside the square root by using $e_{11}+e_{10}+e_{01}+e_{00} = |\mathcal{E}|$, so that

$$\begin{aligned}(|\mathcal{E}| - e_{10} + e_{01})^2 - 4|\mathcal{E}|e_{01} &= (e_{11} + e_{00} + 2e_{01})^2 - 4(e_{11} + e_{10} + e_{01} + e_{00})e_{01} \\ &= (e_{11} + e_{00})^2 - 4e_{10}e_{01} \,.\end{aligned}$$

Therefore,

$$\widehat{\mu}_2 = \frac{e_{11} + e_{00} + 2e_{01} \pm \sqrt{(e_{11} + e_{00})^2 - 4e_{10}e_{01}}}{2|\mathcal{E}|} \,. \tag{6.21}$$

By symmetry (or by repeating the analogous calculations), we get

$$\widehat{\mu}_1 = \frac{e_{11} + e_{00} + 2e_{10} \pm \sqrt{(e_{11} + e_{00})^2 - 4e_{10}e_{01}}}{2|\mathcal{E}|} \,. \tag{6.22}$$

From equation (6.18), we obtain

$$\mu_0 = \frac{e_{11}}{|\mathcal{E}|} - \mu_1\mu_2 \,,$$

which ultimately gives

$$\widehat{\mu}_0 = \frac{e_{11} - e_{00} \mp \sqrt{(e_{11} + e_{00})^2 - 4e_{10}e_{01}}}{2|\mathcal{E}|} \,. \tag{6.23}$$

These calculations have yielded two possible solutions for (μ_0, μ_1, μ_2). Investigating the Hessian of $\mathcal{L}$ at these two critical points, we find that one of them is indeed a maximum of $\mathcal{L}$, whereas the other is a saddle point. The solution that we seek is

$$\begin{aligned}\widehat{\mu}_0 &= \frac{e_{11} - e_{00} + \sqrt{(e_{11} + e_{00})^2 - 4e_{10}e_{01}}}{2|\mathcal{E}|} \,, \\ \widehat{\mu}_1 &= \frac{e_{11} + e_{00} + 2e_{10} - \sqrt{(e_{11} + e_{00})^2 - 4e_{10}e_{01}}}{2|\mathcal{E}|} \,, \\ \widehat{\mu}_2 &= \frac{e_{11} + e_{00} + 2e_{01} - \sqrt{(e_{11} + e_{00})^2 - 4e_{10}e_{01}}}{2|\mathcal{E}|} \,.\end{aligned} \tag{6.24}$$

The parameters of the Poisson random variables that govern edge placement in the two layers are

$$\widehat{\lambda}_1 = \widehat{\mu}_0 + \widehat{\mu}_1 = \frac{e_{11} + e_{10}}{|\mathcal{E}|} = \frac{m_1}{|\mathcal{E}|} \,,$$

$$\widehat{\lambda}_2 = \widehat{\mu}_0 + \widehat{\mu}_2 = \frac{e_{11} + e_{01}}{|\mathcal{E}|} = \frac{m_2}{|\mathcal{E}|}\,. \tag{6.25}$$

Substituting the ML estimates (6.24) into the formula (6.14) for the Pearson correlation, we get

$$\widehat{\rho} = \frac{\widehat{\mu}_0}{\sqrt{\widehat{\lambda}_1 \widehat{\lambda}_2}} = \frac{e_{11} - e_{00} + \sqrt{(e_{11} + e_{00})^2 - 4e_{10}e_{01}}}{2\sqrt{(e_{11} + e_{10})(e_{11} + e_{01})}}\,. \tag{6.26}$$

We have verified that this is indeed an ML estimate of the correlation, as it also maximises a version of the log-likelihood (6.17) that is parametrised in terms of λ_1, λ_2, and ρ (rather than μ_0, μ_1, and μ_2). Similarly to the Bernoulli case, $\widehat{\rho} = 0$ if and only if $e_{11} = |\mathcal{E}|\widehat{\lambda}_1\widehat{\lambda}_2$, and $\rho = 1$ if and only if $e_{10} = e_{01} = 0$. (Recall that negative correlation values are not possible for the bivariate Poisson model that we examine in this section.)

Formulas (6.24) and (6.26) make sense only when the expression under the square root is non-negative. When both layers are sparse, which is the only case when these Poisson models are appropriate (see Section 2.1), the quantity e_{00} is $\mathcal{O}(N^2)$, whereas e_{01}, e_{10}, and e_{11} are at most $\mathcal{O}(N)$. In this case, $(e_{11} + e_{00})^2 \gg 4e_{10}e_{01}$, as desired. We also note that $\widehat{\mu}_1 \geq 0$ and $\widehat{\mu}_2 \geq 0$. The condition $\widehat{\mu}_0 \geq 0$ holds if and only if $e_{11}e_{00} \geq e_{10}e_{01}$; in fact, this is also the condition for $\widehat{\rho} \geq 0$. When this inequality does not hold, the estimated correlation between layers is negative. One should then use the Bernoulli model from Section 6.2.1, because the bivariate Poisson model from this section cannot handle negative correlation values. In regimes in which both models are valid, we find good agreement between them (see Section 6.6).

6.3 Correlated SBMs With Fixed Blocks

To make the approach in Section 6.2 better suited for applications, we extend it to networks that have some form of mesoscale structure. In this section, we assume that layers have identical block structure, but we relax this assumption in Section 6.4.

Similarly to the case of ER layers, we aim to derive a multilayer network model in which the marginal of each layer is a standard SBM. As in Section 6.2, we generate edges that connect copies of the same nodes in different layers in a correlated manner. It is well-known that it is necessary to introduce degree correction for SBMs to give good results on real networks [76]. However, a degree-corrected SBM is significantly more complicated to analyse, especially when dealing with multilayer networks. To gain some intuition, we first consider models without degree correction.

6.3.1 Bernoulli Model Without Degree Correction

Let $\boldsymbol{g}$ be a vector of block assignments, which we take to be identical for all layers of a given network. As explained in Section 6.1, we assume throughout this chapter that $\boldsymbol{g}$ is given, and we aim to estimate the remaining model parameters. Using terminology from Chapter 2, let $\mathcal{B}$ be the set of edge bundles (r, s), each of which is described by its own set of parameters p^1_{rs}, p^2_{rs}, and q_{rs}; these play an analogous role to p_1, p_2, and q for ER layers. The matrices $\boldsymbol{p}^1$, $\boldsymbol{p}^2$, and $\boldsymbol{q}$ have size $K \times K$ for unipartite networks, where K is the number of blocks; and they have size $K_C \times K_P$ for bipartite networks, where K_C and K_P are the number of blocks for each of the two node types (e.g., customers and products for shopping networks). We then have $\mathcal{B} = \{1, \dots, K\} \times \{1, \dots, K\}$ in the former case and $\mathcal{B} = \{1, \dots, K_C\} \times \{1, \dots, K_P\}$ in the latter. With this notation, the derivations that follow apply to both unipartite and bipartite networks.

Letting g_i denote the block assignment of node i, a correlated two-layer SBM is described by the following set of equalities:

$$\begin{aligned} \mathbb{P}(A^1_{ij} = 1) &= p^1_{g_i g_j}\,, \\ \mathbb{P}(A^2_{ij} = 1) &= p^2_{g_i g_j}\,, \\ \mathbb{P}(A^1_{ij} = 1, A^2_{ij} = 1) &= q_{g_i g_j}\,. \end{aligned}$$

This is the same model that was proposed by Lyzinski and collaborators in [96], although they used it to study the graph-matching problem. They did not perform maximum-likelihood inference, which is our focus. Our estimation of correlation parameters for each edge bundle, as well as for the network as a whole, is also novel.

Maximum-likelihood parameter estimates

As in Section 6.2, suppose that we consider each node pair (i, j) independently. The likelihood of observing adjacency matrices $\boldsymbol{A}^1$ and $\boldsymbol{A}^2$ is then

$$\begin{aligned} \mathbb{P}(\boldsymbol{A}^1, \boldsymbol{A}^2 | \boldsymbol{g}, \boldsymbol{p}^1, \boldsymbol{p}^2, \boldsymbol{q}) = \prod_{(i,j)\in\mathcal{E}} \Big[& q_{g_i g_j}^{A^1_{ij} A^2_{ij}} (p^1_{g_i g_j} - q_{g_i g_j})^{A^1_{ij}(1-A^2_{ij})} (p^2_{g_i g_j} - q_{g_i g_j})^{(1-A^1_{ij})A^2_{ij}} \\ & \times (1 - p^1_{g_i g_j} - p^2_{g_i g_j} + q_{g_i g_j})^{(1-A^2_{ij})(1-A^2_{ij})} \Big]\,. \end{aligned}$$

In this product, each factor depends on i and j only via their block memberships g_i and g_j, so we are able to combine terms accordingly. First, define

$$e^{ab}_{rs} := \big|\{(i, j) \in \mathcal{E} : A^1_{ij} = a, A^2_{ij} = b, g_i = r, g_j = s\}\big|$$

for $(a, b) \in \{(1,1),(1,0),(0,1),(0,0)\}$, in analogy with e_{11}, e_{10}, e_{01}, and e_{00} from Section 6.2.1. The log-likelihood can then be written as

$$\mathcal{L} = \sum_{(r,s)\in\mathcal{B}} \Big[e_{rs}^{11} \log q_{rs} + e_{rs}^{10} \log(p_{rs}^1 - q_{rs}) + e_{rs}^{01} \log(p_{rs}^2 - q_{rs}) + e_{rs}^{00} \log(1 - p_{rs}^1 - p_{rs}^2 + q_{rs}) \Big] . \tag{6.27}$$

The advantage of writing the log-likelihood as in (6.27) is that it clearly separates the contribution from different edge bundles. When maximising (6.27) with respect to the parameters p_{rs}^1, p_{rs}^2, and q_{rs}, we obtain a system of three equations for each edge bundle $(r, s) \in \mathcal{B}$:

$$\frac{\partial \mathcal{L}}{\partial p_{rs}^1} = 0 \quad \Leftrightarrow \quad \frac{e_{rs}^{10}}{p_{rs}^1 - q_{rs}} - \frac{e_{rs}^{00}}{1 - p_{rs}^1 - p_{rs}^2 + q_{rs}} = 0 , \tag{6.28}$$

$$\frac{\partial \mathcal{L}}{\partial p_{rs}^2} = 0 \quad \Leftrightarrow \quad \frac{e_{rs}^{01}}{p_{rs}^2 - q_{rs}} - \frac{e_{rs}^{00}}{1 - p_{rs}^1 - p_{rs}^2 + q_{rs}} = 0 , \tag{6.29}$$

$$\frac{\partial \mathcal{L}}{\partial q_{rs}} = 0 \quad \Leftrightarrow \quad \frac{e_{rs}^{11}}{q_{rs}} - \frac{e_{rs}^{10}}{p_{rs}^1 - q_{rs}} - \frac{e_{rs}^{01}}{p_{rs}^2 - q_{rs}} + \frac{e_{rs}^{00}}{1 - p_{rs}^1 - p_{rs}^2 + q_{rs}} = 0 . \tag{6.30}$$

One can solve these equations analogously to the ER case from Section 6.2.1 to obtain the following ML estimates of the parameters:

$$\widehat{p}_{rs}^1 = \frac{e_{rs}^{11} + e_{rs}^{10}}{e_{rs}^{11} + e_{rs}^{10} + e_{rs}^{01} + e_{rs}^{00}} = \frac{m_{rs}^1}{e_{rs}} , \tag{6.31}$$

$$\widehat{p}_{rs}^2 = \frac{e_{rs}^{11} + e_{rs}^{01}}{e_{rs}^{11} + e_{rs}^{10} + e_{rs}^{01} + e_{rs}^{00}} = \frac{m_{rs}^2}{e_{rs}} , \tag{6.32}$$

$$\widehat{q}_{rs} = \frac{e_{rs}^{11}}{e_{rs}^{11} + e_{rs}^{10} + e_{rs}^{01} + e_{rs}^{00}} = \frac{e_{rs}^{11}}{e_{rs}} , \tag{6.33}$$

where m_{rs}^1 and m_{rs}^2 denote the number of edges in layers 1 and 2, respectively, and e_{rs} is the number of possible edges between nodes in block r and block s. When there is a single edge bundle (i.e., when there is no assumed block structure in the network), the ML estimates (6.31)–(6.33) recover those obtained for correlated ER graphs in Section 6.2.1. Each edge bundle also has a corresponding Pearson correlation, whose ML estimate is

$$\widehat{\rho}_{rs} = \frac{e_{rs}^{00} e_{rs}^{11} - e_{rs}^{10} e_{rs}^{01}}{\sqrt{(e_{rs}^{11} + e_{rs}^{10})(e_{rs}^{11} + e_{rs}^{01})(e_{rs}^{10} + e_{rs}^{00})(e_{rs}^{01} + e_{rs}^{00})}} . \tag{6.34}$$

In our applications to temporal consumer–product networks, we find that different edge bundles have vastly different correlation values (see Section 6.6). We anticipate that other empirical multilayer networks have similar properties.

Effective correlation

Although having different correlation values for different edge bundles can be useful, it is also helpful to have a single correlation measurement for a given network. For example, one may wish to use such a network diagnostic for one of the purposes outlined in Section 6.1. One way to define an "effective correlation" is to first sample two node indices I and J uniformly at random, and then compute the Pearson correlation of the random variables A^1_{IJ} and A^2_{IJ}:

$$\operatorname{corr}(A^1_{IJ}, A^2_{IJ}) = \frac{\mathbb{E}[A^1_{IJ}A^2_{IJ}] - E[A^1_{IJ}]\mathbb{E}[A^2_{IJ}]}{\sigma[A^1_{IJ}]\sigma[A^2_{IJ}]}\,. \tag{6.35}$$

Note that we use capital letters for the node indices I and J to emphasise that they are random variables.

We can calculate each term on the right-hand side of (6.35) by conditioning on the block assignments of the randomly chosen nodes I and J. First, for $l \in \{1, 2\}$,

$$\begin{aligned}\mathbb{E}[A^l_{IJ}] = \mathbb{P}(A^l_{IJ} = 1) &= \sum_{(r,s)\in\mathcal{B}} \mathbb{P}(A_{IJ} = 1 | g_I = r, g_J = s)\mathbb{P}(g_I = r, g_J = s) \\ &= \sum_{(r,s)\in\mathcal{B}} \widehat{p}^1_{rs}\frac{e_{rs}}{|\mathcal{E}|} = \sum_{(r,s)\in\mathcal{B}} \frac{m^l_{rs}}{e_{rs}}\frac{e_{rs}}{|\mathcal{E}|} = \frac{m_l}{|\mathcal{E}|}\,,\end{aligned}$$

where m_l denotes the number of edges in layer l. This expression is the same as the probability p_l of generating an edge in layer l in the ER case. Because A^l_{IJ} is a Bernoulli random variable (in other words, it can only take the values 1 or 0), its standard deviation is

$$\sigma[A^l_{IJ}] = \sqrt{\frac{m_l}{|\mathcal{E}|}\left(1 - \frac{m_l}{|\mathcal{E}|}\right)}\,.$$

Lastly, we have

$$\begin{aligned}\mathbb{E}[A^1_{IJ}A^2_{IJ}] &= \mathbb{P}(A^1_{IJ} = 1, A^2_{IJ} = 1) \\ &= \sum_{(r,s)\in\mathcal{B}} \mathbb{P}(A^1_{IJ} = 1, A^2_{IJ} = 1 | g_I = r, g_J = s)\mathbb{P}(g_I = r, g_J = s) \\ &= \sum_{(r,s)\in\mathcal{B}} \widehat{q}_{rs}\frac{e_{rs}}{|\mathcal{E}|} = \sum_{(r,s)\in\mathcal{B}} \frac{e^{11}_{rs}}{e_{rs}}\frac{e_{rs}}{|\mathcal{E}|} = \frac{e_{11}}{|\mathcal{E}|}\,.\end{aligned}$$

Thus, the estimated value of the effective correlation is

$$\widehat{\rho} = \operatorname{corr}(A^1_{IJ}, A^2_{IJ}) = \frac{e_{00}e_{11} - e_{10}e_{01}}{\sqrt{(e_{11} + e_{10})(e_{11} + e_{01})(e_{10} + e_{00})(e_{01} + e_{00})}}\,, \tag{6.36}$$

which recovers the value in (6.13) for ER layers without any block structure. There is no reason a priori to expect this outcome; in fact, as we show in Section 6.3.2,

the analogous result does not hold for Poisson models. The fact that there is such a correspondence between models is convenient for practical reasons, as it implies that one can perform the simpler calculations from Section 6.2.1 to obtain correlation values, even for networks with nontrivial mesoscale structure.

6.3.2 Poisson Model Without Degree Correction

When edges are placed using Poisson random variables, the relevant edge-bundle parameters are μ^0_{rs}, μ^1_{rs}, and μ^2_{rs} for edge bundles $(r,s) \in \mathcal{B}$. For any two nodes i and j, the random variables A^1_{ij} and A^2_{ij} for the number of edges in the respective layers are

$$A^1_{ij} = Y^0_{ij} + Y^1_{ij}\,, \tag{6.37}$$

$$A^2_{ij} = Y^0_{ij} + Y^2_{ij}\,, \tag{6.38}$$

where the Y^l_{ij} are independent Poisson random variables with parameters $\mu^l_{g_ig_j}$ (for $l \in \{0,1,2\}$). The variables A^1_{ij} and A^2_{ij} are themselves Poisson-distributed, with means $\lambda^1_{g_ig_j} = \mu^0_{g_ig_j} + \mu^1_{g_ig_j}$ and $\lambda^2_{g_ig_j} = \mu^0_{g_ig_j} + \mu^2_{g_ig_j}$.

Maximum-likelihood parameter estimates

Similarly to Section 6.2.2, the likelihood is

$$\begin{aligned}&\mathbb{P}(\boldsymbol{A}^1, \boldsymbol{A}^2 | \boldsymbol{g}, \boldsymbol{\mu}^0, \boldsymbol{\mu}^1, \boldsymbol{\mu}^2) \\ &\quad = \prod_{(i,j)\in\mathcal{E}} e^{-(\mu^0_{g_ig_j}+\mu^1_{g_ig_j}+\mu^2_{g_ig_j})} \sum_{k=0}^{\min\{A^1_{ij},A^2_{ij}\}} \frac{(\mu^0_{g_ig_j})^k (\mu^1_{g_ig_j})^{A^1_{ij}-k} (\mu^2_{g_ig_j})^{A^2_{ij}-k}}{k!(A^1_{ij}-k)!(A^2_{ij}-k)!}\,.\end{aligned}$$

We can group terms that correspond to the same edge bundle (r,s). The log-likelihood becomes

$$\mathcal{L} = \sum_{(r,s)\in\mathcal{B}} \left[-e_{rs}(\mu^0_{rs}+\mu^1_{rs}+\mu^2_{rs}) + e^{11}_{rs}\log(\mu^0_{rs}+\mu^1_{rs}\mu^2_{rs}) + e^{10}_{rs}\log\mu^1_{rs} + e^{01}_{rs}\log\mu^2_{rs}\right], \tag{6.39}$$

where e^{11}_{rs}, e^{10}_{rs}, e^{01}_{rs}, and e_{rs} are defined as in Section 6.3.1. Finding the parameter values that maximise each term of (6.39) is analogous to the calculation for ER layers from Section 6.2.2. The resulting ML estimates are

$$\widehat{\mu}^0_{rs} = \frac{e^{11}_{rs} - e^{00}_{rs} + \sqrt{(e^{11}_{rs}+e^{00}_{rs})^2 - 4e^{10}_{rs}e^{01}_{rs}}}{2e_{rs}}\,, \tag{6.40}$$

$$\widehat{\mu}^1_{rs} = \frac{e^{11}_{rs} + e^{00}_{rs} + 2e^{10}_{rs} - \sqrt{(e^{11}_{rs}+e^{00}_{rs})^2 - 4e^{10}_{rs}e^{01}_{rs}}}{2e_{rs}}\,, \tag{6.41}$$

$$\widehat{\mu}_{rs}^2 = \frac{e_{rs}^{11} + e_{rs}^{00} + 2e_{rs}^{01} - \sqrt{(e_{rs}^{11} + e_{rs}^{00})^2 - 4e_{rs}^{10}e_{rs}^{01}}}{2e_{rs}} \,. \tag{6.42}$$

The Poisson parameters λ_{rs}^1 and λ_{rs}^2, which give the marginal edge probabilities in each of the two layers, are

$$\widehat{\lambda}_{rs}^1 = \frac{m_{rs}^1}{e_{rs}} \,, \qquad \widehat{\lambda}_{rs}^2 = \frac{m_{rs}^2}{e_{rs}} \,.$$

Note that these match the ML parameter estimates of single-layer Poisson SBMs with no degree correction. Lastly, each edge bundle (r, s) has an associated Pearson correlation ρ_{rs}, whose ML estimate is

$$\widehat{\rho}_{rs} = \frac{e_{rs}^{11} - e_{rs}^{00} + \sqrt{(e_{rs}^{11} + e_{rs}^{00})^2 - 4e_{rs}^{10}e_{rs}^{01}}}{\sqrt{m_{rs}^1 m_{rs}^2}} \,. \tag{6.43}$$

Effective correlation

As in the Bernoulli case, we seek to estimate the effective correlation

$$\widehat{\rho} = \mathrm{corr}(A_{IJ}^1, A_{IJ}^2) = \frac{\mathbb{E}[A_{IJ}^1 A_{IJ}^2] - \mathbb{E}[A_{IJ}^1]\mathbb{E}[A_{IJ}^2]}{\sigma[A_{IJ}^1]\sigma[A_{IJ}^2]} \,, \tag{6.44}$$

with nodes I and J chosen uniformly at random.

For $l \in \{1, 2\}$, we have

$$\begin{aligned}
\mathbb{E}[A_{IJ}^l] &= \sum_{k=0}^{\infty} k\mathbb{P}(A_{IJ}^l = k) = \sum_{k=0}^{\infty} k \sum_{(r,s)\in\mathcal{B}} \mathbb{P}(A_{IJ}^l = k | g_I = r, g_J = s)\mathbb{P}(g_I = r, g_J = s) \\
&= \sum_{k=0}^{\infty} k \sum_{(r,s)\in\mathcal{B}} \frac{(\widehat{\lambda}_{rs}^l)^k e^{-\widehat{\lambda}_{rs}^l}}{k!} \frac{e_{rs}}{|\mathcal{E}|} = \sum_{k=0}^{\infty} k \sum_{(r,s)\in\mathcal{B}} \frac{(m_{rs}^l/e_{rs})^k e^{-m_{rs}^l/e_{rs}}}{k!} \frac{e_{rs}}{|\mathcal{E}|} \\
&= \sum_{(r,s)\in\mathcal{B}} \frac{e_{rs}}{|\mathcal{E}|} e^{-m_{rs}^l/e_{rs}} \sum_{k=0}^{\infty} \frac{k(m_{rs}^l/e_{rs})^k}{k!} = \sum_{(r,s)\in\mathcal{B}} \frac{e_{rs}}{|\mathcal{E}|} e^{-m_{rs}^l/e_{rs}} \sum_{k=1}^{\infty} \frac{(m_{rs}^l/e_{rs})^k}{(k-1)!} \\
&= \sum_{(r,s)\in\mathcal{B}} \frac{e_{rs}}{|\mathcal{E}|} e^{-m_{rs}^l/e_{rs}} \frac{m_{rs}^l}{e_{rs}} e^{m_{rs}^l/e_{rs}} = \sum_{(r,s)\in\mathcal{B}} \frac{m_{rs}^l}{|\mathcal{E}|} = \frac{m_l}{|\mathcal{E}|} \,,
\end{aligned} \tag{6.45}$$

which is equal to the expected value of A_{ij}^l in the ER case. (Recall that m_l is the number of edges in layer l.)

To compute the standard deviation $\sigma[A_{IJ}^l]$, we first calculate $\mathbb{E}[(A_{IJ}^l)^2]$. This turns out to be more complicated:

$$\mathbb{E}\left[(A_{IJ}^l)^2\right] = \sum_{k=0}^{\infty} k^2 \mathbb{P}(A_{IJ}^l = k)$$

$$
\begin{aligned}
&= \sum_{k=0}^{\infty} k^2 \sum_{(r,s)\in\mathcal{B}} \mathbb{P}(A_{IJ}^l = k | g_I = r, g_J = s)\mathbb{P}(g_I = r, g_J = s) \\
&= \sum_{k=0}^{\infty} k^2 \sum_{(r,s)\in\mathcal{B}} \frac{(\widehat{\lambda}_{rs}^l)^k e^{-\widehat{\lambda}_{rs}^l}}{k!} \frac{e_{rs}}{|\mathcal{E}|} = \sum_{(r,s)\in\mathcal{B}} \frac{e_{rs}}{|\mathcal{E}|} e^{-\widehat{\lambda}_{rs}^l} \sum_{k=0}^{\infty} \frac{k^2 (\widehat{\lambda}_{rs}^l)^k}{k!} \\
&= \sum_{(r,s)\in\mathcal{B}} \frac{e_{rs}}{|\mathcal{E}|} e^{-\widehat{\lambda}_{rs}^l} \widehat{\lambda}_{rs}^l (1 + \widehat{\lambda}_{rs}^l) e^{\widehat{\lambda}_{rs}^l} = \sum_{(r,s)\in\mathcal{B}} \frac{e_{rs}}{|\mathcal{E}|} \frac{m_{rs}^l}{e_{rs}} \left(1 + \frac{m_{rs}^l}{e_{rs}}\right) \\
&= \frac{1}{|\mathcal{E}|} \sum_{(r,s)\in\mathcal{B}} m_{rs}^l + \frac{1}{|\mathcal{E}|} \sum_{(r,s)\in\mathcal{B}} \frac{(m_{rs}^l)^2}{e_{rs}} \\
&= \frac{m_l}{|\mathcal{E}|} + \frac{1}{|\mathcal{E}|} \sum_{(r,s)\in\mathcal{B}} \frac{(m_{rs}^l)^2}{e_{rs}} \,.
\end{aligned}
$$

We then obtain an expression for the variance:

$$
\mathrm{Var}\left[A_{IJ}^l\right] = \mathbb{E}\left[(A_{IJ}^l)^2\right] - \left(\mathbb{E}\left[A_{IJ}^l\right]\right)^2 = \frac{m_l}{|\mathcal{E}|} - \left(\frac{m_l}{|\mathcal{E}|}\right)^2 + \frac{1}{|\mathcal{E}|} \sum_{(r,s)\in\mathcal{B}} \frac{(m_{rs}^l)^2}{e_{rs}} \,. \tag{6.46}
$$

In particular, this shows that A_{IJ}^l does not follow a Poisson distribution when I and J are chosen uniformly at random. (If it did, then the variance would be equal to the mean $m_l/|\mathcal{E}|$.)

Finally, we need to compute $\mathbb{E}[A_{IJ}^1 A_{IJ}^2]$. To do this, consider first two nodes i and j, which are in groups r and s, respectively. We write $A_{ij}^1 = Y_{ij}^0 + Y_{ij}^1$ and $A_{ij}^2 = Y_{ij}^0 + Y_{ij}^2$, as in equations (6.37)–(6.38), where the Y_{ij}^l are independent Poisson random variables with rate parameters μ_{rs}^l. We calculate

$$
\begin{aligned}
\mathbb{E}[A_{ij}^1 A_{ij}^2] &= \mathbb{E}[(Y_{ij}^0 + Y_{ij}^1)(Y_{ij}^0 + Y_{ij}^2)] \\
&= \mathbb{E}[(Y_{ij}^0)^2] + \mathbb{E}[Y_{ij}^0]\left(\mathbb{E}[Y_{ij}^1] + \mathbb{E}[Y_{ij}^2]\right) + \mathbb{E}[Y_{ij}^1]\mathbb{E}[Y_{ij}^2] \\
&= \mu_{rs}^0 + (\mu_{rs}^0)^2 + \mu_{rs}^0(\mu_{rs}^1 + \mu_{rs}^2) + \mu_{rs}^1\mu_{rs}^2 \\
&= \mu_{rs}^0 + (\mu_{rs}^0)^2 + \mu_{rs}^0(\lambda_{rs}^0 + \lambda_{rs}^1 - 2\mu_{rs}^0) + (\lambda_{rs}^1 - \mu_{rs}^0)(\lambda_{rs}^2 - \mu_{rs}^0) \\
&= \mu_{rs}^0 + \lambda_{rs}^1\lambda_{rs}^2 \,.
\end{aligned}
$$

By the law of total expectation,

$$
\begin{aligned}
\mathbb{E}[A_{IJ}^1 A_{IJ}^2] &= \sum_{(r,s)\in\mathcal{B}} \mathbb{E}[A_{IJ}^1 A_{IJ}^2 | g_I = r, g_J = s]\mathbb{P}(g_I = r, g_J = s) \\
&= \sum_{(r,s)\in\mathcal{B}} \frac{e_{rs}}{|\mathcal{E}|}(\widehat{\mu}_{rs}^0 + \widehat{\lambda}_{rs}^1\widehat{\lambda}_{rs}^2) = \sum_{(r,s)\in\mathcal{B}} \frac{e_{rs}}{|\mathcal{E}|}\left(\widehat{\mu}_{rs}^0 + \frac{m_{rs}^1}{e_{rs}}\frac{m_{rs}^2}{e_{rs}}\right) \\
&= \frac{1}{|\mathcal{E}|} \sum_{(r,s)\in\mathcal{B}} e_{rs}\widehat{\mu}_{rs}^0 + \frac{1}{|\mathcal{E}|} \sum_{(r,s)\in\mathcal{B}} \frac{m_{rs}^1 m_{rs}^2}{e_{rs}} \,.
\end{aligned} \tag{6.47}
$$

The expression for $\widehat{\mu}^0_{rs}$ comes from equation (6.40):

$$\widehat{\mu}^0_{rs} = \frac{e^{11}_{rs} - e^{00}_{rs} + \sqrt{(e^{11}_{rs} + e^{00}_{rs})^2 - 4e^{10}_{rs}e^{01}_{rs}}}{e_{rs}} \,. \tag{6.48}$$

Equations (6.45)–(6.48), when plugged into (6.44), give an aggregate measure of correlation for the network. Unlike for Bernoulli layers, the expression that we obtain for this correlation is not the same as for a network with Poisson ER layers. This is another reason for using Bernoulli (rather than Poisson) random variables to model correlated networks.

6.3.3 Bernoulli Model With Degree Correction

The models that we have discussed thus far generate networks where nodes in the same block have the same expected degree. SBMs that make this kind of assumption tend to perform poorly when used to infer mesoscale structure in real networks, many of which have highly heterogeneous degree distributions. This observation has led to the development of degree-corrected SBMs for community detection [76]. We expect that such adjustments can also make a difference when modelling edge correlations, so we now extend the model from Section 6.3.1 to incorporate degree correction.

We continue to work with two-layer networks, specified in terms of two intralayer adjacency matrices $\boldsymbol{A}^1$ and $\boldsymbol{A}^2$, with a common block structure specified by a vector $\boldsymbol{g}$. For each node pair $(i, j) \in \mathcal{E}$, we place edges in the two layers according to the probabilities

$$\mathbb{P}(A^1_{ij} = 1) = \theta^1_i \theta^1_j p^1_{g_i g_j} \,, \tag{6.49}$$

$$\mathbb{P}(A^2_{ij} = 1) = \theta^2_i \theta^2_j p^2_{g_i g_j} \,, \tag{6.50}$$

$$\mathbb{P}(A^1_{ij} = 1, A^2_{ij} = 1) = \sqrt{\theta^1_i \theta^1_j \theta^2_i \theta^2_j} q_{g_i g_j} \,. \tag{6.51}$$

We will justify the expression in (6.51) shortly. The quantities θ^l_i and θ^l_j, with $l \in \{1, 2\}$, are the degrees of nodes i and j normalised by the mean degrees. (These are calculated directly from an input degree sequence, so they are not model parameters.) For unipartite networks,

$$\theta^l_i = \frac{d^l_i}{\langle d^l \rangle} \,, \quad \theta^l_j = \frac{d^l_j}{\langle d^l \rangle} \quad \text{for } i, j \in \mathcal{N} \,.$$

For bipartite networks we normalise by the mean degree across the relevant node type (i.e., we normalise customer degrees by the mean degree across all customer nodes, and we normalise product degrees by the mean degree across all product nodes):

$$\theta^l_i = \frac{d^l_i}{\langle d^l_C \rangle} \text{ for } i \in \mathcal{N}_C \,, \quad \theta^l_j = \frac{d^l_j}{\langle d^l_P \rangle} \text{ for } j \in \mathcal{N}_P \,.$$

This normalisation recovers the model in Section 6.3.1 when $\theta_i^l = \theta_j^l = 1$ (i.e., when all nodes of the same type have the same degree). We emphasise that θ_i^l and θ_j^l are known quantities rather than model parameters. Their role is to ensure that the model produces networks whose expected degree sequence in layer $l \in \{1, 2\}$ matches that of the given adjacency matrices $\boldsymbol{A}^1$ and $\boldsymbol{A}^2$.

The probabilities in equations (6.49)–(6.50) ensure that, marginally, $\boldsymbol{A}^1$ and $\boldsymbol{A}^2$ are generated according to single-layer degree-corrected SBMs [76]. It is not obvious how to model the joint probability $\mathbb{P}(A_{ij}^1 = 1, A_{ij}^2 = 1)$. In particular, it is not clear how it should depend on the observed degrees of nodes i and j in layers 1 and 2. Part of the complication is that there are four such quantities in total for each pair of nodes (i, j). The choice from (6.51) works particularly well when $\rho = 1$ and the normalised degree sequences $\boldsymbol{\theta}^1$ and $\boldsymbol{\theta}^2$ are equal, as it reduces to a single degree-corrected SBM that generates two identical network layers. Another sensible option is to set

$$\mathbb{P}(A_{ij}^1 = 1, A_{ij}^2 = 1) = \theta_i^1 \theta_j^1 \theta_i^2 \theta_j^2 q_{g_i g_j} \,. \tag{6.52}$$

This choice has the nice property that edges in a particular edge bundle $(r, s) \in \mathcal{B}$ are independent if and only if $q_{rs} = p_{rs}^1 p_{rs}^2$, which matches the independence condition from Section 6.3.1 for non-degree-corrected settings. However, we found that this model underperforms the one from (6.49)–(6.51) for edge prediction (see Section 6.5). Specifically, this model makes inaccurate predictions for node pairs (i, j) with $A_{ij}^1 = 0$, $A_{ij}^2 = 1$, and normalised degrees θ_i^1, θ_j^1 that are sufficiently large. The reason for this is that the probability

$$\mathbb{P}(A_{ij}^2 = 1 | A_{ij}^1 = 0) = \frac{\mathbb{P}(A_{ij}^1 = 0, A_{ij}^2 = 1)}{\mathbb{P}(A_{ij}^1 = 1)} = \frac{\theta_i^2 \theta_j^2 p_{g_i g_j}^2 - \theta_i^1 \theta_j^1 \theta_i^2 \theta_j^2 q_{g_i g_j}}{1 - \theta_i^1 \theta_j^1 p_{g_i g_j}^1}$$

becomes negative for such nodes, as $p_{g_i g_j}^2 - \theta_i^1 \theta_j^1 q_{g_i g_j} < 0$. The result is that the model misclassifies these node pairs as non-edges, affecting performance. The problem becomes worse for larger values of $q_{g_i g_j}$, which correspond to larger correlation values. Indeed, we have confirmed using numerical experiments that performance degrades with this model as ρ increases.

It may be worthwhile to consider a hybrid model that uses the formula (6.52) for small values of ρ and (6.51) for large values of ρ. However, we settle on the model described by equations (6.49)–(6.51) as our correlated degree-corrected SBM (DCSBM) for the rest of this chapter.

Maximum-likelihood parameter estimates

The log-likelihood of generating adjacency matrices $\boldsymbol{A}^1$ and $\boldsymbol{A}^2$ from our correlated DCSBM is

$$\begin{aligned}\mathcal{L} = \sum_{(i,j)\in\mathcal{E}} \Bigg[& A^1_{ij}A^2_{ij}\log q_{g_ig_j} \\ & + A^1_{ij}(1-A^2_{ij})\log\left(p^1_{g_ig_j} - \sqrt{\frac{\theta^2_i\theta^2_j}{\theta^1_i\theta^1_j}}q_{g_ig_j}\right) + (1-A^1_{ij})A^2_{ij}\log\left(p^2_{g_ig_j} - \sqrt{\frac{\theta^1_i\theta^1_j}{\theta^2_i\theta^2_j}}q_{g_ig_j}\right) \\ & + (1-A^1_{ij})(1-A^2_{ij})\log(1-\theta^1_i\theta^1_j p^1_{g_ig_j} - \theta^2_i\theta^2_j p^2_{g_ig_j} + \sqrt{\theta^1_i\theta^1_j\theta^2_i\theta^2_j}q_{g_ig_j}) + (\text{const.})\Bigg],\end{aligned}$$

where we have ignored additive terms that only involve known quantities. Alternatively, we can sum over edge bundles $(r,s)\in\mathcal{B}$ and write

$$\begin{aligned}\mathcal{L} = \sum_{(r,s)\in\mathcal{B}}\sum_{(i,j)\in\mathcal{E}} \Bigg[& A^1_{ij}A^2_{ij}\log q_{rs} \\ & + A^1_{ij}(1-A^2_{ij})\log\left(p^1_{rs} - \sqrt{\frac{\theta^2_i\theta^2_j}{\theta^1_i\theta^1_j}}q_{rs}\right) + (1-A^1_{ij})A^2_{ij}\log\left(p^2_{rs} - \sqrt{\frac{\theta^1_i\theta^1_j}{\theta^2_i\theta^2_j}}q_{rs}\right) \\ & + (1-A^1_{ij})(1-A^2_{ij})\log(1-\theta^1_i\theta^1_j p^1_{rs} - \theta^2_i\theta^2_j p^2_{rs} + \theta^1_i\theta^1_j\theta^2_i\theta^2_j q_{rs})\Bigg]\delta(g_i,r)\delta(g_j,s) \\ & + (\text{const.})\,.\end{aligned}$$

For convenience, we use the notation $\sum_{ijrs}(\ldots) = \sum_{(i,j)\in\mathcal{E}}(\ldots)\delta(g_i,r)\delta(g_j,s)$ to indicate sums over node pairs (i,j) that belong to a particular edge bundle (r,s). As in previous models, we seek to maximise $\mathcal{L}$ with respect to the parameters p^1_{rs}, p^2_{rs}, q_{rs} by setting the corresponding derivatives to 0. For each different edge bundle (r,s), we obtain the following set of equations:

$$\sum_{ijrs}\frac{A^1_{ij}(1-A^2_{ij})}{p^1_{rs} - \sqrt{\frac{\theta^2_i\theta^2_j}{\theta^1_i\theta^1_j}}q_{rs}} - \sum_{ijrs}\frac{(1-A^1_{ij})(1-A^2_{ij})\theta^1_i\theta^1_j}{1-\theta^1_i\theta^1_j p^1_{rs} - \theta^2_i\theta^2_j p^2_{rs} + \sqrt{\theta^1_i\theta^1_j\theta^2_i\theta^2_j}q_{rs}} = 0\,, \tag{6.53}$$

$$\sum_{ijrs}\frac{(1-A^1_{ij})A^2_{ij}}{p^2_{rs} - \sqrt{\frac{\theta^1_i\theta^1_j}{\theta^2_i\theta^2_j}}q_{rs}} - \sum_{ijrs}\frac{(1-A^1_{ij})(1-A^2_{ij})\theta^2_i\theta^2_j}{1-\theta^1_i\theta^1_j p^1_{rs} - \theta^2_i\theta^2_j p^2_{rs} + \sqrt{\theta^1_i\theta^1_j\theta^2_i\theta^2_j}q_{rs}} = 0\,, \tag{6.54}$$

$$\sum_{ijrs}\frac{A^1_{ij}A^2_{ij}}{q_{rs}} - \sum_{ijrs}\frac{A^1_{ij}(1-A^2_{ij})\sqrt{\frac{\theta^2_i\theta^2_j}{\theta^1_i\theta^1_j}}}{p^1_{rs} - \sqrt{\frac{\theta^2_i\theta^2_j}{\theta^1_i\theta^1_j}}q_{rs}} - \sum_{ijrs}\frac{(1-A^1_{ij})A^2_{ij}\sqrt{\frac{\theta^1_i\theta^1_j}{\theta^2_i\theta^2_j}}}{p^2_{rs} - \sqrt{\frac{\theta^1_i\theta^1_j}{\theta^2_i\theta^2_j}}q_{rs}}$$

$$+\sum_{ijrs}\frac{(1-A^1_{ij})(1-A^2_{ij})\sqrt{\theta^1_i\theta^1_j\theta^2_i\theta^2_j}}{1-\theta^1_i\theta^1_j p^1_{rs}-\theta^2_i\theta^2_j p^2_{rs}+\sqrt{\theta^1_i\theta^1_j\theta^2_i\theta^2_j}q_{rs}}=0\,. \qquad (6.55)$$

Solving equations (6.53)–(6.55) efficiently is difficult, because the sums in each involve $\mathcal{O}(N^2/K^2)$ terms, where K is the number of blocks. However, we are able to simplify these equations with some approximations. Recall that $\theta^l_i = 1$ if the degree of node i is equal to the mean degree. If deviations from this mean are typically small, we can write (for $(i,j)\in\mathcal{E}$) the following:

$$\theta^1_i\theta^1_j = 1+\varepsilon^1_{ij}\,,$$
$$\theta^2_i\theta^2_j = 1+\varepsilon^2_{ij}\,,$$

where ε^1_{ij} and ε^2_{ij} are small parameters (which can be either positive or negative). Additionally,

$$\sqrt{\theta^1_i\theta^1_j\theta^2_i\theta^2_j}=\sqrt{(1+\varepsilon^1_{ij})(1+\varepsilon^2_{ij})}=\sqrt{1+\varepsilon^1_{ij}+\varepsilon^2_{ij}+\varepsilon^1_{ij}\varepsilon^2_{ij}}\approx 1+\frac{\varepsilon^1_{ij}+\varepsilon^2_{ij}}{2}\,,$$

using a first-order Taylor expansion for the approximation and ignoring the quadratic term. We also calculate

$$\sqrt{\frac{\theta^1_i\theta^1_j}{\theta^2_i\theta^2_j}}=\sqrt{\frac{1+\varepsilon^1_{ij}}{1+\varepsilon^2_{ij}}}\approx\sqrt{(1+\varepsilon^1_{ij})(1-\varepsilon^2_{ij})}\approx 1+\frac{\varepsilon^1_{ij}-\varepsilon^2_{ij}}{2}$$

and

$$\sqrt{\frac{\theta^2_i\theta^2_j}{\theta^1_i\theta^1_j}}\approx 1+\frac{\varepsilon^2_{ij}-\varepsilon^1_{ij}}{2}\,.$$

We now seek to expand the left-hand sides of equations (6.53)–(6.55) to first order in ε^1_{ij} and ε^2_{ij}. (We write $\mathcal{O}\left((\varepsilon)^2\right)$ for higher-order terms that we ignore in the resulting approximations.) We start with the first sum in equation (6.53), which we approximate as follows:

$$\begin{aligned}\sum_{ijrs}\frac{A^1_{ij}(1-A^2_{ij})}{p^1_{rs}-\sqrt{\frac{\theta^2_i\theta^2_j}{\theta^1_i\theta^1_j}}q_{rs}} &\approx \sum_{ijrs}\frac{A^1_{ij}(1-A^2_{ij})}{p^1_{rs}-q_{rs}-\frac{\varepsilon^2_{ij}-\varepsilon^1_{ij}}{2}q_{rs}}\\ &= \frac{1}{p^1_{rs}-q_{rs}}\sum_{ijrs}A^1_{ij}(1-A^2_{ij})\\ &\quad+\frac{q_{rs}}{(p^1_{rs}-q_{rs})^2}\sum_{ijrs}\frac{A^1_{ij}(1-A^2_{ij})(\varepsilon^2_{ij}-\varepsilon^1_{ij})}{2}+\mathcal{O}\left((\varepsilon)^2\right)\,. \qquad (6.56)\end{aligned}$$

As in Section 6.3, let

$$e_{rs}^{10} = \big|\{(i,j) \in \mathcal{E} : A_{ij}^1 = 1, A_{ij}^2 = 0, g_i = r, g_j = s\}\big|$$

denote the number of node pairs (i,j) in edge bundle (r,s) that are adjacent in the first layer but not in the second. Similarly, we define the quantity

$$g_{rs}^{10} = \sum_{ijrs} A_{ij}^1(1 - A_{ij}^2)(\varepsilon_{ij}^2 - \varepsilon_{ij}^1) = \sum_{ijrs} A_{ij}^1(1 - A_{ij}^2)(\theta_i^2\theta_j^2 - \theta_i^1\theta_j^1)\,,$$

which is a constant that can be calculated for any given network $\boldsymbol{A}$ and block partition $\boldsymbol{g}$. The approximation in (6.56) then becomes

$$\sum_{ijrs} \frac{A_{ij}^1(1 - A_{ij}^2)}{p_{rs}^1 - \sqrt{\frac{\theta_i^2\theta_j^2}{\theta_i^1\theta_j^1}}q_{rs}} \approx \frac{e_{rs}^{10}}{p_{rs}^1 - q_{rs}} + \frac{g_{rs}^{10}}{2}\frac{q_{rs}}{(p_{rs}^1 - q_{rs})^2}\,.$$

The other terms from (6.53)–(6.55) can be expanded to first order in a similar way, yielding the following system of equations:

$$\begin{aligned}
&\frac{e_{rs}^{10}}{p_{rs}^1 - q_{rs}} - \frac{e_{rs}^{00}}{1 - p_{rs}^1 - p_{rs}^2 + q_{rs}} \\
&\quad + \frac{g_{rs}^{10}}{2}\frac{q_{rs}}{(p_{rs}^1 - q_{rs})^2} - f_{rs}^1\frac{1 - p_{rs}^2 + q_{rs}/2}{(1 - p_{rs}^1 - p_{rs}^2 + q_{rs})^2} - f_{rs}^2\frac{p_{rs}^2 - q_{rs}/2}{(1 - p_{rs}^1 - p_{rs}^2 + q_{rs})^2} &&= 0\,, \\
&\frac{e_{rs}^{01}}{p_{rs}^2 - q_{rs}} - \frac{e_{rs}^{00}}{1 - p_{rs}^1 - p_{rs}^2 + q_{rs}} \\
&\quad + \frac{g_{rs}^{01}}{2}\frac{q_{rs}}{(p_{rs}^2 - q_{rs})^2} - f_{rs}^1\frac{p_{rs}^1 - q_{rs}/2}{(1 - p_{rs}^1 - p_{rs}^2 + q_{rs})^2} - f_{rs}^2\frac{1 - p_{rs}^1 + q_{rs}/2}{(1 - p_{rs}^1 - p_{rs}^2 + q_{rs})^2} &&= 0\,, \\
&\frac{e_{rs}^{11}}{q_{rs}} - \frac{e_{rs}^{10}}{p_{rs}^1 - q_{rs}} - \frac{e_{rs}^{01}}{p_{rs}^2 - q_{rs}} + \frac{e_{rs}^{00}}{1 - p_{rs}^1 - p_{rs}^2 + q_{rs}} \\
&\quad - \frac{g_{rs}^{10}}{2}\frac{p_{rs}^1}{(p_{rs}^1 - q_{rs})^2} - \frac{g_{rs}^{01}}{2}\frac{p_{rs}^2}{(p_{rs}^2 - q_{rs})^2} + \frac{1}{2}\frac{f_{rs}^1(1 + p_{rs}^1 - p_{rs}^2) + f_{rs}^2(1 - p_{rs}^1 + p_{rs}^2)}{(1 - p_{rs}^1 - p_{rs}^2 + q_{rs})^2} &&= 0\,.
\end{aligned} \tag{6.57}$$

In these equations, e_{rs}^{11}, e_{rs}^{10}, e_{rs}^{01}, and e_{rs}^{00} are defined as in Section 6.3.1. Additionally, we set

$$\begin{aligned}
g_{rs}^{10} &= \sum_{ijrs} A_{ij}^1(1 - A_{ij}^2)(\varepsilon_{ij}^2 - \varepsilon_{ij}^1) = \sum_{ijrs} A_{ij}^1(1 - A_{ij}^2)(\theta_i^2\theta_j^2 - \theta_i^1\theta_j^1)\,, \\
g_{rs}^{01} &= \sum_{ijrs} (1 - A_{ij}^1)A_{ij}^2(\varepsilon_{ij}^1 - \varepsilon_{ij}^2) = \sum_{ijrs} (1 - A_{ij}^1)A_{ij}^2(\theta_i^1\theta_j^1 - \theta_i^2\theta_j^2)\,, \\
f_{rs}^1 &= \sum_{ijrs} (1 - A_{ij}^1)(1 - A_{ij}^2)\varepsilon_{ij}^1 = \sum_{ijrs} (1 - A_{ij}^1)(1 - A_{ij}^2)(\theta_i^1\theta_j^1 - 1)\,,
\end{aligned}$$

$$f_{rs}^2 = \sum_{ijrs} (1 - A_{ij}^1)(1 - A_{ij}^2)\varepsilon_{ij}^2 = \sum_{ijrs} (1 - A_{ij}^1)(1 - A_{ij}^2)(\theta_i^2\theta_j^2 - 1)\,,$$

all of which can be calculated from the matrices $\boldsymbol{A}^1$ and $\boldsymbol{A}^2$.

Equations (6.57) reduce to equations (6.28)–(6.30) from Section 6.3.1 (where we considered non-degree-corrected Bernoulli SBMs) if we ignore all of the terms that depend on ε_{ij}^l (i.e., the terms that correspond to perturbations of the degrees from their mean values). In fact, the zeroth-order solution that we obtain from ignoring these terms provides a good initialisation of a numerical algorithm that solves equations (6.57) for the parameters p_{rs}^1, p_{rs}^2, and q_{rs}. In practice, when using correlated DCSBMs for edge prediction (see Section 6.5), we find that using a first-order approximation to determine p_{rs}^1, p_{rs}^2, and q_{rs} does not improve performance significantly over using the zeroth-order solution. Given that the latter is significantly easier to determine than the former, we suggest using the zeroth-order parameter estimates in first instance.

Correlation values

For the non-degree-corrected SBMs from Section 6.3, pairs of nodes (i, j) in a given edge bundle (r, s) have the same Pearson correlation ρ_{rs}. This no longer holds for degree-corrected models. Instead, each pair of nodes (i, j) has its own correlation value

$$\varrho_{ij} = \frac{\mathbb{E}[A_{ij}^1 A_{ij}^2] - \mathbb{E}[A_{ij}^1]\mathbb{E}[A_{ij}^2]}{\sigma[A_{ij}^1]\sigma[A_{ij}^2]} = \frac{\sqrt{\theta_i^1\theta_j^1\theta_i^2\theta_j^2}q_{rs} - \theta_i^1\theta_j^1\theta_i^2\theta_j^2 p_{rs}^1 p_{rs}^2}{\sqrt{\theta_i^1\theta_j^1 p_{rs}^1(1 - \theta_i^1\theta_j^1 p_{rs}^1)\theta_i^2\theta_j^2 p_{rs}^2(1 - \theta_i^2\theta_j^2 p_{rs}^2)}}\,. \quad (6.58)$$

Similarly to our earlier expansions of the ML equations, we can approximate ϱ_{ij} to first order in ε_{ij}^1 and ε_{ij}^2. Rearranging (6.58), we obtain

$$\begin{aligned}\varrho_{ij} &= \frac{q_{rs} - p_{rs}^1 p_{rs}^2}{\sqrt{p_{rs}^1(1 - p_{rs}^1)p_{rs}^2(1 - p_{rs}^2)}}\sqrt{\frac{1 - p_{rs}^1}{1 - \theta_i^1\theta_j^1 p_{rs}^1}}\sqrt{\frac{1 - p_{rs}^2}{1 - \theta_i^2\theta_j^2 p_{rs}^2}}\frac{q_{rs} - \sqrt{\theta_i^1\theta_j^1\theta_i^2\theta_j^2}p_{rs}^1 p_{rs}^2}{q_{rs} - p_{rs}^1 p_{rs}^2} \\ &= \rho_{rs}\sqrt{\frac{1 - p_{rs}^1}{1 - \theta_i^1\theta_j^1 p_{rs}^1}}\sqrt{\frac{1 - p_{rs}^2}{1 - \theta_i^2\theta_j^2 p_{rs}^2}}\frac{q_{rs} - \sqrt{\theta_i^1\theta_j^1\theta_i^2\theta_j^2}p_{rs}^1 p_{rs}^2}{q_{rs} - p_{rs}^1 p_{rs}^2}\,,\end{aligned}$$

where ρ_{rs} is the Pearson correlation for edge bundle (r, s) when not accounting for degree correction. We have

$$\sqrt{\frac{1 - p_{rs}^1}{1 - \theta_i^1\theta_j^1 p_{rs}^1}} = \sqrt{\frac{1 - p_{rs}^1}{1 - p_{rs}^1 - \varepsilon_{ij}^1 p_{rs}^1}} = \sqrt{\frac{1}{1 - \varepsilon_{ij}^1\frac{p_{rs}^1}{1 - p_{rs}^1}}} \approx 1 + \frac{\varepsilon_{ij}^1}{2}\frac{p_{rs}^1}{1 - p_{rs}^1}\,,$$

and similarly

$$\sqrt{\frac{1-p_{rs}^2}{1-\theta_i^2\theta_j^2 p_{rs}^2}} \approx 1 + \frac{\varepsilon_{ij}^2}{2}\frac{p_{rs}^2}{1-p_{rs}^2}\,.$$

Additionally,

$$\frac{q_{rs} - \sqrt{\theta_i^1\theta_j^1\theta_i^2\theta_j^2}p_{rs}^1 p_{rs}^2}{q_{rs} - p_{rs}^1 p_{rs}^2} \approx \frac{q_{rs} - \left(1 + \frac{\varepsilon_{ij}^1+\varepsilon_{ij}^2}{2}\right)p_{rs}^1 p_{rs}^2}{q_{rs} - p_{rs}^1 p_{rs}^2} = 1 - \frac{\varepsilon_{ij}^1+\varepsilon_{ij}^2}{2}\frac{p_{rs}^1 p_{rs}^2}{q_{rs} - p_{rs}^1 p_{rs}^2}\,.$$

It follows that the first-order approximation of ϱ_{ij} is

$$\begin{aligned}\varrho_{ij} &\approx \rho_{rs}\left(1 + \frac{\varepsilon_{ij}^1}{2}\frac{p_{rs}^1}{1-p_{rs}^1}\right)\left(1 + \frac{\varepsilon_{ij}^2}{2}\frac{p_{rs}^2}{1-p_{rs}^2}\right)\left(1 - \frac{\varepsilon_{ij}^1+\varepsilon_{ij}^2}{2}\frac{p_{rs}^1 p_{rs}^2}{q_{rs} - p_{rs}^1 p_{rs}^2}\right) \\ &\approx \rho_{rs} + \rho_{rs}\left(\frac{\varepsilon_{ij}^1}{2}\frac{p_{rs}^1}{1-p_{rs}^1} + \frac{\varepsilon_{ij}^2}{2}\frac{p_{rs}^2}{1-p_{rs}^2} - \frac{\varepsilon_{ij}^1+\varepsilon_{ij}^2}{2}\frac{p_{rs}^1 p_{rs}^2}{q_{rs} - p_{rs}^1 p_{rs}^2}\right)\,. \qquad (6.59)\end{aligned}$$

Note that ignoring terms that depend on ε_{ij}^1 and ε_{ij}^2 (i.e., terms that correspond to perturbations of the degrees from their mean values), we have $\varrho_{ij} \approx \rho_{rs}$. This approximation works especially well when p_{rs}^1 and p_{rs}^2 are also small (so that the respective network layers are sparse). We note also that the case $q_{rs} = p_{rs}^1 p_{rs}^2$ requires separate consideration, to avoid dividing by 0. Similar approximations give

$$\varrho_{ij} \approx -\frac{\varepsilon_{ij}^1+\varepsilon_{ij}^2}{2}\left(1 + \frac{\varepsilon_{ij}^1}{2}\frac{p_{rs}^1}{1-p_{rs}^1}\right)\left(1 + \frac{\varepsilon_{ij}^2}{2}\frac{p_{rs}^2}{1-p_{rs}^2}\right)\sqrt{\frac{p_{rs}^1}{1-p_{rs}^1}}\sqrt{\frac{p_{rs}^2}{1-p_{rs}^2}}\,.$$

In particular, the zeroth-order solution gives $\varrho \approx 0$, in agreement with the non-degree-corrected SBM from Section 6.3.1.

6.4 Correlated SBMs with Heterogeneous Blocks

Our objective in this section is to describe a model of correlated networks that does not constrain the layers to have identical block structures. In defining such a model, one key requirement is that, marginally, each layer reduces to a standard SBM (with or without degree correction). This way, if a given network's layers are uncorrelated, our proposed model reduces to two independent SBMs (one for each layer).

Let $\boldsymbol{g}^1$ and $\boldsymbol{g}^2$ denote the block assignments of nodes in the first and second layer, respectively. As in our previous discussions in this chapter, we consider these vectors to be given quantities, and we focus on inferring the remaining model parameters. Assuming that edges are sampled using Bernoulli random variables, the condition that each layer is marginally an SBM gives the following equalities:

$$\mathbb{P}(A_{ij}^l = 1) = p_{g_i^l g_j^l}^l \qquad \text{for } l = 1, 2 \quad \text{(no degree correction)}\,,$$

$$\mathbb{P}(A^l_{ij} = 1) = \theta^l_i \theta^l_j p^l_{g^l_i g^l_j} \qquad \text{for } l = 1, 2 \quad \text{(degree correction)}\,.$$

To complete the model, we need to prescribe the joint probabilities $\mathbb{P}(A^1_{ij} = 1, A^2_{ij} = 1)$. We continue to assume that nodes in the same block are statistically equivalent (aside from degree correction), which implies that these joint probabilities depend only on the block assignments of nodes i and j in the two layers. In the most general case, there are four such labels to consider: g^1_i, g^1_j, g^2_i, and g^2_j. If each different combination of groups has its own parameter, there is a fourth-order tensor $\boldsymbol{q}$ prescribing the joint probabilities $\mathbb{P}(A^1_{ij} = 1, A^2_{ij} = 1)$.

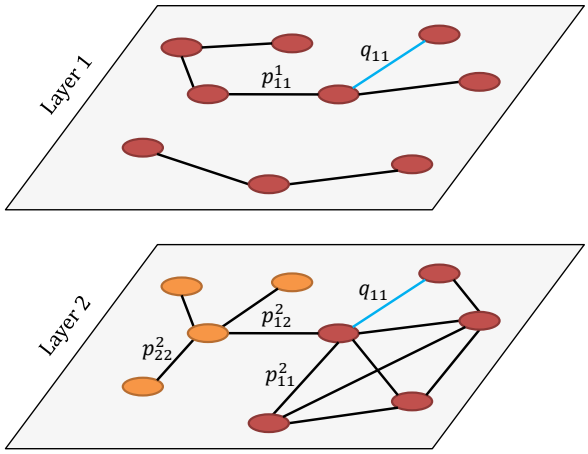


Figure 6.3: Example of a network with one block in layer 1 and two blocks in layer 2. (Node color indicates block membership.) The marginal edge probability in layer 1 is p^1_{11}, and the marginal edge probabilities in layer 2 are p^2_{11}, p^2_{12}, and p^2_{22}. Correlated node pairs are adjacent in both layers with probability q_{11}.

We consider a slightly less general case, where the edges between two nodes i and j are correlated if and only if both i and j have the same block assignment in both layers. (Within each layer, i and j may still belong to different blocks.) In other words, if there is no degree correction, we impose

$$\mathbb{P}(A^1_{ij} = 1, A^2_{ij} = 1) = \begin{cases} q_{g_i g_j}\,, & \text{if } (g^1_i, g^1_j) = (g^2_i, g^2_j) \equiv (g_i, g_j)\,, \\ p^1_{g^1_i g^1_j} p^2_{g^2_i g^2_j}\,, & \text{if } (g^1_i, g^1_j) \neq (g^2_i, g^2_j)\,. \end{cases}$$

Similarly, in the degree-corrected case, we extend the model from Section 6.3.3 and set

$$\mathbb{P}(A^1_{ij} = 1, A^2_{ij} = 1) = \begin{cases} \sqrt{\theta^1_i \theta^1_j \theta^2_i \theta^2_j}\, q_{g_i g_j}\,, & \text{if } (g^1_i, g^1_j) = (g^2_i, g^2_j) \equiv (g_i, g_j)\,, \\ \theta^1_i \theta^1_j \theta^2_i \theta^2_j p^1_{g^1_i g^1_j} p^2_{g^2_i g^2_j}\,, & \text{if } (g^1_i, g^1_j) \neq (g^2_i, g^2_j)\,. \end{cases}$$

This modelling choice defines two sets of (potential) edges:

$$\mathcal{C} = \{(i,j) : (g_i^1, g_j^1) = (g_i^2, g_j^2)\} = \{\text{correlated edges}\}\,,$$
$$\mathcal{U} = \{(i,j) : (g_i^1, g_j^1) \neq (g_i^2, g_j^2)\} = \{\text{uncorrelated edges}\}\,.$$

Note that $\mathcal{C} \cap \mathcal{U} = \emptyset$ and $\mathcal{C} \cup \mathcal{U} = \mathcal{E}$. For uncorrelated edges, with or without degree correction, the following independence condition holds:

$$\mathbb{P}(A_{ij}^1 = 1, A_{ij}^2 = 1) = \mathbb{P}(A_{ij}^1 = 1)\mathbb{P}(A_{ij}^2 = 1)\,.$$

For correlated edges, the random variables A_{ij}^1 and A_{ij}^2 are no longer independent, and the amount of correlation between them is controlled by the value of $q_{g_i g_j}$. We illustrate this model with a toy network in Figure 6.3.

6.4.1 Bernoulli Model Without Degree Correction

We now discuss in more detail the version of this model without degree correction, and we leave the derivation of degree-corrected variants for future work.

The likelihood of observing a multilayer network with intralayer adjacency matrices $\boldsymbol{A}^1$ and $\boldsymbol{A}^2$ given block structures $\boldsymbol{g}^1$ and $\boldsymbol{g}^2$ and parameter values $\boldsymbol{p}^1$, $\boldsymbol{p}^2$, and $\boldsymbol{q}$ is

$$\begin{aligned}
&\mathbb{P}(\boldsymbol{A}^1, \boldsymbol{A}^2 | \boldsymbol{g}, \boldsymbol{p}^1, \boldsymbol{p}^2, \boldsymbol{q}) \\
&\quad = \prod_{(i,j)\in\mathcal{C}} \Bigg[q_{g_i g_j}^{A_{ij}^1 A_{ij}^2} \left(p_{g_i g_j}^1 - q_{g_i g_j}\right)^{A_{ij}^1\left(1-A_{ij}^2\right)} \left(p_{g_i g_j}^2 - q_{g_i g_j}\right)^{\left(1-A_{ij}^1\right)A_{ij}^2} \\
&\qquad\qquad \times \left(1 - p_{g_i g_j}^1 - p_{g_i g_j}^2 + q_{g_i g_j}\right)^{\left(1-A_{ij}^1\right)\left(1-A_{ij}^2\right)} \Bigg] \\
&\quad \times \prod_{(i,j)\in\mathcal{U}} \left[\left(p_{g_i^1 g_j^1}^1\right)^{A_{ij}^1} \left(1 - p_{g_i^1 g_j^1}^1\right)^{1-A_{ij}^1} \left(p_{g_i^2 g_j^2}^2\right)^{A_{ij}^2} \left(1 - p_{g_i^2 g_j^2}^2\right)^{1-A_{ij}^2} \right].
\end{aligned} \tag{6.60}$$

Taking the logarithm of both sides of (6.60) yields the following expression for the log-likelihood:

$$\begin{aligned}
\mathcal{L} = \sum_{(i,j)\in\mathcal{C}} \Big[& A_{ij}^1 A_{ij}^2 \log q_{g_i g_j} \\
& + A_{ij}^1 \left(1 - A_{ij}^2\right) \log\left(p_{g_i g_j}^1 - q_{g_i g_j}\right) + \left(1 - A_{ij}^1\right) A_{ij}^2 \log\left(p_{g_i g_j}^2 - q_{g_i g_j}\right) \\
& + \left(1 - A_{ij}^1\right)\left(1 - A_{ij}^2\right) \log\left(1 - p_{g_i g_j}^1 - p_{g_i g_j}^2 + q_{g_i g_j}\right) \Big] \\
+ \sum_{(i,j)\in\mathcal{U}} \Big[& A_{ij}^1 \log p_{g_i^1 g_j^1}^1 + \left(1 - A_{ij}^1\right) \log\left(1 - p_{g_i^1 g_j^1}^1\right) \\
& + A_{ij}^2 \log p_{g_i^2 g_j^2}^2 + \left(1 - A_{ij}^2\right) \log\left(1 - p_{g_i^2 g_j^2}^2\right) \Big].
\end{aligned}$$

Going from a sum over nodes pairs (i, j) to a sum over edge bundles (r, s) yields

$$\begin{aligned}\mathcal{L} = &\sum_{(r,s)\in\mathcal{B}} \Big[e_{rs}^{11,\mathcal{C}} \log q_{rs} + e_{rs}^{10,\mathcal{C}} \log\left(p_{rs}^1 - q_{rs}\right) + e_{rs}^{01,\mathcal{C}} \log\left(p_{rs}^2 - q_{rs}\right) \\ &\qquad + e_{rs}^{00,\mathcal{C}} \log(1 - p_{rs}^1 - p_{rs}^2 + q_{rs}) \Big] \\ &+ \sum_{(r,s)\in\mathcal{B}} \left[m_{rs}^{1,\mathcal{U}} \log p_{rs}^1 + (e_{rs}^{1,\mathcal{U}} - m_{rs}^{1,\mathcal{U}}) \log(1 - p_{rs}^1) \right] \\ &+ \sum_{(r,s)\in\mathcal{B}} \left[m_{rs}^{2,\mathcal{U}} \log p_{rs}^2 + (e_{rs}^{2,\mathcal{U}} - m_{rs}^{2,\mathcal{U}}) \log(1 - p_{rs}^2) \right],\end{aligned}$$

where

$$\begin{aligned} e_{rs}^{11,\mathcal{C}} &:= \left|\{(i,j) : g_i^1 = g_i^2 = r, g_i^2 = g_j^2 = s, A_{ij}^1 = 1, A_{ij}^2 = 1\}\right|, \\ e_{rs}^{10,\mathcal{C}} &:= \left|\{(i,j) : g_i^1 = g_i^2 = r, g_i^2 = g_j^2 = s, A_{ij}^1 = 1, A_{ij}^2 = 0\}\right|, \\ e_{rs}^{01,\mathcal{C}} &:= \left|\{(i,j) : g_i^1 = g_i^2 = r, g_i^2 = g_j^2 = s, A_{ij}^1 = 0, A_{ij}^2 = 1\}\right|, \\ e_{rs}^{00,\mathcal{C}} &:= \left|\{(i,j) : g_i^1 = g_i^2 = r, g_i^2 = g_j^2 = s, A_{ij}^1 = 0, A_{ij}^2 = 0\}\right|, \\ m_{rs}^{l,\mathcal{U}} &:= \left|\{(i,j) : (i,j) \in \mathcal{U}, g_i^l = r, g_j^l = s, A_{ij}^l = 1\}\right| \text{ for } l = 1, 2, \\ e_{rs}^{l,\mathcal{U}} &:= \left|\{(i,j) : (i,j) \in \mathcal{U}, g_i^l = r, g_j^l = s\}\right| \text{ for } l = 1, 2. \end{aligned} \tag{6.61}$$

The first four quantities in (6.61) denote, respectively, the number of correlated node pairs in blocks r and s that are adjacent in both layers, the number that are adjacent in the first layer but not in the second, the number that are adjacent in the second layer but not in the first, and the number that are not adjacent in either layer. For each layer l, the quantity $m_{rs}^{l,\mathcal{U}}$ gives the number of uncorrelated edges between blocks r and s, and $e_{rs}^{l,\mathcal{U}}$ gives the number of uncorrelated node pairs (both edges and non-edges) in blocks r and s.

With $\boldsymbol{g}^1$ and $\boldsymbol{g}^2$ fixed, we can find ML estimates of the parameters p_{rs}^1, p_{rs}^2 and q_{rs} by taking derivatives of $\mathcal{L}$ and equating them to 0:

$$\frac{\partial\mathcal{L}}{\partial p_{rs}^1} = 0 \quad\Leftrightarrow\quad \frac{e_{rs}^{10,\mathcal{C}}}{p_{rs}^1 - q_{rs}} - \frac{e_{rs}^{00,\mathcal{C}}}{1 - p_{rs}^1 - p_{rs}^2 + q_{rs}} + \frac{m_{rs}^{1,\mathcal{U}}}{p_{rs}^1} - \frac{e_{rs}^{1,\mathcal{U}} - m_{rs}^{1,\mathcal{U}}}{1 - p_{rs}^1} = 0, \tag{6.62}$$

$$\frac{\partial\mathcal{L}}{\partial p_{rs}^2} = 0 \quad\Leftrightarrow\quad \frac{e_{rs}^{01,\mathcal{C}}}{p_{rs}^2 - q_{rs}} - \frac{e_{rs}^{00,\mathcal{C}}}{1 - p_{rs}^1 - p_{rs}^2 + q_{rs}} + \frac{m_{rs}^{2,\mathcal{U}}}{p_{rs}^2} - \frac{e_{rs}^{2,\mathcal{U}} - m_{rs}^{2,\mathcal{U}}}{1 - p_{rs}^2} = 0, \tag{6.63}$$

$$\frac{\partial\mathcal{L}}{\partial q_{rs}} = 0 \quad\Leftrightarrow\quad \frac{e_{rs}^{11,\mathcal{C}}}{q_{rs}} - \frac{e_{rs}^{10,\mathcal{C}}}{p_{rs}^1 - q_{rs}} - \frac{e_{rs}^{01,\mathcal{C}}}{p_{rs}^2 - q_{rs}} + \frac{e_{rs}^{00,\mathcal{C}}}{1 - p_{rs}^1 - p_{rs}^2 + q_{rs}} = 0. \tag{6.64}$$

The system of equations (6.62)–(6.64) does not have a closed-form solution, so it needs to be solved numerically (similarly to the correlated degree-corrected SBM from Section 6.3.3). After obtaining ML estimates $\widehat{p}_{rs}^1$, $\widehat{p}_{rs}^2$, and $\widehat{q}_{rs}$, the estimated

correlation value $\widehat{\rho}_{rs}$ for edge bundle (r, s) follows from the definition of the Pearson correlation (see equation (6.6)):

$$\widehat{\rho}_{rs} = \frac{\widehat{q}_{rs} - \widehat{p}^1_{rs}\widehat{p}^2_{rs}}{\sqrt{\widehat{p}^1_{rs}\left(1 - \widehat{p}^1_{rs}\right)\widehat{p}^2_{rs}\left(1 - \widehat{p}^2_{rs}\right)}}\,. \tag{6.65}$$

Although we do not do so here, one can potentially use this model (and its degree-corrected variants) to detect correlated communities in multilayer networks. The distinction that we have made between correlated and uncorrelated edges ensures that communities that persist across multiple layers do so because the edges in those communities tend to be correlated across those layers. This offers a different interpretation from that presented in Chapter 5 of what it means for a community to span multiple layers.

In Table 6.1, we list all of the models that we proposed in this chapter, along with several other models that we have not analysed but which we believe to be straightforward modifications of the existing ones. For each model, we provide either formulas for the ML parameter and correlation estimates or the equations that one needs to solve to get these estimates.

Table 6.1: Maximum-likelihood parameter estimates and correlation estimates for all the models considered in Sections 6.2–6.4.

Block structure	Degree corrected?	Edge model	Parameter estimates	Correlation estimate
None (ER)	No	Bernoulli	$\widehat{p_1} = \frac{m_1}{\lvert\mathcal{E}\rvert}$, $\widehat{p_2} = \frac{m_2}{\lvert\mathcal{E}\rvert}$, $\widehat{q} = \frac{e_{11}}{\lvert\mathcal{E}\rvert}$	$\widehat{\rho} = \frac{e_{00}e_{11} - e_{10}e_{01}}{\sqrt{(e_{11}+e_{10})(e_{11}+e_{01})(e_{10}+e_{00})(e_{01}+e_{00})}}$
None (ER)	No	Poisson	$\widehat{\mu}_0 = \frac{e_{11} - e_{00} + \sqrt{(e_{11}+e_{00})^2 - 4e_{10}e_{01}}}{2\lvert\mathcal{E}\rvert}$, $\widehat{\lambda}_1 = \frac{m_1}{\lvert\mathcal{E}\rvert}$, $\widehat{\lambda}_2 = \frac{m_2}{\lvert\mathcal{E}\rvert}$	$\widehat{\rho} = \frac{e_{11} - e_{00} + \sqrt{(e_{11}+e_{00})^2 - 4e_{10}e_{01}}}{\sqrt{(e_{11}+e_{10})(e_{11}+e_{01})}}$
Fixed	No	Bernoulli	$\widehat{p}^1_{rs} = \frac{m^1_{rs}}{e_{rs}}$, $\widehat{p}^2_{rs} = \frac{m^2_{rs}}{e_{rs}}$, $\widehat{q} = \frac{e^{11}_{rs}}{e_{rs}}$	$\widehat{\rho}_{rs} = \frac{e^{00}_{rs}e^{11}_{rs} - e^{10}_{rs}e^{01}_{rs}}{\sqrt{(e^{11}_{rs}+e^{10}_{rs})(e^{11}_{rs}+e^{01}_{rs})(e^{10}_{rs}+e^{00}_{rs})(e^{01}_{rs}+e^{00}_{rs})}}$
Fixed	No	Poisson	$\widehat{\mu}^0_{rs} = \frac{e^{11}_{rs} - e^{00}_{rs} + \sqrt{(e^{11}_{rs}+e^{00}_{rs})^2 - 4e^{10}_{rs}e^{01}_{rs}}}{e_{rs}}$, $\widehat{\lambda}^1_{rs} = \frac{m^1_{rs}}{e_{rs}}$, $\widehat{\lambda}^2_{rs} = \frac{m^2_{rs}}{e_{rs}}$	$\widehat{\rho}_{rs} = \frac{e^{11}_{rs} - e^{00}_{rs} + \sqrt{(e^{11}_{rs}+e^{00}_{rs})^2 - 4e^{10}_{rs}e^{01}_{rs}}}{\sqrt{m^1_{rs}m^2_{rs}}}$
Fixed	Yes	Bernoulli	Solutions to system of equations (6.57)	(6.58) or approximation (6.59)
Fixed	Yes	Poisson	Have not considered	Have not considered
Heterogeneous	No	Bernoulli	Solutions to equations (6.62)–(6.64)	$\widehat{\rho}_{rs} = \frac{\widehat{q}_{rs} - \widehat{p}^1_{rs}\widehat{p}^2_{rs}}{\sqrt{\widehat{p}^1_{rs}(1-\widehat{p}^1_{rs})\widehat{p}^2_{rs}(1-\widehat{p}^2_{rs})}}$
Heterogeneous	No	Poisson	Have not considered	Have not considered
Heterogeneous	Yes	Bernoulli	Have not considered	Have not considered
Heterogeneous	Yes	Poisson	Have not considered	Have not considered

6.5 Edge Prediction

Recall from Section 2.5.2 that the aim of edge prediction in networks is to infer likely missing edges and/or spurious edges. Edge prediction is often model-driven, and the first step typically consists of computing probabilities for pairs of nodes (i, j) to form edges. For example, one can use a single-layer degree-corrected Bernoulli SBM for this task. Having inferred some block structure $\boldsymbol{g}$ and edge propensities $\boldsymbol{p}$, the probability that two nodes i and j are adjacent according to the model is

$$\mathbb{P}(A_{ij} = 1) = \theta_i \theta_j p_{g_i g_j} \,. \tag{6.66}$$

The pairs (i, j) for which these probabilities are large, but which are not adjacent in the actual network (i.e., with $A_{ij} = 0$), produce a list of likely candidates for missing edges. Similarly, pairs (i, j) for which these probabilities are small but which are adjacent in the given network (i.e., with $A_{ij} = 1$) are possible spurious edges.

6.5.1 Edge Prediction Using Correlated SBMs

Recently, there have been various attempts to perform edge prediction in multilayer networks [34, 148, 153]. All of these methods use multilayer information to infer mesoscale structures in given networks, but then they perform edge prediction independently in each layer conditioned on the inferred mesoscale structure. In particular, when using one of these approaches, two nodes being adjacent in one layer has no bearing on their probability to be adjacent in a different layer. We aim to address this limitation using our correlated models from Sections 6.2–6.4.

As in our prior discussions in this chapter, consider a network with two layers whose adjacency matrices are $\boldsymbol{A}^1$ and $\boldsymbol{A}^2$. Let $\boldsymbol{g}$ be the shared block structure of these layers. (We leave the task of performing edge prediction using the model from Section 6.4, in which the layers are not restricted to have identical mesoscale structure, for future work.) Our goal is to predict edges in the second layer, conditioned on the adjacency structure of the first layer. For each pair of nodes $(i, j) \in \mathcal{E}$, the key quantities to calculate are the probabilities $\mathbb{P}(A^2_{ij} = 1 | A^1_{ij} = 1)$ and $\mathbb{P}(A^2_{ij} = 1 | A^1_{ij} = 0)$ for i and j to be adjacent in the second layer, conditioned on them either being adjacent or non-adjacent in the first layer. For example, using the correlated Bernoulli SBM from Section 6.3.1 (which has no degree correction), we have

$$\mathbb{P}(A^2_{ij} = 1 | A^1_{ij} = 1) = \frac{\mathbb{P}(A^1_{ij} = 1, A^2_{ij} = 1)}{\mathbb{P}(A^1_{ij} = 1)} = \frac{q_{g_i g_j}}{p^1_{g_i g_j}} \,, \tag{6.67}$$

$$\mathbb{P}(A^2_{ij} = 1 | A^1_{ij} = 0) = \frac{\mathbb{P}(A^1_{ij} = 0, A^2_{ij} = 1)}{\mathbb{P}(A^1_{ij} = 0)} = \frac{p^2_{g_i g_j} - q_{g_i g_j}}{1 - p^1_{g_i g_j}} \,. \tag{6.68}$$

Note that this set of probabilities is the same across all node pairs (i, j) in the same edge bundle (r, s). Now, suppose that we have positive correlation in this edge bundle, so $\rho_{rs} > 0$. From the definition of the Pearson correlation, it follows that $q_{rs} > p^1_{rs} p^2_{rs}$. We then find that $\mathbb{P}(A^2_{ij} = 1 | A^1_{ij} = 1) > p^2_{rs}$ and $\mathbb{P}(A^2_{ij} = 1 | A^1_{ij} = 0) < p^2_{rs}$, whereas using a single-layer SBM would entail that $\mathbb{P}(A^2_{ij} = 1) = p^2_{rs}$. Thus, the effect of incorporating correlations into our edge-prediction model when these correlations are positive is to increase the probability that nodes i and j are adjacent in the second layer when the corresponding edge also exists in the first layer, and to decrease the same probability when the corresponding edge is absent from the first layer. (The effect is reversed for negative correlations.)

Table 6.2: Edge-prediction probabilities for various correlated and single-layer models

Model	$\mathbb{P}(A^2_{ij} = 1 \mid A^1_{ij} = 1)$	$\mathbb{P}(A^2_{ij} = 1 \mid A^1_{ij} = 0)$
Corr. ER	q/p_1	$(p_2 - q)/(1 - p_1)$
Corr. SBM	q_{rs}/p^1_{rs}	$(p^2_{rs} - q_{rs})/(1 - p^1_{rs})$
Corr. CM	$\sqrt{\theta^1_i \theta^1_j \theta^2_i \theta^2_j}\, q/(\theta^1_i \theta^1_j p_1)$	$(\theta^2_i \theta^2_j p_2 - \sqrt{\theta^1_i \theta^1_j \theta^2_i \theta^2_j}\, q)/(1 - \theta^1_i \theta^1_j p_1)$
Corr. DCSBM	$\sqrt{\theta^1_i \theta^1_j \theta^2_i \theta^2_j}\, q_{rs}/(\theta^1_i \theta^1_j p^1_{rs})$	$(\theta^2_i \theta^2_j p^2_{rs} - \sqrt{\theta^1_i \theta^1_j \theta^2_i \theta^2_j}\, q_{rs})/(1 - \theta^1_i \theta^1_j p^1_{rs})$
SBM	p^2_{rs}	
DCSBM	$\theta^2_i \theta^2_j p^2_{rs}$	

In Table 6.2, we summarise the two key probabilities (6.67)–(6.68) for four different correlated models (using the Bernoulli variants), alongside the probabilities $\mathbb{P}(A^2_{ij} = 1)$ that correspond to single-layer SBMs and DCSBMs. We note the inclusion of a correlated "configuration model" (CM), which is a special case of the degree-corrected SBM from Section 6.3.3 when there is only one block. (Alternatively, one can think of correlated CMs as extensions of correlated ER models that incorporate degree correction.) We use all of these models for edge prediction in synthetic networks in Section 6.5.2 and in consumer–product networks in Section 6.6.2. In particular, the two single-layer models are baselines that we hope to outperform using our correlated models.

6.5.2 Tests on Synthetic Networks

We use K-fold[6] cross-validation to assess the performance of the models from Table 6.2 on the edge-prediction task. In machine learning, this is a robust way to measure predictive performance [92]. After partitioning a given data set into K parts, one then fits a model to $K - 1$ of these subsets and uses it to make predictions on the leftover (i.e., "holdout") set, repeating the process K times (using each subset once as a holdout). For our problem, we perform 5-fold cross-validation (which is standard in the machine-learning literature) by splitting the data in the second layer of a given network into 5 subsets. Effectively, this consists of hiding 20% of the entries of the adjacency matrix $\boldsymbol{A}^2$, so that we do not know whether they are edges or not. We then train a model on 100% of the entries of $\boldsymbol{A}^1$ and 80% of the entries of $\boldsymbol{A}^2$ and use it to make predictions about the 20% holdout data, repeating the process 5 times.

A common way to assess the performance of a binary classification model (i.e., a model that assigns one of two possible values to test data) is using a so-called receiver operating characteristic (ROC) curve. An ROC curve plots the true-positive rate (TPR) of a classifier against the false-positive rate (FPR) for various choices of a threshold. Many models — including those used for edge prediction in networks — make probabilistic predictions, so specifying a threshold is necessary to convert these into binary predictions. Lowering the threshold increases both the TPR and the FPR. A model has predictive power if the former grows faster than the latter, such that the entire ROC curve lies above the diagonal line TPR = FPR, which gives the performance of a random classifier. (See Figure 6.4 for some examples.) As a single summary measure of a model's predictive performance, it is common to report the area under an ROC curve (AUC). Larger AUC values are better, with AUC = 0.5 being equally successful as random guessing and AUC = 1 corresponding to perfect prediction. (In the latter case, note that one still needs to determine a choice of threshold that completely separates true positives from false positives.)

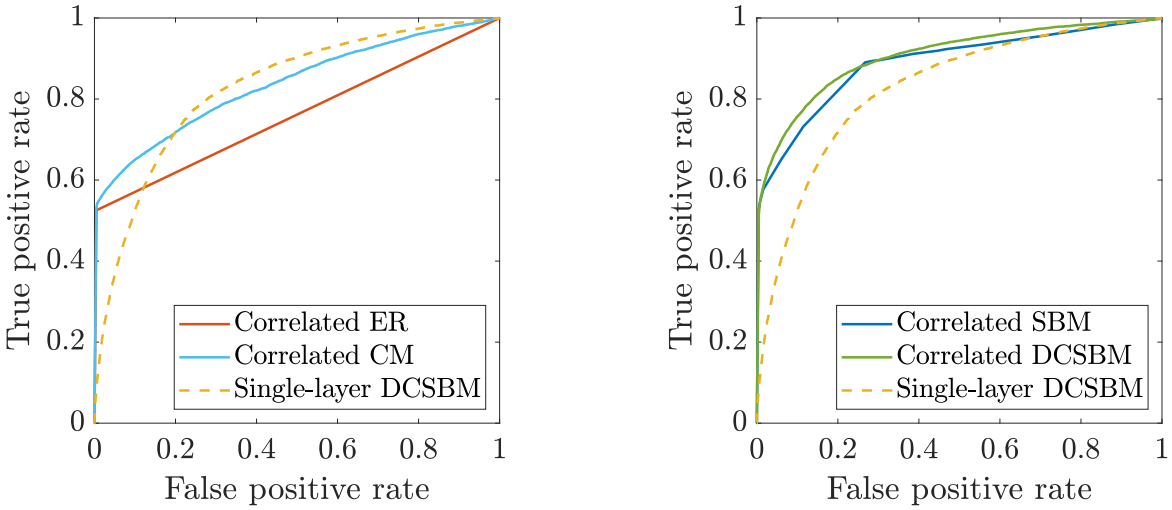
We now describe how to generate synthetic networks that are suitable for testing the models in Table 6.2. We construct these networks so that they have two tunable parameters: the correlation $\rho \in [-1, 1]$ and the community-mixing parameter $\mu \in [0, 1]$, which controls the strength of the planted mesoscale structure. (See Bazzi et al. [17] for more details about the definition of μ.) One can also control the degree distribution explicitly, such as by including a third parameter τ for the slope of a truncated power law (e.g., as used in [17] to sample a degree sequence in each layer).

[6]Note that this K is different from the one that denotes the number of blocks elsewhere in this thesis.

For the experiments in this section, we fix $\tau = -2$ and set $k_{\min} = 10$ and $k_{\max} = 50$ for the minimum and maximum degrees, respectively. It would be interesting to explore the performance gap between degree-corrected and non-degree-corrected models as one varies τ, $k_{\min}$, and $k_{\max}$, although we do not investigate this in the present work. Lastly, for the numerical experiments in this section, we set $N = 2000$ for the number of nodes in each layer and $n_c = 5$ for the number of communities, with community sizes sampled from a flat Dirichlet distribution (i.e., one with $q = 1$ and $\theta = 1$ in the notation of [17]).

We propose two versions, which we call CorrSBM and CorrDCSBM, of a correlated benchmark parametrised by ρ and μ. For both versions, the (undirected and unipartite) adjacency matrix $\boldsymbol{A}^1$ of the first layer is generated in the same way. Specifically, given μ and degree-distribution parameters τ, $k_{\min}$, and $k_{\max}$, we use the code provided in [72] to generate $\boldsymbol{A}^1$ and its associated block structure $\boldsymbol{g}$. We fit a single-layer model to $\boldsymbol{A}^1$ — either an SBM or a DCSBM, depending on the selected version of the benchmark — to obtain the marginal edge propensities $\boldsymbol{p}^1$ for the first layer. If $\rho \geq 0$, we then set $\boldsymbol{p}^2 = \boldsymbol{p}^1$; if $\rho \leq 0$, we set $\boldsymbol{p}^2 = \boldsymbol{J} - \boldsymbol{p}^1$, where $\boldsymbol{J}$ is an all-ones matrix of the appropriate size. This choice ensures that we can generate networks when $\rho = 1$ or when $\rho = -1$. Given $\boldsymbol{p}^1$, $\boldsymbol{p}^2$, and ρ, we then determine $\boldsymbol{q}$ using either the correlated SBM of Section 6.3.1 or the correlated DCSBM of Section 6.3.3. For the CorrDCSBM benchmark with $\rho \geq 0$, we set the normalised degrees θ_i^2 to be equal to the corresponding quantities θ_i^1 from the first layer. Again, this choice ensures that we can generate networks all the way to $\rho = 1$. (We also implemented a version of this benchmark that samples degrees independently in the second layer, and we found qualitatively similar results.) The final step consists of generating $\boldsymbol{A}^2$ given $\boldsymbol{A}^1$, parameter values $\boldsymbol{p}^2$ and $\boldsymbol{q}$, and — for the CorrDCSBM benchmark only — the normalised degree sequences $\boldsymbol{\theta}^1$ and $\boldsymbol{\theta}^2$ for both layers. To do this, one first computes edge probabilities using either of the correlated models from Sections 6.3.1 and 6.3.3, and one then generates edges according to these probabilities.

We now present results for these two benchmarks. In Figure 6.4, we show sample ROC curves using various models for edge prediction in one network sampled from the CorrDCSBM benchmark with $\mu = 0.3$ and $\rho = 0.5$. We compare the performance of our correlated models against a single-layer DCSBM baseline, which performs edge prediction using only information from the second network layer. Two of the correlated models outperform this baseline, and the correlated CM performs equally well (i.e., it has similar AUC).



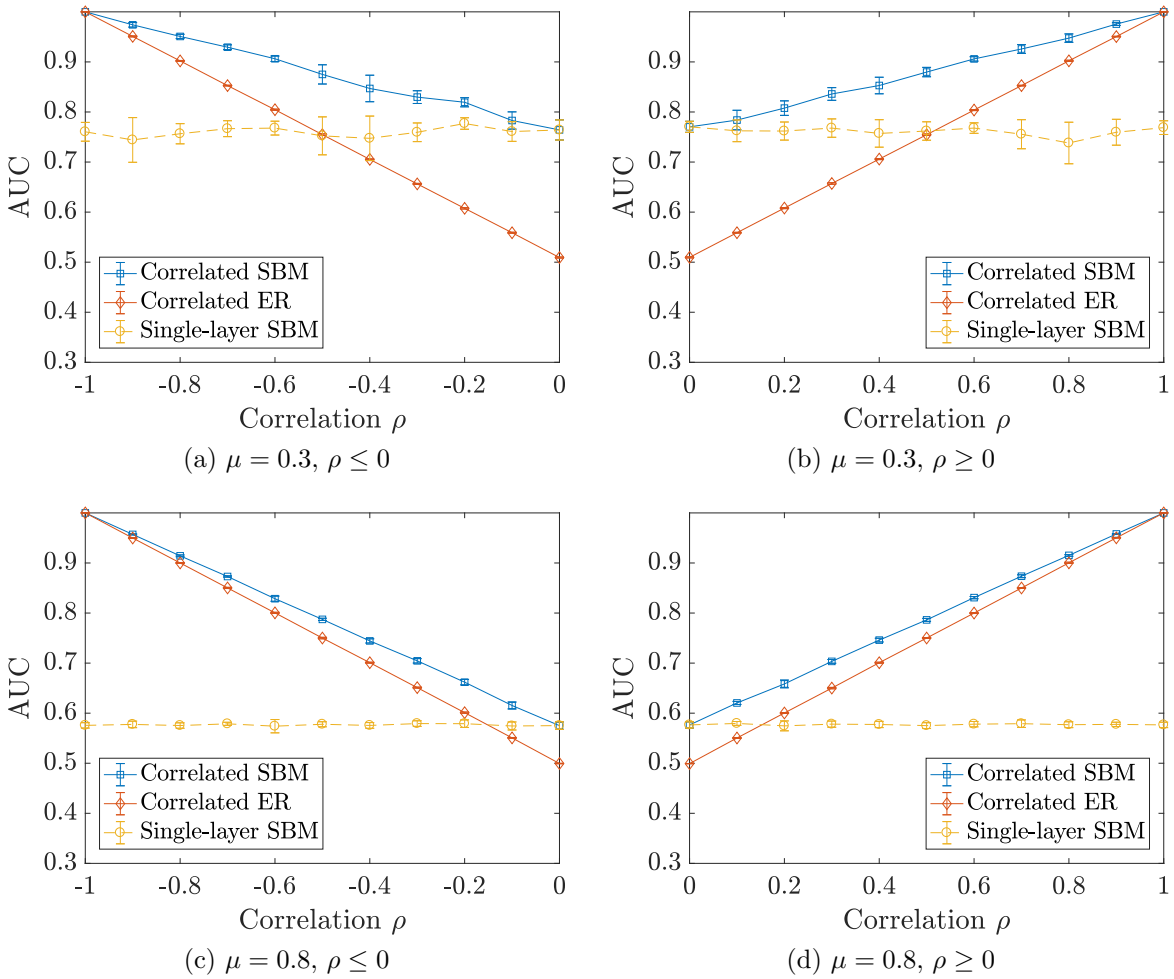
(a) Models without mesoscale structure

(b) Models with mesoscale structure

Figure 6.4: ROC curves from 5-fold cross-validation for a network sampled from the CorrDCSBM benchmark with $\mu = 0.3$ and $\rho = 0.5$. (a) Correlated models that do not incorporate any mesoscale structure, compared to a single-layer DCSBM baseline (AUC ≈ 0.83). The AUC values for the two correlated models are approximately 0.76 (correlated ER) and 0.83 (correlated CM). (b) Correlated models that incorporate mesoscale structure, compared to a single-layer DCSBM baseline (AUC ≈ 0.83). The AUC values for the two correlated models are approximately 0.89 (correlated SBM) and 0.91 (correlated DCSBM).

In Figure 6.5, we show results for the CorrSBM benchmark for two choices of the mixing parameter μ and several correlation values ρ (both positive and negative). As expected, the AUC values for single-layer SBMs are independent of ρ, whereas the predictive performance of correlated ER models and correlated SBMs improves as $|\rho|$ increases. In particular, when $|\rho| = 1$, the two correlated models make perfect predictions. When $\rho = 0$, the performance of the correlated ER model is indistinguishable from chance (AUC ≈ 0.5), whereas correlated SBMs perform equally well to single-layer SBMs. The gap between the two correlated models is smaller when μ is larger, as the underlying block structure is weaker. (Note additionally that the AUC of the single-layer baseline is itself smaller when μ is larger.)

One striking feature in Figure 6.5 is that all curves appear to be straight lines (to within sampling error). It makes sense that the performance of single-layer SBMs does not vary with ρ, as these models do not use any information from the other layer, but the linear dependence on ρ of the other two curves is less obvious. For the correlated ER model, we can establish the following result:



(a) $\mu = 0.3$, $\rho \leq 0$ (b) $\mu = 0.3$, $\rho \geq 0$

(c) $\mu = 0.8$, $\rho \leq 0$ (d) $\mu = 0.8$, $\rho \geq 0$

Figure 6.5: Edge-prediction results on synthetic networks from the CorrSBM benchmark with $\rho \leq 0$ (left column) and $\rho \geq 0$ (right column), using two choices of the mixing parameter μ. We use $\mu = 0.3$ in the top row, and $\mu = 0.8$ in the bottom row. In all plots, we vary the correlation ρ used to generate network instances along the horizontal axis. On the vertical axis, we indicate the AUC for 5-fold cross-validation using a single-layer SBM (dashed lines) or a correlated SBM or ER model (solid lines). Each data point is a mean across 10 trials, and the error bars correspond to one standard deviation from that mean. As expected, the AUC does not change with ρ for the single-layer model, but it increases with $|\rho|$ for the two correlated models. When μ is larger, so that the sampled networks have weaker mesoscale structure, there is a smaller performance gap between correlated ER models and correlated SBMs.

Proposition. *The AUC for a correlated ER model is an affine function of ρ. In particular, when $p_1 \approx p_2$ or $p_1 \approx 1 - p_2$, we have $AUC_{\text{ER}} \approx (1 + |\rho|)/2$.*

Proof. Suppose that $\rho > 0$; the case $\rho \leq 0$ is similar. With a correlated ER model, all unobserved interactions (i, j) in the second layer are assigned one of two probabilities:

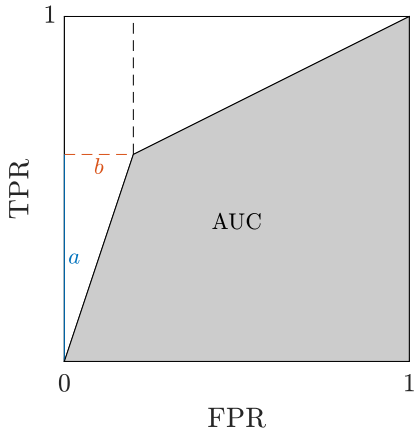


Figure 6.6: Diagram of the ROC curve for an ER model, which assigns to every pair of nodes one of two possible edge probabilities.

q/p_1 if $A^1_{ij} = 1$ and $(p_2 - q)/(1 - p_1)$ if $A^1_{ij} = 0$. Because $\rho > 0$, we have $q > p_1 p_2$, which implies that $q/p_1 > (p_2 - q)/(1 - p_1)$. Selecting a threshold between these two probabilities amounts to predicting that everything that is an edge in the first layer is also an edge in the second layer, and that everything that is not an edge in the first layer is also not an edge in the second layer. Let a and b denote the TPR and FPR, respectively, at such an intermediate threshold. In this case, a and b are the coordinates of the inflection point of the ROC curve from Figure 6.6. By straightforward geometry,

$$\mathrm{AUC}_{\mathrm{ER}} = 1 - \frac{ab}{2} - \frac{(1-a)(1-b)}{2} - (1-a)b = \frac{1}{2} + \frac{a-b}{2}$$

is the area under this ROC curve. The next step is to estimate a and b. Because of how the correlated ER model is set up, the number of true positives is proportional[7] to e_{11}: the model's prediction is correct every time that an edge that is present in the first layer is also present in the second layer. To find the TPR, one needs to divide this quantity by the number of edges in the second layer, which is equal to $e_{11} + e_{01}$. Therefore,

$$a \approx \frac{e_{11}}{e_{11} + e_{01}} \approx \frac{q}{p_2} = p_1 + \rho\sqrt{\frac{1-p_2}{p_2} p_1(1-p_1)}\,. \tag{6.69}$$

Similarly, it counts as an incorrect guess by the model every time an edge that is present in the first layer is not present in the second. Thus, the number of false positives is proportional to e_{10}. Dividing this by the number of non-edges in the

[7]It is not exactly equal because with K-fold cross validation, we are only performing prediction on $1/K$ of the data.

second layer, we obtain

$$b \approx \frac{e_{10}}{e_{10} + e_{00}} \approx \frac{p_1 - q}{1 - p_2} = p_1 - \rho\sqrt{\frac{p_2}{1 - p_2} p_1(1 - p_1)}\,. \tag{6.70}$$

From (6.69) and (6.70), it follows that

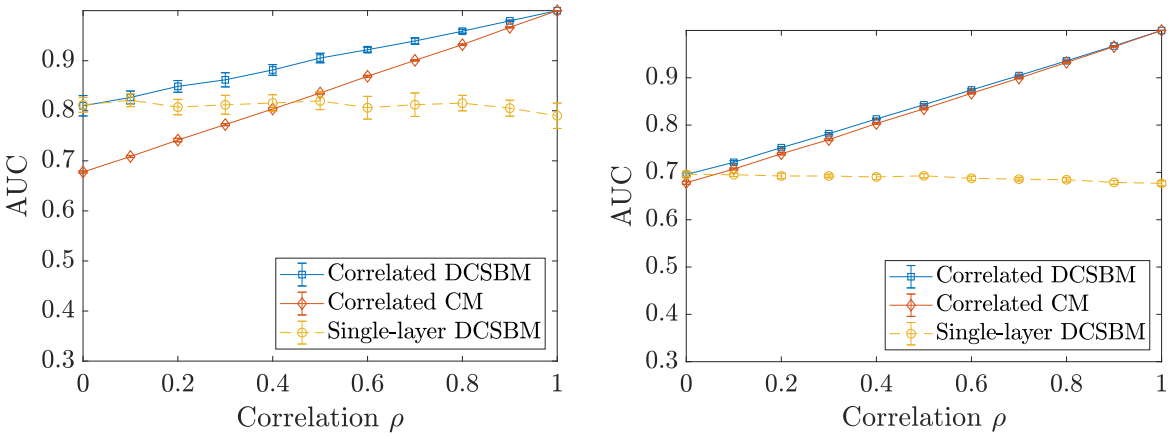
$$\mathrm{AUC}_{\mathrm{ER}} \approx \frac{1}{2} + \frac{\rho}{2}\sqrt{p_1(1 - p_1)}\left(\sqrt{\frac{1 - p_2}{p_2}} + \sqrt{\frac{p_2}{1 - p_2}}\right),$$

which is an affine function of ρ. When $p_1 \approx p_2$ or $p_1 \approx 1 - p_2$ — as is the case in Figure 6.4 — we get $\mathrm{AUC}_{\mathrm{ER}} \approx (1 + |\rho|)/2$, as desired. Using a similar argument, one can show that the same result holds (with the same assumptions on p_1 and p_2) for the case with $\rho \leq 0$. □

Given that the correlated SBM curves from Figure 6.5 also exhibit linear dependence on ρ, we believe that it is possible to establish similar results for correlated models that incorporate mesoscale structure. These results have practical importance, as they allow one to quickly estimate the benefits of switching to correlated models for edge prediction.

In Figure 6.7, we show systematic results for the CorrDCSBM benchmark for two choices of the mixing parameter μ and non-negative[8] correlation values ρ. As expected, when $\rho = 0$, correlated DCSBMs perform similarly to single-layer DCSBMs. As in Figure 6.5, the performance of the single-layer model is roughly independent of ρ, whereas the two correlated models do better as ρ increases. The gap between the two correlated models narrows substantially when increasing μ from 0.3 to 0.8.

[8]The case $\rho \leq 0$ requires a different way of sampling normalised degrees θ_i^2 of nodes in the second layer to be able to take ρ all the way to $\rho = -1$. More precisely, when ρ is close to -1, the second layer is dense if the first layer is sparse (and vice versa), so setting $\boldsymbol{\theta}^1 = \boldsymbol{\theta}^2$ is inadequate.



(a) $\mu = 0.3$ (b) $\mu = 0.8$

Figure 6.7: Edge-prediction results on synthetic networks from the CorrDCSBM benchmark with $\rho \geq 0$ using (a) $\mu = 0.3$ and (b) $\mu = 0.8$. In both panels, we vary the correlation ρ that we use to generate network instances along the horizontal axis. On the vertical axis, we indicate the AUC for 5-fold cross-validation using a single-layer DCSBM (dashed lines) or a correlated DCSBM or CM (solid lines). Each data point is a mean across 10 trials, and the error bars correspond to one standard deviation from that mean. As expected, the AUC is roughly independent of ρ for the single-layer model, but it increases with ρ for the two correlated models. When μ is larger, so that the sampled networks have weaker mesoscale structure, there is a substantial narrowing of the performance gap between correlated CMs and correlated DCSBMs.

6.6 Applications to Shopping Data

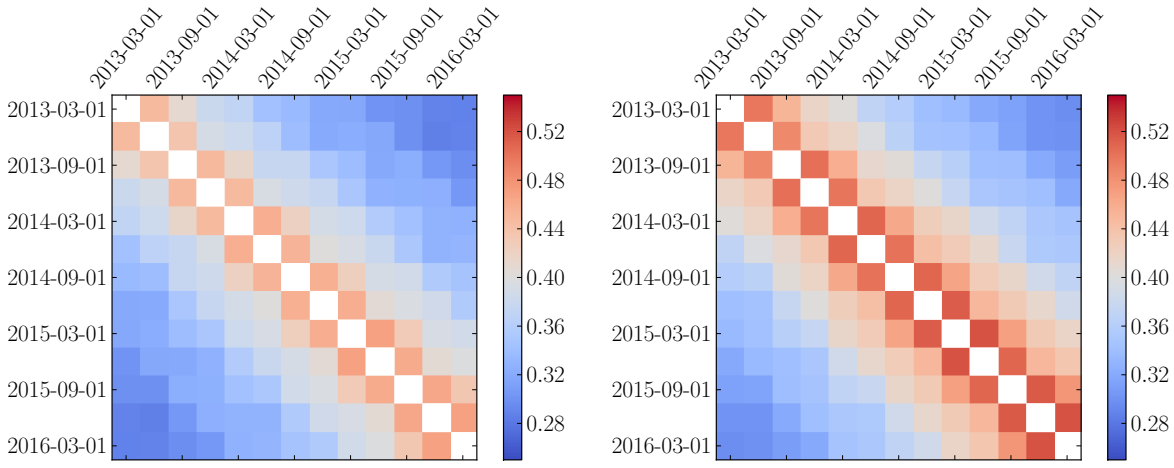
6.6.1 Correlation Calculations

We now use some of the models from Sections 6.2–6.4 (which we summarised in Table 6.1) to examine correlations between different time windows of a temporal consumer–product network. The network consists of a sequence of non-overlapping 3-month time windows of purchases from store `S2` (see Section 3.2.3) between March 2013 and June 2016. In each layer, we use item-penetration weighting on the edges, calculated as described in Section 3.2.2. This defines a weighted adjacency matrix $\boldsymbol{W}^t$ in each temporal layer t. Because our proposed edge-correlated models work on unweighted networks, we convert each matrix $\boldsymbol{W}^t$ to an unweighted adjacency matrix $\boldsymbol{A}^t$ using a percentile threshold. For example, if this threshold is set at the 50^{th} percentile, we keep half of the original edges (those with the largest weights) when constructing the unweighted network.

We first use formulas (6.13) and (6.26), which we derived for Bernoulli and Poisson ER networks, respectively, to estimate pairwise layer correlations in our shopping

network. As a reminder, the correlation estimate (6.13) matches the expression of the effective correlation that we derived for correlated Bernoulli SBMs in Section 6.3.1. Additionally, this expression also matches, to first order, the correlation estimate that we obtained for correlated degree-corrected Bernoulli SBMs in Section 6.3.3. Thus, we expect the simple correlation estimates derived for correlated ER networks to be meaningful even for networks (such as our shopping networks) that are far from being correlated ER graphs.

Computing pairwise correlations between different layers (i.e., between different 3-month time windows) of a temporal shopping network produces a correlation matrix. In Figure 6.8, we show two such correlation matrices, corresponding to two different choices of threshold for generating unweighted networks from the initial weighted network. A higher threshold tends to increase correlations, especially those near the main diagonal, that correspond to layers that are closer in time. This is not surprising, as using a threshold to discard lower-weight edges tends to remove purchases that are less likely to be persistent shopping preferences and more likely to be one-off buys.



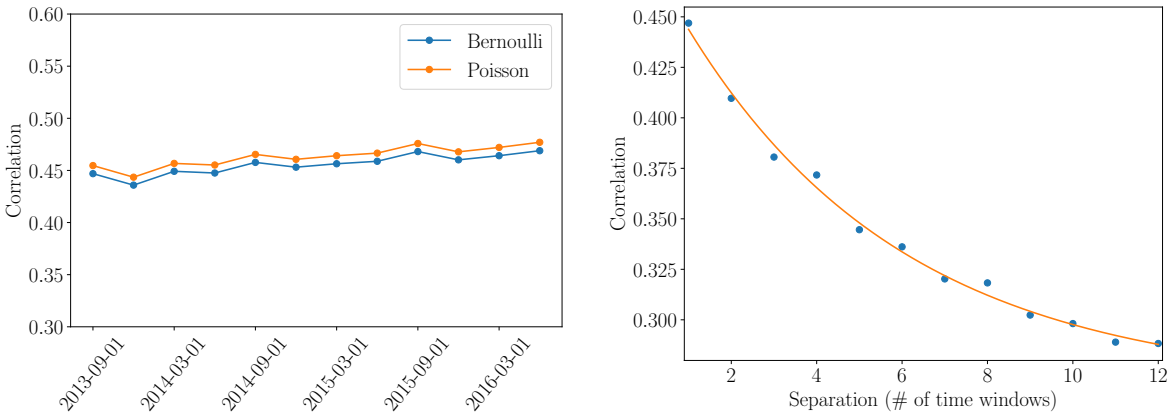
(a) Threshold at 0^{th} percentile (b) Threshold at 50^{th} percentile

Figure 6.8: Correlations between layers of a temporal shopping network using (a) a 0^{th}-percentile threshold and (b) a 50^{th}-percentile threshold to convert the initial weighted network to an unweighted network. The higher threshold yields larger correlations, especially close to the diagonal.

We also use the shopping data to compare the effects of using Bernoulli or Poisson random variables to model edges. In Figure 6.9(a), we plot the correlation between successive layers over time. (This corresponds to the first upper diagonal of the correlation matrix in Figure 6.8(a).) The differences between these two models are minimal, with the Poisson one giving slightly higher correlation values. We also

conclude from this plot that the correlation between successive layers is relatively consistent over time.

In Figure 6.9(b), we show how correlation decays as we increase the time separation between two time windows. (This corresponds to the first row of the correlation matrix in Figure 6.8(a).) The orange curve gives an exponential fit of the form $f(x) = ae^{-bx}+c$ to the observed data points. The fitted value for the additive constant, $c \approx 0.27$, suggests a baseline level of correlation that persists even in the limit in which the time separation between layers becomes infinite. This could be due to products that certain customers buy consistently, such as specific kinds of milk, bananas, eggs, and other frequently-purchased items.



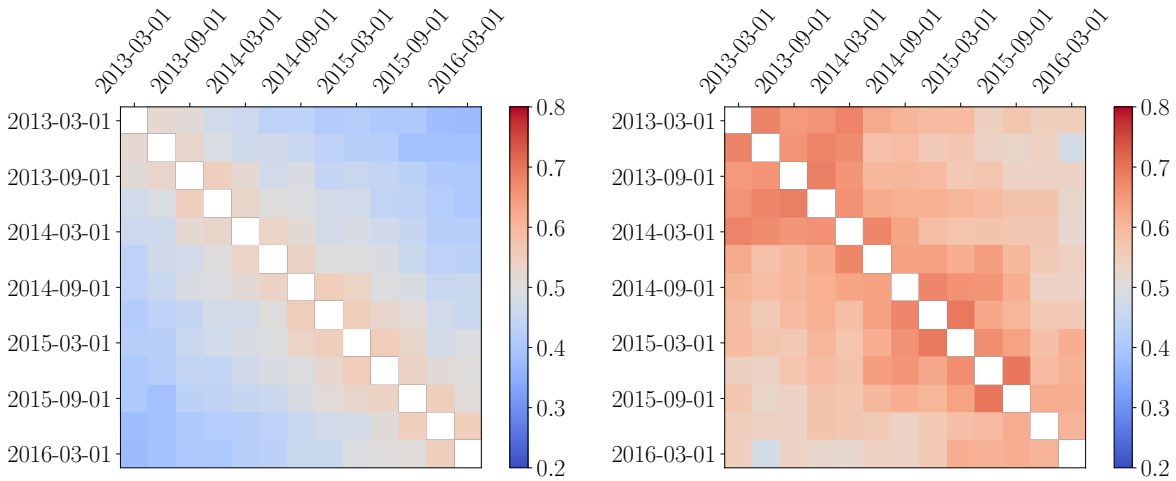
(a) Correlation over time

(b) Correlation decay

Figure 6.9: Correlation patterns in a temporal consumer–product network. (a) The mean Pearson correlation between consecutive time windows is consistent over time. There are small differences between using Bernoulli and Poisson random variables to generate edges, although the amount of deviation does not seem to depend on the absolute level of correlation. (b) The Pearson correlation decreases exponentially with time separation between snapshots. Even at a separation of 3 years (i.e., 12 time windows), the correlation is still almost 0.3.

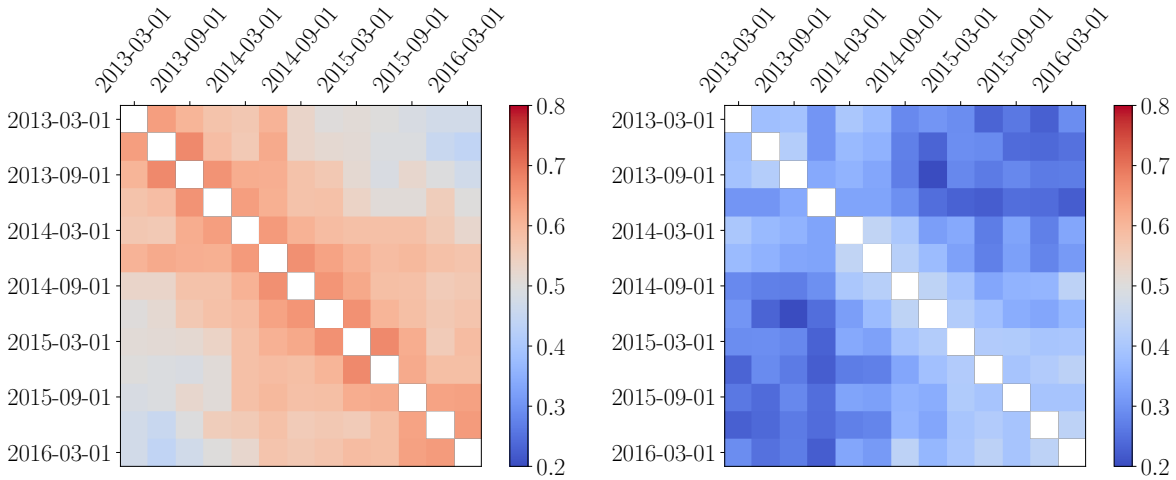
Moving away from single measurements, we calculate correlations for each edge bundle (r, s) using the non-degree-corrected, Bernoulli SBM from Section 6.3.1. (Using the Poisson version of this model gives similar results.) To calculate ML estimates $\widehat{\rho}_{rs}$ of the correlation values, we have to fix a block partition $\boldsymbol{g}$. We use the partition resulting from one modularity-maximisation run on the weighted temporal network with $\gamma = 1.2$ and $\omega = 5.0$. The large value of the interlayer coupling ω ensures that communities are nearly identical across all layers. We then use the community labels from the first layer in all subsequent layers to ensure that the partition is strictly the

same for all time windows. This produces a partition with $K = 33$ communities. Note that, although modularity detects mixed communities of customers and products, we can easily separate these into $K_1 = 33$ customer blocks and $K_2 = 33$ product blocks to fit the framework of our statistical models (see our discussion in Section 2.4.2).



(a) Community 1 (83 customers, 288 products) (b) Community 6 (30 customers, 90 products)

Figure 6.10: Different communities exhibit very different intra-community correlation patterns, as we illustrate here for 2 of the 33 communities. Smaller communities tend to have larger correlations across all pair of layers.



(a) Intra-community correlation (b) Inter-community correlation

Figure 6.11: Intra-community correlations are generally larger than inter-community correlations. (a) Correlation matrix for $(r, s) = (2, 2)$. (b) Correlation matrix for $(r, s) = (2, 3)$.

For each edge bundle (r, s), we can calculate a correlation matrix similar to those

from Figure 6.8. When $r = s$, these are intra-community correlations; when $r \neq s$, these are inter-community correlations. In Figure 6.10, we show the intra-community correlation matrices for two of the 33 communities. These plots illustrate that different communities can exhibit very different correlation patterns. We also find that intra-community correlations tend to be larger than inter-community correlations, as illustrated in Figure 6.11. This suggests that persistent communities — like the ones that we detected here using modularity maximisation with a large ω value — are characterised not only by the property of assortativity (i.e., nodes are more likely to form connections within their own community), but also by the persistence of edges.

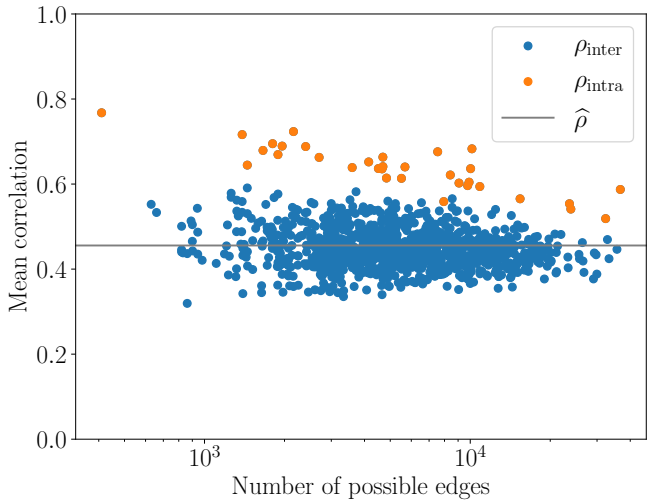


Figure 6.12: Scatter plot of intra-community correlations ρ_{intra} and inter-community correlations ρ_{inter} for all $33 + \binom{33}{2} = 561$ distinct edge bundles. We calculate these correlations between successive time periods, and we then average them over the entire 3-year history in the data set. The grey line indicates a mean of the effective correlation values of each consecutive pair of layers, calculated using equation (6.36).

In Figure 6.12, we explore these patterns more systematically by calculating, for each edge bundle (r, s), the mean correlation between consecutive layers. Each of these values is a point in the resulting scatter plot, with intra-community correlations in orange and inter-community correlations in blue. As we noted before, the former correlations are systematically larger than the latter. We also observe that intra-community correlations decay with the number of potential edges (i.e., the number of customer nodes multiplied by the number of product nodes) in their community. This suggests that smaller communities often have "higher quality", which is consistent with previous observations [71]. We observe no such dependence for inter-community correlation values.

The business value of such an analysis is that it illustrates which products consumers tend to buy consistently and which product types they tend to buy more haphazardly (or seasonally). Incorporating mesoscale structure when calculating these correlations, as we have done here with correlated SBMs, provides insights that are both less detailed and more accurate (because they aggregate multiple pieces of data) than computing correlations for individual products.

6.6.2 Edge Prediction

We now apply the edge-prediction methodology from Section 6.5 to shopping networks. As with our tests on synthetic networks from Section 6.5.2, we use 5-fold cross-validation and the resulting AUC values to assess predictive performance. For our data, an alternative way to evaluate the quality of these predictions is to send promotional coupons to customers and assess whether the probabilities computed using one of our edge-prediction models are systematically larger for customers who redeem the coupons than for customers who do not. (This is similar to the approach that we took in Section 3.3.2 using data from an existing promotional campaign.) However, obtaining this kind of data for a large number of customers and products is slow, costly, and subject to various business constraints. Using cross-validation is therefore more practical for an initial model evaluation.

Table 6.3: Predictive performance of different models, as measured by the AUC.

Model	AUC (ShoppingMod)	AUC (ShoppingSBM)
Single-layer SBM	0.549	0.633
Correlated ER	0.724	0.743
Correlated SBM	0.742	0.793
Single-layer DCSBM	0.725	0.797
Correlated CM	0.817	0.870
Correlated DCSBM	0.818	0.875

We construct temporal networks with two layers covering the three-month time periods of March–May 2013 and June–August 2013, respectively. Using the same underlying transaction data, we construct two networks that we then analyse further. For the first network, we use basket-penetration weights on the edges (see Section 3.2.2) and apply multilayer modularity maximisation to the weighted network (setting $\gamma = 1.2$ and $\omega = 5.0$) to determine the community structure $\boldsymbol{g}$, which we fix throughout the rest of the analysis. For the second network, we initially calculate item-penetration

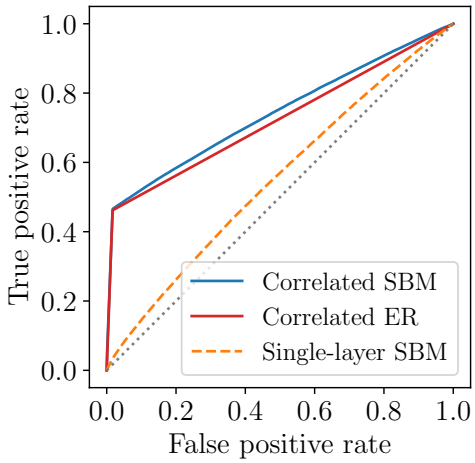
weights, and then we apply a threshold to remove 50% of the edges. We fit a degree-corrected SBM to the resulting unweighted network using the belief-propagation algorithm that we described in Section 4.2.1 (but without any annotations) to infer $\boldsymbol{g}$. These two network representations of the same underlying data thus differ in a few key ways; an important one is that we use different methods to infer their mesoscale structure. In particular, we expect better edge-prediction performance on the second network, because we detected its block structure using an SBM (as opposed to using modularity maximisation, which is more restrictive). For convenience, we refer to the two networks as ShoppingMod and ShoppingSBM.

We summarise the performance of our various correlated Bernoulli models and the single-layer baselines in Table 6.3, and we show sample ROC curves in Figure 6.13. We make a few observations about these results. First, we note that our approximation AUC $\approx (1+\rho)/2$ for correlated ER models is very precise for these two networks, whose correlations are approximately 0.44 and 0.48, respectively. Second, our correlated models outperform the single-layer baselines for both networks. In particular, the very simple correlated ER model — which only ever assigns one of two probabilities to edges, as indicated in Table 6.2 — performs about as well as the more sophisticated single-layer DCSBM for the ShoppingMod network. Third, as expected, AUC values are systematically larger for ShoppingSBM than for ShoppingMod. Finally, although incorporating mesoscale structure leads to better performance when there is no degree correction, this does not seem to be the case for degree-corrected models: correlated DCSBMs do not perform significantly better than correlated CMs. This is also apparent in Figures 6.13(b) and 6.13(d), where the ROC curves for the two models are virtually identical. This result suggests that for some networks, taking into account layer correlations and degree heterogeneity alleviates the need to also consider mesoscale structure when performing edge prediction. This observation has practical implications, as a correlated CM is much easier than a correlated DCSBM to fit to data and to use for edge prediction.[9]

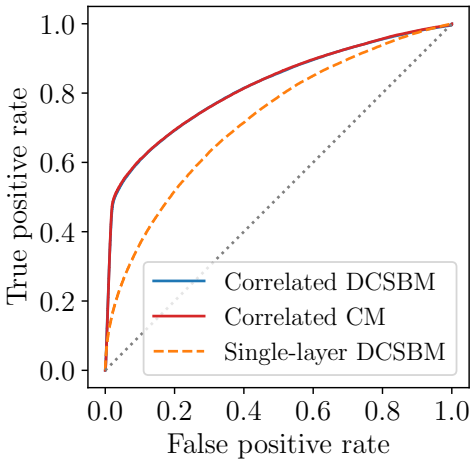
6.7 Summary and Discussion

In this chapter, we introduced models of multilayer networks in which edges that connect the same nodes in different layers are not independent (in contrast to previous

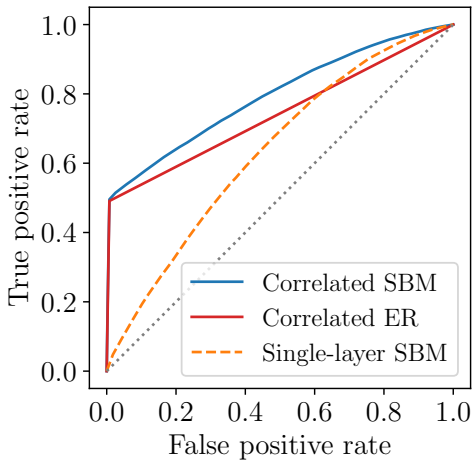
[9]For recommender systems, there are situations in which fitting a correlated DCSBM is beneficial, even if its edge-prediction performance is similar to that of a correlated CM. For instance, one may wish to identify relevant customers for a chosen product irrespective of how much they buy (i.e., their degree). SBMs are able to distinguish between customers with equal degrees and identify those with the highest predisposition for buying a particular product, whereas CMs are not.



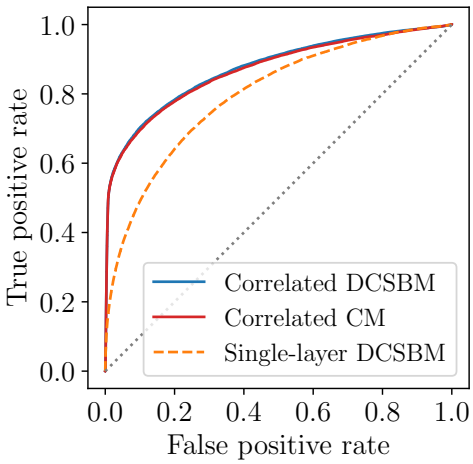
(a) ShoppingMod, no degree correction



(b) ShoppingMod, degree-corrected models



(c) ShoppingSBM, no degree correction



(d) ShoppingSBM, degree-corrected models

Figure 6.13: ROC curves for the edge-prediction task using 5-fold cross-validation on two temporal networks, ShoppingMod (top row) and ShoppingSBM (bottom row). We consider both models without degree correction (left column) and models with degree correction (right column). The dotted diagonal line in each plot indicates the expected ROC curve for a random classifier. All other curves lie above this diagonal line, suggesting that they have predictive power. In all cases, correlated models outperform their single-layer counterparts. For both networks, correlated SBMs outperform correlated ER models, whereas correlated DCSBMs perform similarly to correlated CMs (as illustrated by the nearly overlapping curves in panels (b) and (d)).

work). These models offer an improved representation of many empirical networks in which, for example, flights between major airports are serviced by multiple airlines, individuals interact repeatedly with the same people, and consumers often buy the same products over time. Among other potential applications, our models can be used to improve edge prediction, to study the graph-matching problem on more

realistic benchmark networks, to detect correlated communities, and to calculate layer correlations as new summary statistics for networks.

Throughout this chapter, we have considered both bivariate Bernoulli and bivariate Poisson random variables for generating edges simultaneously in two network layers. The prevailing view [122, 171] is that Poisson models simplify calculations for single-layer degree-corrected SBMs, but we have found that Bernoulli models are simpler when incorporating correlations. Bernoulli models have several other advantages. These include the fact that they work for both sparse and dense networks and that they can handle the entire correlation range $\rho \in [-1, 1]$. Bernoulli models have the additional benefit that defining an effective correlation measure for correlated SBMs recovers the correlation value when there is no block structure (i.e., for correlated ER graphs), which is not the case for Poisson models.

Correlated Bernoulli SBMs have been proposed previously [96], although our derivation of ML parameter estimates is novel. Two additional key contributions are extensions of this model that incorporate degree correction and that allow for non-identical block structure across layers. The ML equations are significantly more difficult to solve for degree-corrected models than for models without degree correction, but we were able to make some simplifications using suitable approximations. Notably, these simplified equations closely approximate those for non-degree-corrected models for networks with homogeneous degree distributions.

An interesting application of our work is to edge prediction in multilayer networks. Our numerical experiments reveal that simple correlated models (e.g., a correlated configuration model or a correlated SBM without degree correction) can outperform single-layer DCSBMs even for moderate correlation values. We found that this improved performance is also true for consumer–product networks, which exhibit significant layer correlations ($\rho \approx 0.45$). Indeed, we expect that a correlated DCSBM will outperform a single-layer DCSBM for most empirical networks, even for lower levels of correlation.

The models in this chapter that incorporate some mesoscale structure $\boldsymbol{g}$ assume this structure to be given. This setup has the benefit that one can use any desired algorithms to produce a network partition, including ones that operate on weighted or annotated networks or that use non-standard null models in defining a modularity function. This makes our approach for analysing correlations suitable for a wide variety of applications.

For consumer-behaviour applications, the most obvious use of our work is for personalised recommendations. Our approach is particularly appropriate in cases

in which a consumer interacts repeatedly with the same item — e.g., by regularly purchasing the same grocery product or by listening to a song several times. Using correlated models for recommendations is less suitable when a previously purchased item is not a desirable candidate for a recommendation.

There are many interesting ways to build further on our work. For example, it would be useful to be able to model all layers simultaneously rather than in a pairwise fashion, especially for multiplex networks, in which the layers do not have a natural ordering. One challenge is that a multivariate Bernoulli distribution of dimension L has $2^L - 1$ parameters, which grows quickly with the number L of layers. For the temporal setting, we have proposed generating correlated networks in a sequential way, by conditioning each layer on the previous one. In some cases, it would be useful to relax this "memoryless" assumption and condition a layer on all the ones that came before it, rather than only on the most recent layer. For example, consumers' purchases in December one year are strongly related not just with their purchases in November, but also to what they bought in December the previous year.

In this work, we only briefly considered the case of non-identical mesoscale structure across layers. In particular, we have not introduced degree correction in this model, and we have not used it for edge prediction (both of which are natural next steps). Yet another idea is to derive correlated models for networks with overlapping communities. Previous research [34] suggests that this can substantially improve edge-prediction performance. Lastly, our work is a starting point for designing an algorithm for detecting correlated communities in networks, by inferring $\boldsymbol{g}$ alongside the other model parameters.

Chapter 7

Conclusions

The analysis and computation of mesoscale structures in networks is one of the most important topics in network science, with a vast literature dedicated to new algorithms, theoretical analyses (e.g., of detectability thresholds), and applications across a variety of domains [107]. Earlier work on the subject focused on the topic of community detection [46, 48, 124], which involves partitioning a network into sets of nodes that are densely connected internally and sparsely connected with other dense sets of nodes. More recently, in recognition that many networks exhibit other forms of mesoscale organisation beyond assortative communities, there has been an increased focus on methods such as SBM inference that can detect these more general types of structure [1, 118].

In this thesis, we examined mesoscale structures in networks with various complications — specifically, annotated, multilayer, and correlated networks — using modularity maximisation and SBM inference. Although each topic was motivated by applications to bipartite consumer–product networks, the methodology that we have developed applies more broadly. We summarise our contributions in these three areas in turn; see also the summary sections of Chapters 4–6 for more in-depth discussion of our findings and of potential ideas for future work.

In Chapter 4, we studied networks that incorporate additional data in the form of node labels. These annotations influence the outputs of algorithms that infer block structure in a given network, making partitions that align with the metadata more likely in the posterior distribution. On the methodological side, our main contribution was extending previously-proposed methods for inference in annotated networks [109] and in weighted networks [3] to a bipartite setting. We also modified the Bayesian approach from [129] for inferring the number of blocks to handle bipartite networks. We then applied these algorithms to consumer–product networks, in which we used

product categories as annotations on the product nodes. Our main finding for this application was that it is more difficult to assign product nodes than customer nodes with high certainty to a single block. However, it appears that incorporating annotations helps reduce this uncertainty relative to a network that does not have this metadata. A promising use of the methods of this chapter to a consumer-behaviour application is to address the cold-start problem of recommender systems. Using parameters learned through annotated-networks inference, one can provide introductory offers to customers who are most likely to buy a newly-launched product, despite there being no historical purchasing data for that particular product. For future work, it would be useful to develop algorithms that work reliably on weighted annotated networks, possibly with more than one type of metadata (e.g., also incorporating price information in consumer–product networks).

In Chapter 5, we studied the problem of community detection in multilayer networks. We showed that maximising a multilayer generalisation of the modularity function is equivalent, under certain conditions, to maximising the posterior probability of community assignments under suitably chosen stochastic block models. We derived versions of this equivalence for various types of multilayer structure, including temporal, multiplex, and multilevel networks. We considered cases in which the key parameters are constant as well as ones in which they vary across layers; in the latter case, this yields a novel, layer-weighted version of the modularity function. Our results also help address a longstanding difficulty of multilayer modularity-maximisation algorithms, which require the specification of two sets of tuning parameters that have been difficult to choose in practice. We showed how to perform this parameter selection in a statistically-grounded way, and we applied our approach to synthetic benchmarks and to both temporal and multilevel shopping networks. In future work, we believe that it is important to improve on our iterative parameter-selection algorithm and to extend our results to weighted multilayer networks.

In Chapter 6, we proposed a way to relax a common assumption of multilayer generative network models; traditionally one samples edges in different layers independently, conditioned on the community labels of the nodes. In many empirical networks, including in consumer–product networks, we expect significant correlations between edges that connect the same nodes in multiple layers. Our work provides ways to incorporate such correlations explicitly in an SBM-like model under various scenarios (e.g., Bernoulli versus Poisson random variables for generating edges, with or without degree correction, and with identical or non-identical blocks across layers). Fitting these models to a temporal consumer–product network revealed large correlations, of

approximately 50%, between successive three-month time periods; thus, events that correspond to a customer buying a particular product in two different time periods are far from statistically independent. From a practical standpoint, the most promising application of our work is to edge prediction. Performing cross-validation on both synthetic networks and empirical shopping networks, we showed that models that account for cross-layer correlations can vastly outperform single-layer models. Topics for future research include extensions of our models to networks with overlapping communities, which would further improve edge-prediction performance, and inference algorithms for detecting correlated communities in multilayer networks.

The study of mesoscale structures in networks has many open problems. We mention four topics that we did not study in the present thesis, but that we believe will provide useful insights into the structure of consumer–product networks: (1) overlapping community detection; (2) local community detection; (3) models of noisy network data; and (4) simultaneous consideration of multiple edge weights.

Throughout this thesis, we have made the assumption that each node — customer or product — belongs to one community.[1] However, it is possible for a customer to be meaningfully associated to more than one community (e.g., a community associated to healthy eating and another associated to having young children). Similarly, popular products, such as milk and bananas, are purchased by consumers of all types. (In fact, we observed this to some extent in Chapter 4.) This motivates the detection of overlapping communities in networks [161]. Popular approaches for tackling this problem include non-negative matrix factorisation [125, 164], link clustering [43, 140], and inference in mixed-membership SBMs (both monolayer and multilayer ones) [34, 52]. We note in passing that one can typically approximate overlapping community structure using a larger number of disjoint communities [117], although this tends to reduce computational efficiency.

When using network methods for product recommendations, one is often only interested in a small sample of customers (e.g., those in a particular geographical area) or in a small selection of products (recall the promotional campaign for a particular brand of yoghurt that we analysed in Section 3.3.2). In these cases, performing community detection in a network with thousands of products and millions of customers is unnecessarily inefficient, as we are interested only in the structure of a small part of the network. Local community-detection methods, which seek to identify communities around a small set of seed nodes, provide a more computationally efficient

[1] Importantly, the probabilistic block assignments returned by the algorithms from Chapter 4 reflect the uncertainty around what is the correct community label for a particular node, rather than a notion of overlapping blocks in the underlying SBM.

alternative. Thus, if one is interested in identifying relevant customers for promotions on a particular set of products, those products would be the seed nodes of a local community-detection algorithm. Existing local approaches include some based on quality functions [9], including modularity [27], and some based on an interpretation of communities as bottlenecks to dynamical processes in networks [71, 74].

Like most empirical data sets, consumer–product networks that are constructed from transaction data are bound to contain noise. A common source of error is that of missing edges, which occur when a customer does not use their loyalty card (so the store is not able to match their current shopping trip to previous purchases) or when they buy some items in different stores. Spurious edges can also occur — for example because a customer lends their loyalty card to someone else, which results in purchases that are not accurate representations of the preferences of the original customer. Some new methods for modelling and performing inference in noisy networks [106, 120] may prove useful in detecting when these scenarios occur and for harnessing the inferred models for improved edge prediction.

In our analysis of shopping networks, we used two different weightings — item penetration and basket penetration — but there are other weightings that one can examine (e.g., based on how many times a customer buys a specific product over similar alternatives). Because there is no single best weighting for aggregating transactions into a network structure, it will be useful to develop methods that can simultaneously incorporate information from multiple weightings. One possibility is to use a multiplex representation in which network topology is identical across layers, but the weights of edges are different. Because different weightings are likely to be correlated with each other, one can model these weights as mixtures of multivariate distributions, with each edge bundle having its own parameters. We anticipate that this problem is also relevant to other applications for which multiple sensible weightings are possible.

Our work in this thesis illustrates the role of applications in motivating the development of new methodology. (See also the discussion in [123] on the role of network science in industrial applications.) For example, the presence of exogenous data in the form of product hierarchies has led us to study annotated and multilevel networks; the question of how consumer preferences evolve over time has led us to examine changing community structure in temporal networks; and the observation that consumers tend to buy the same products over time guided our development of correlated multilayer networks. In turn, these new methods can provide insights into new applications, leading to a virtuous cycle of scientific progress.

Bibliography

[1] E. Abbe. Community detection and stochastic block models: Recent developments. *Journal of Machine Learning Research*, 18(177):1–86, 2018.

[2] D. Abbott. *Applied Predictive Analytics: Principles and Techniques for the Professional Data Analyst*. John Wiley & Sons, 2014.

[3] C. Aicher, A. Z. Jacobs, and A. Clauset. Learning latent block structure in weighted networks. *Journal of Complex Networks*, 3(2):221–248, 2014.

[4] A. Amelio and A. Tagarelli. Revisiting resolution and inter-layer coupling factors in modularity for multilayer networks. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 266–273. ACM, 2017.

[5] C. J. Anderson, S. Wasserman, and K. Faust. Building stochastic blockmodels. *Social Networks*, 14(1-2):137–161, 1992.

[6] A. Arenas, J. Duch, A. Fernández, and S. Gómez. Size reduction of complex networks preserving modularity. *New Journal of Physics*, 9(6):176, 2007.

[7] A. Arenas, A. Fernandez, and S. Gomez. Analysis of the structure of complex networks at different resolution levels. *New Journal of Physics*, 10(5):053039, 2008.

[8] U. Aslak, M. Rosvall, and S. Lehmann. Constrained information flows in temporal networks reveal intermittent communities. *Physical Review E*, 97(6):062312, 2018.

[9] J. P. Bagrow. Evaluating local community methods in networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(05):P05001, 2008.

[10] J. P. Bagrow. Communities and bottlenecks: Trees and treelike networks have high modularity. *Physical Review E*, 85(6):066118, 2012.

[11] M. J. Barber. Modularity and community detection in bipartite networks. *Physical Review E*, 76(6):066102, 2007.

[12] P. Barbillon, S. Donnet, E. Lazega, and A. Bar-Hen. Stochastic block models for multiplex networks: An application to a multilevel network of researchers. *Journal of the Royal Statistical Society: Series A*, 180(1):295–314, 2017.

[13] P. Barucca, F. Lillo, P. Mazzarisi, and D. Tantari. Disentangling group and link persistence in dynamic stochastic block models. *Journal of Statistical Mechanics: Theory and Experiment*, 2018(12):123407, 2018.

[14] D. S. Bassett, E. T. Owens, M. A. Porter, M. L. Manning, and K. E. Daniels. Extraction of force-chain network architecture in granular materials using community detection. *Soft Matter*, 11(14):2731–2744, 2015.

[15] D. S. Bassett, M. A. Porter, N. F. Wymbs, S. T. Grafton, J. M. Carlson, and P. J. Mucha. Robust detection of dynamic community structure in networks. *Chaos*, 23(1):013142, 2013.

[16] D. S. Bassett and O. Sporns. Network neuroscience. *Nature Neuroscience*, 20(3):353, 2017.

[17] M. Bazzi, L. G. S. Jeub, A. Arenas, S. D. Howison, and M. A. Porter. Generative benchmark models for mesoscale structure in multilayer networks. *arXiv preprint arXiv:1608.06196*, 2016.

[18] M. Bazzi, M. A. Porter, S. Williams, M. McDonald, D. J. Fenn, and S. D. Howison. Community detection in temporal multilayer networks, with an application to correlation networks. *Multiscale Modeling & Simulation: A SIAM Interdisciplinary Journal*, 14(1):1–41, 2016.

[19] J. W. Berry, B. Hendrickson, R. A. LaViolette, and C. A. Phillips. Tolerating the community detection resolution limit with edge weighting. *Physical Review E*, 83(5):056119, 2011.

[20] R. F. Betzel and D. S. Bassett. Multi-scale brain networks. *NeuroImage*, 160:73–83, 2017.

[21] R. C. Blattberg and S. K. Sen. Market segmentation using models of multidimensional purchasing behavior. *The Journal of Marketing*, 38:17–28, 1974.

[22] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

[23] S. Boccaletti, G. Bianconi, R. Criado, C. D. Genio, J. Gómez-Gardenes, M. Romance, I. Sendina-Nadal, Z. Wang, and M. Zanin. The structure and dynamics of multilayer networks. *Physics Reports*, 544(1):1–122, 2014.

[24] U. Brandes, D. Delling, M. Gaertler, R. Görke, M. Hoefer, Z. Nikoloski, and D. Wagner. On modularity clustering. *IEEE Transactions on Knowledge and Data Engineering*, 20(2):172–188, 2008.

[25] L. Cantini, E. Medico, S. Fortunato, and M. Caselle. Detection of gene communities in multi-networks reveals cancer drivers. *Scientific Reports*, 5:17386, 2015.

[26] A. Cardillo, J. Gómez-Gardenes, M. Zanin, M. Romance, D. Papo, F. Del Pozo, and S. Boccaletti. Emergence of network features from multiplexity. *Scientific Reports*, 3:1344, 2013.

[27] A. Clauset. Finding local community structure in networks. *Physical Review E*, 72(2):026132, 2005.

[28] A. Clauset, C. Moore, and M. E. J. Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98–101, 2008.

[29] A. Clauset, M. E. Newman, and C. Moore. Finding community structure in very large networks. *Physical Review E*, 70(6):066111, 2004.

[30] D. Combe, C. Largeron, M. Géry, and E. Egyed-Zsigmond. I-Louvain: An attributed graph clustering method. In *International Symposium on Intelligent Data Analysis*, pages 181–192. Springer, 2015.

[31] A. Condon and R. M. Karp. Algorithms for graph partitioning on the planted partition model. *Random Structures and Algorithms*, 18(2):116–140, 2001.

[32] D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty years of graph matching in pattern recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(03):265–298, 2004.

[33] P. Csermely, A. London, L.-Y. Wu, and B. Uzzi. Structure and dynamics of core/periphery networks. *Journal of Complex Networks*, 1(2):93–123, 2013.

[34] C. De Bacco, E. A. Power, D. B. Larremore, and C. Moore. Community detection, link prediction, and layer interdependence in multilayer networks. *Physical Review E*, 95(4):042317, 2017.

[35] M. De Domenico, C. Granell, M. A. Porter, and A. Arenas. The physics of spreading processes in multilayer networks. *Nature Physics*, 12:901–906, 2016.

[36] M. De Domenico, A. Lancichinetti, A. Arenas, and M. Rosvall. Identifying modular flows on multilayer networks reveals highly overlapping organization in interconnected systems. *Physical Review X*, 5(1):011027, 2015.

[37] A. Decelle, F. Krzakala, C. Moore, and L. Zdeborová. Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications. *Physical Review E*, 84(6):066106, 2011.

[38] A. Decelle, F. Krzakala, C. Moore, and L. Zdeborová. Inference and phase transitions in the detection of modules in sparse networks. *Physical Review Letters*, 107(6):065701, 2011.

[39] J.-C. Delvenne, S. N. Yaliraki, and M. Barahona. Stability of graph communities across time scales. *Proceedings of the National Academy of Sciences of the United States of America*, 107(29):12755–12760, 2010.

[40] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B*, 39(1):1–38, 1977.

[41] R. Dunbar. *How Many Friends Does One Person Need? Dunbar's Number and Other Evolutionary Quirks*. Faber & Faber, 2010.

[42] S. Emmons and P. J. Mucha. A map equation with metadata: Varying the role of attributes in community detection. *arXiv preprint arXiv:1810.10433*, 2018.

[43] T. S. Evans and R. Lambiotte. Line graphs of weighted networks for overlapping communities. *The European Physical Journal B*, 77(2):265–272, 2010.

[44] J. Faskowitz, X. Yan, X.-N. Zuo, and O. Sporns. Weighted stochastic block models of the human connectome across the life span. *Scientific Reports*, 8(1):12997, 2018.

[45] D. J. Fenn, M. A. Porter, P. J. Mucha, M. McDonald, S. Williams, N. F. Johnson, and N. S. Jones. Dynamical clustering of exchange rates. *Quantitative Finance*, 12(10):1493–1520, 2012.

[46] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010.

[47] S. Fortunato and M. Barthelemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences of the United States of America*, 104(1):36–41, 2007.

[48] S. Fortunato and D. Hric. Community detection in networks: A user guide. *Physics Reports*, 659:1–44, 2016.

[49] B. K. Fosdick, D. B. Larremore, J. Nishimura, and J. Ugander. Configuring random graph models with fixed degree sequences. *SIAM Review*, 60(2):315–355, 2018.

[50] A. Ghasemian, H. Hosseinmardi, and A. Clauset. Evaluating overfit and underfit in models of network community structure. *arXiv preprint arXiv:1802.10582*, 2018.

[51] A. Ghasemian, P. Zhang, A. Clauset, C. Moore, and L. Peel. Detectability thresholds and optimal algorithms for community structure in dynamic networks. *Physical Review X*, 6(3):031005, 2016.

[52] A. Godoy-Lorite, R. Guimerà, C. Moore, and M. Sales-Pardo. Accurate and scalable social recommendation using mixed-membership stochastic block models. *Proceedings of the National Academy of Sciences*, 113(50):14207–14212, 2016.

[53] S. Gomez, A. Diaz-Guilera, J. Gomez-Gardenes, C. J. Perez-Vicente, Y. Moreno, and A. Arenas. Diffusion dynamics on multiplex networks. *Physical Review Letters*, 110(2):028701, 2013.

[54] S. Gómez, P. Jensen, and A. Arenas. Analysis of community structure in networks of correlated data. *Physical Review E*, 80(1):016114, 2009.

[55] B. Gonçalves, N. Perra, and A. Vespignani. Modeling users' activity on Twitter networks: Validation of Dunbar's number. *PloS One*, 6(8):e22656, 2011.

[56] B. H. Good, Y.-A. de Montjoye, and A. Clauset. Performance of modularity maximization in practical contexts. *Physical Review E*, 81(4):046106, 2010.

[57] R. Guimerà and M. Sales-Pardo. Missing and spurious interactions and the reconstruction of complex networks. *Proceedings of the National Academy of Sciences of the United States of America*, 106(52):22073–22078, 2009.

[58] R. Guimerà and M. Sales-Pardo. A network inference method for large-scale unsupervised identification of novel drug-drug interactions. *PLoS Computational Biology*, 9(12):e1003374, 2013.

[59] M. B. Hastings. Community detection as an inference problem. *Physical Review E*, 74(3):035102, 2006.

[60] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *ACM SIGIR Forum*, volume 51, pages 227–234. ACM, 2017.

[61] J. M. Hofman and C. H. Wiggins. Bayesian approach to network modularity. *Physical Review Letters*, 100(25):258701, 2008.

[62] P. Holgate. Estimation for the bivariate Poisson distribution. *Biometrika*, 51(1-2):241–287, 1964.

[63] P. Holme and J. Saramäki. Temporal networks. *Physics Reports*, 519(3):97–125, 2012.

[64] D. Hric, K. Kaski, and M. Kivelä. Stochastic block model reveals maps of citation patterns and their evolution in time. *Journal of Informetrics*, 12(3):757–783, 2018.

[65] D. Hric, T. P. Peixoto, and S. Fortunato. Network structure, metadata, and the prediction of missing nodes and annotations. *Physical Review X*, 6(3):031038, 2016.

[66] L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985.

[67] Y. Hulovatyy and T. Milenković. SCOUT: Simultaneous time segmentation and community detection in dynamic networks. *Scientific Reports*, 6:37557, 2016.

[68] M. D. Intriligator. *Mathematical optimization and economic theory*, volume 39. SIAM, 1971.

[69] A. K. Jain, M. N. Murty, and P. J. Flynn. Data Clustering: A Review. *ACM Computing Surveys (CSUR)*, 31(3):264–323, 1999.

[70] L. G. S. Jeub. *Networks, Communities, and Consumer Behaviour*. PhD thesis, University of Oxford, 2015.

[71] L. G. S. Jeub, P. Balachandran, M. A. Porter, P. J. Mucha, and M. W. Mahoney. Think locally, act locally: Detection of small, medium-sized, and large communities in large networks. *Physical Review E*, 91(1):012821, 2015.

[72] L. G. S. Jeub and M. Bazzi. A generative model for mesoscale structure in multilayer networks implemented in MATLAB. `https://github.com/MultilayerBenchmark/MultilayerBenchmark/` (2016), version 1.0.

[73] L. G. S. Jeub, M. Bazzi, I. S. Jutla, and P. J. Mucha. A generalized Louvain method for community detection implemented in MATLAB. `http://netwiki.amath.unc.edu/GenLouvain` (2011–2018), version 2.1.

[74] L. G. S. Jeub, M. W. Mahoney, P. J. Mucha, and M. A. Porter. A local perspective on community structure in multilayer networks. *Network Science*, 5(2):144–163, 2017.

[75] L. G. S. Jeub, O. Sporns, and S. Fortunato. Multiresolution consensus clustering in networks. *Scientific Reports*, 8(1):3259, 2018.

[76] B. Karrer and M. E. J. Newman. Stochastic blockmodels and community structure in networks. *Physical Review E*, 83(1):016107, 2011.

[77] K.-J. Kim and H. Ahn. A recommender system using GA K-means clustering in an online shopping market. *Expert Systems with Applications*, 34(2):1200–1209, 2008.

[78] M. Kivelä, A. Arenas, M. Barthelemy, J. P. Gleeson, Y. Moreno, and M. A. Porter. Multilayer networks. *Journal of Complex Networks*, 2(3):203–271, 2014.

[79] Y. Koren. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 426–434. ACM, 2008.

[80] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 1(8):30–37, 2009.

[81] R. Kuo, L. Ho, and C. M. Hu. Integration of self-organizing feature map and K-means algorithm for market segmentation. *Computers & Operations Research*, 29(11):1475–1493, 2002.

[82] R. Lambiotte. Multi-scale modularity in complex networks. In *Proceedings of the 8th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, pages 546–553. IEEE, 2010.

[83] R. Lambiotte, J.-C. Delvenne, and M. Barahona. Laplacian dynamics and multiscale modular structure in networks. *arXiv preprint arXiv:0812.1770*, 2008.

[84] R. Lambiotte, J.-C. Delvenne, and M. Barahona. Random walks, Markov processes and the multiscale modular organization of complex networks. *IEEE Transactions on Network Science and Engineering*, 1(2):76–90, 2014.

[85] A. Lancichinetti and S. Fortunato. Community detection algorithms: A comparative analysis. *Physical Review E*, 80(5):056117, 2009.

[86] A. Lancichinetti and S. Fortunato. Limits of modularity maximization in community detection. *Physical Review E*, 84(6):066122, 2011.

[87] A. Lancichinetti and S. Fortunato. Consensus clustering in complex networks. *Scientific Reports*, 2:336, 2012.

[88] D. B. Larremore, A. Clauset, and A. Z. Jacobs. Efficiently inferring community structure in bipartite networks. *Physical Review E*, 90(1):012805, 2014.

[89] E. Lazega. *The Collegial Phenomenon: The Social Mechanisms of Cooperation Among Peers in a Corporate Law Partnership*. Oxford University Press, 2001.

[90] A. Lomi, G. Robins, and M. Tranmer. Editorial: Introduction to multilevel social networks. *Social Networks*, 44:266–268, 2016.

[91] L. Lü, M. Medo, C. H. Yeung, Y.-C. Zhang, Z.-K. Zhang, and T. Zhou. Recommender systems. *Physics Reports*, 519(1):1–49, 2012.

[92] L. Lü and T. Zhou. Link prediction in complex networks: A survey. *Physica A*, 390(6):1150–1170, 2011.

[93] F. Luo, J. Z. Wang, and E. Promislow. Exploring local community structures in large networks. *Web Intelligence and Agent Systems: An International Journal*, 6(4):387–400, 2008.

[94] V. Lyzinski, D. E. Fishkind, and C. E. Priebe. Seeded graph matching for correlated Erdős–Rényi graphs. *Journal of Machine Learning Research*, 15(1):3513–3540, 2014.

[95] V. Lyzinski and D. L. Sussman. Graph matching the matchable nodes when some nodes are unmatchable. *arXiv preprint arXiv:1705.02294*, 2017.

[96] V. Lyzinski, D. L. Sussman, D. E. Fishkind, H. Pao, L. Chen, J. T. Vogelstein, Y. Park, and C. E. Priebe. Spectral clustering for divide-and-conquer graph matching. *Parallel Computing*, 47:70–87, 2015.

[97] C. Moore. The computer science and physics of community detection: Landscapes, phase transitions, and hardness. *Bulletin of the European Association for Theoretical Computer Science*, 121:25–61, 2017.

[98] P. J. Mucha, T. Richardson, K. Macon, M. A. Porter, and J.-P. Onnela. Community structure in time-dependent, multiscale, and multiplex networks. *Science*, 328(5980):876–878, 2010.

[99] A. C. Murphy, S. Gu, A. N. Khambhati, N. F. Wymbs, S. T. Grafton, T. D. Satterthwaite, and D. S. Bassett. Explicitly linking regional activation and function connectivity: Community structure of weighted networks with continuous annotation. *arXiv preprint arXiv:1611.07962*, 2016.

[100] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *IEEE Symposium on Security and Privacy*, pages 111–125. IEEE, 2008.

[101] M. E. Newman and T. P. Peixoto. Generalized communities in networks. *Physical Review Letters*, 115(8):088701, 2015.

[102] M. E. Newman and G. Reinert. Estimating the number of communities in a network. *Physical Review Letters*, 117(7):078301, 2016.

[103] M. E. J. Newman. Analysis of weighted networks. *Physical Review E*, 70(5):056131, 2004.

[104] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 74(3):036104, 2006.

[105] M. E. J. Newman. Equivalence between modularity optimization and maximum likelihood methods for community detection. *Physical Review E*, 94(5):052315, 2016.

[106] M. E. J. Newman. Network structure from rich but noisy data. *Nature Physics*, 14(6):542, 2018.

[107] M. E. J. Newman. *Networks*. Oxford University Press, 2018.

[108] M. E. J. Newman and G. T. Barkema. *Monte Carlo Methods in Statistical Physics*. Oxford University Press, 1999.

[109] M. E. J. Newman and A. Clauset. Structure and inference in annotated networks. *Nature Communications*, 7(11863), 2016.

[110] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2):026113, 2004.

[111] A. R. Pamfil, S. D. Howison, R. Lambiotte, and M. A. Porter. Relating modularity maximization and stochastic block models in multilayer networks. *arXiv preprint arXiv:1804.01964*, 2018.

[112] L. Peel and A. Clauset. Detecting change points in the large-scale structure of evolving networks. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2914–2920, 2015.

[113] L. Peel, D. B. Larremore, and A. Clauset. The ground truth about metadata and community detection in networks. *Science Advances*, 3(5):e1602548, 2017.

[114] T. P. Peixoto. Parsimonious module inference in large networks. *Physical Review Letters*, 110(14):148701, 2013.

[115] T. P. Peixoto. Hierarchical block structures and high-resolution model selection in large networks. *Physical Review X*, 4(1):011047, 2014.

[116] T. P. Peixoto. Inferring the mesoscale structure of layered, edge-valued, and time-varying networks. *Physical Review E*, 92(4):042807, 2015.

[117] T. P. Peixoto. Model selection and hypothesis testing for large-scale network models with overlapping groups. *Physical Review X*, 5(1):011033, 2015.

[118] T. P. Peixoto. Bayesian stochastic blockmodeling. *arXiv preprint arXiv:1705.10225*, 2017.

[119] T. P. Peixoto. Nonparametric weighted stochastic block models. *Physical Review E*, 97(1):012306, 2018.

[120] T. P. Peixoto. Reconstructing networks with unknown and heterogeneous errors. *arXiv preprint arXiv:1806.07956*, 2018.

[121] T. P. Peixoto and M. Rosvall. Modelling sequences and temporal networks with dynamic community structures. *Nature Communications*, 8(1):582, 2017.

[122] P. O. Perry and P. J. Wolfe. Null models for network data. *arXiv preprint arXiv:1201.5871*, 2012.

[123] M. A. Porter and S. D. Howison. The role of network analysis in industrial and applied mathematics: A physical-applied-mathematics perspective. *arXiv preprint arXiv:1703.06843*, 2017.

[124] M. A. Porter, J.-P. Onnela, and P. J. Mucha. Communities in networks. *Notices of the AMS*, 56(9):1082–1097,1164–1166, 2009.

[125] I. Psorakis, S. Roberts, M. Ebden, and B. Sheldon. Overlapping community detection using Bayesian non-negative matrix factorization. *Physical Review E*, 83(6):066114, 2011.

[126] P. K. Reddy, M. Kitsuregawa, P. Sreekanth, and S. S. Rao. A graph based approach to extract a neighborhood customer community for collaborative filtering. In *International Workshop on Databases in Networked Information Systems*, pages 188–200. Springer, 2002.

[127] J. Reichardt and S. Bornholdt. Statistical mechanics of community detection. *Physical Review E*, 74(1):016110, 2006.

[128] F. Ricci, L. Rokach, and B. Shapira. Recommender systems: Introduction and challenges. In *Recommender Systems Handbook*, pages 1–34. Springer, 2015.

[129] M. A. Riolo, G. T. Cantwell, G. Reinert, and M. E. J. Newman. Efficient method for estimating the number of communities in a network. *Physical Review E*, 96(3):032310, 2017.

[130] P. Rombach, M. A. Porter, J. H. Fowler, and P. J. Mucha. Core–periphery structure in networks (revisited). *SIAM Review*, 59(3):619–646, 2017.

[131] R. A. Rossi and N. K. Ahmed. Role discovery in networks. *IEEE Transactions on Knowledge and Data Engineering*, 27:1112–1131, 2015.

[132] M. Rosvall and C. T. Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences of the United States of America*, 105(4):1118–1123, 2008.

[133] M. Rosvall, A. V. Esquivel, A. Lancichinetti, J. D. West, and R. Lambiotte. Memory in network flows and its effects on spreading dynamics and community detection. *Nature Communications*, 5:4630, 2014.

[134] M. Sales-Pardo, R. Guimerà, A. A. Moreira, and L. A. N. Amaral. Extracting the hierarchical organization of complex systems. *Proceedings of the National Academy of Sciences of the United States of America*, 104(39):15224–15229, 2007.

[135] M. Sarzynska, E. A. Leicht, G. Chowell, and M. A. Porter. Null models for community detection in spatially-embedded, temporal networks. *Journal of Complex Networks*, 4(3):363–406, 2016.

[136] J. B. Schafer, J. Konstan, and J. Riedl. Recommender systems in e-commerce. In *Proceedings of the 1st ACM Conference on Electronic Commerce*, pages 158–166. ACM, 1999.

[137] M. T. Schaub, J.-C. Delvenne, M. Rosvall, and R. Lambiotte. The many facets of community detection in complex networks. *Applied Network Science*, 2(1):4, 2017.

[138] M. T. Schaub, J.-C. Delvenne, S. N. Yaliraki, and M. Barahona. Markov dynamics as a zooming lens for multiscale community detection: Non clique-like communities and the field-of-view limit. *PLoS ONE*, 7(2):e32210, 2012.

[139] S. Shai, N. Stanley, C. Granell, D. Taylor, and P. J. Mucha. Case studies in network community detection. *arXiv preprint arXiv:1705.02305*, 2017.

[140] C. Shi, Y. Cai, D. Fu, Y. Dong, and B. Wu. A link clustering based overlapping community detection algorithm. *Data & Knowledge Engineering*, 87:394–404, 2013.

[141] J. G. Skellam. The frequency distribution of the difference between two Poisson variates belonging to different populations. *Journal of the Royal Statistical Society: Series A*, 109(3):296, 1946.

[142] T. A. B. Snijders and C. Baerveldt. A multilevel network study of the effects of delinquent behavior on friendship evolution. *Journal of Mathematical Sociology*, 27(2-3):123–151, 2003.

[143] T. A. B. Snijders, M. Spreen, and R. Zwaagstra. The use of multilevel modeling for analysing personal networks: Networks of cocaine users in an urban area. *Journal of Quantitative Anthropology*, 5(2):85–105, 1995.

[144] N. Stanley, T. Bonacci, R. Kwitt, M. Niethammer, and P. J. Mucha. Stochastic block models with multiple continuous attributes. *arXiv preprint arXiv:1803.02726*, 2018.

[145] N. Stanley, S. Shai, D. Taylor, and P. J. Mucha. Clustering network layers with the strata multilayer stochastic block model. *IEEE Transactions on Network Science and Engineering*, 3(2):95–105, 2016.

[146] A. Strehl and J. Ghosh. Cluster ensembles—A knowledge reuse framework for combining multiple partitions. *The Journal of Machine Learning Research*, 3:583–617, 2003.

[147] X. Su and T. M. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, 2009:421425, 2009.

[148] M. Tarres-Deulofeu, A. Godoy-Lorite, R. Guimera, and M. Sales-Pardo. Tensorial and bipartite block models for link prediction in layered networks and temporal networks. *arXiv preprint arXiv:1803.01616*, 2018.

[149] D. Taylor, S. A. Myers, A. Clauset, M. A. Porter, and P. J. Mucha. Eigenvector-based centrality measures for temporal networks. *Multiscale Modeling & Simulation*, 15(1):537–574, 2017.

[150] D. Taylor, S. Shai, N. Stanley, and P. J. Mucha. Enhanced detectability of community structure in multilayer networks through layer aggregation. *Physical Review Letters*, 116(22):228301, 2016.

[151] A. L. Traud, E. D. Kelsic, P. J. Mucha, and M. A. Porter. Comparing community structure to characteristics in online collegiate social networks. *SIAM Review*, 53(3):526–543, 2011.

[152] E. Valdano, L. Ferreri, C. Poletto, and V. Colizza. Analytical computation of the epidemic threshold on temporal networks. *Physical Review X*, 5(2):021005, 2015.

[153] T. Vallès-Català, F. A. Massucci, R. Guimerà, and M. Sales-Pardo. Multilayer stochastic block models reveal the multilayer structure of complex networks. *Physical Review X*, 6(1):011036, 2016.

[154] M. A. J. Van Duijn, J. T. Van Busschbach, and T. A. B. Snijders. Multilevel analysis of personal networks as dependent variables. *Social Networks*, 21(2):187–210, 1999.

[155] A. Vellido, P. Lisboa, and K. Meehan. Segmentation of the on-line shopping market using neural networks. *Expert Systems with Applications*, 17(4):303–314, 1999.

[156] N. X. Vinh, J. Epps, and J. Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research*, 11(Oct):2837–2854, 2010.

[157] S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press, 1994.

[158] W. H. Weir, S. Emmons, R. Gibson, D. Taylor, and P. J. Mucha. Post-processing partitions to identify domains of modularity optimization. *Algorithms*, 10(3):93, 2017.

[159] J. Wu, H. Xiong, and J. Chen. Adapting the right measures for K-means clustering. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 877–886. ACM, 2009.

[160] J. Xiang and K. Hu. Limitation of multi-resolution methods in community detection. *Physica A: Statistical Mechanics and its Applications*, 391(20):4995–5003, 2012.

[161] J.-R. Xie and B.-H. Wang. Modularity-like objective function in annotated networks. *Frontiers of Physics*, 12(6):128903, 2017.

[162] X. Yan, L. G. S. Jeub, A. Flammini, F. Radicchi, and S. Fortunato. Weight thresholding on complex networks. *arXiv preprint arXiv:1806.07479*, 2018.

[163] X. Yan, C. Shalizi, J. E. Jensen, F. Krzakala, C. Moore, L. Zdeborová, P. Zhang, and Y. Zhu. Model selection for degree-corrected block models. *Journal of Statistical Mechanics: Theory and Experiment*, 2014(5):P05007, 2014.

[164] J. Yang and J. Leskovec. Overlapping community detection at scale: A nonnegative matrix factorization approach. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, pages 587–596. ACM, 2013.

[165] J. Yang, J. McAuley, and J. Leskovec. Community detection in networks with node attributes. In *2013 IEEE 13th International Conference on Data Mining (ICDM)*, pages 1151–1156. IEEE, 2013.

[166] J.-G. Young, G. St-Onge, P. Desrosiers, and L. J. Dubé. On the universality of the stochastic block model. *arXiv preprint arXiv:1806.04214*, 2018.

[167] P. Zhang, C. Moore, and M. E. J. Newman. Community detection in networks with unequal groups. *Physical Review E*, 93(1):012303, 2016.

[168] X. Zhang, C. Moore, and M. E. J. Newman. Random graph models for dynamic networks. *The European Physical Journal B*, 90(10):200, 2017.

[169] Y. Zhang, E. Levina, and J. Zhu. Community detection in networks with node features. *Electronic Journal of Statistics*, 10(2):3153–3178, 2016.

[170] Z.-K. Zhang, C. Liu, Y.-C. Zhang, and T. Zhou. Solving the cold-start problem in recommender systems with social tags. *Europhysics Letters*, 92(2):28002, 2010.

[171] Y. Zhao, E. Levina, and J. Zhu. Consistency of community detection in networks under degree-corrected stochastic block models. *The Annals of Statistics*, 40(4):2266–2292, 2012.

[172] T. Zhou, J. Ren, M. Medo, and Y.-C. Zhang. Bipartite network projection and personal recommendation. *Physical Review E*, 76(4):046115, 2007.

Appendix A

Additional Results for Annotated Networks

A.1 Number of Blocks

Table A.1: Number of customer and product blocks determined using Algorithm 4.1.

Threshold	*Item penetration*		*Basket penetration*	
	K_C (mode)	K_P (mode)	K_C (mode)	K_P (mode)
0.0	21	21	23	23
0.1	22	23	28	23
0.2	24	23	25	22
0.3	24	19	20	21
0.4	21	22	22	23
0.5	24	21	21	18
0.6	23	18	25	17
0.7	17	18	19	15
0.8	11	15	12	12

A.2 Partitions with Six Customer Blocks and Six Product Blocks

Basket-penetration weighting

In this section, we present results analogous to those from Section 4.4.3. We use basket-penetration weighting (rather than item-penetration weighting) and a 50^{th}-percentile threshold to generate an input network for our inference algorithm for unweighted annotated bipartite networks.

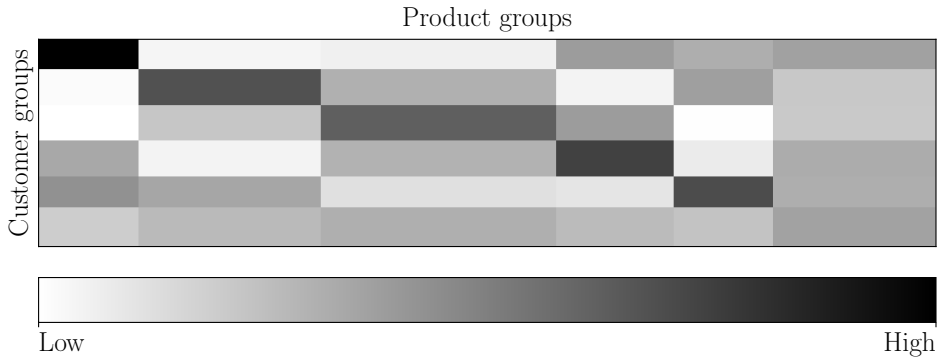
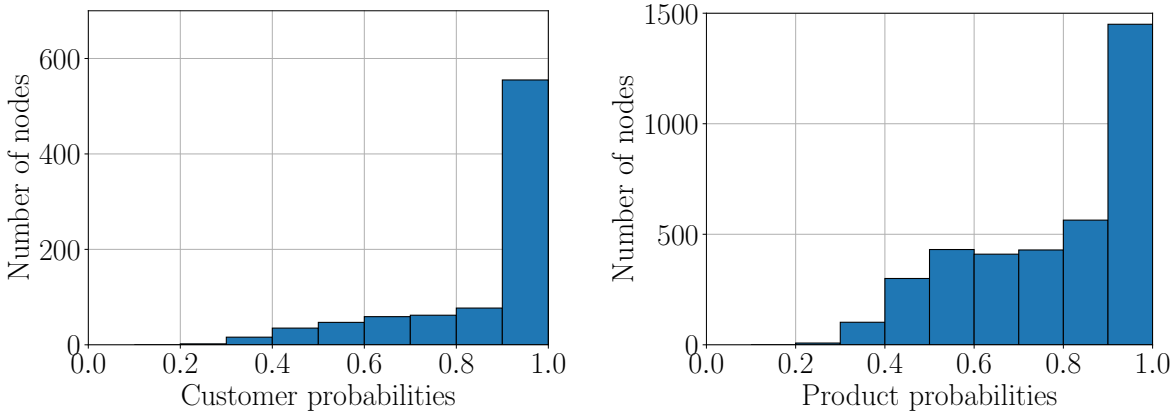


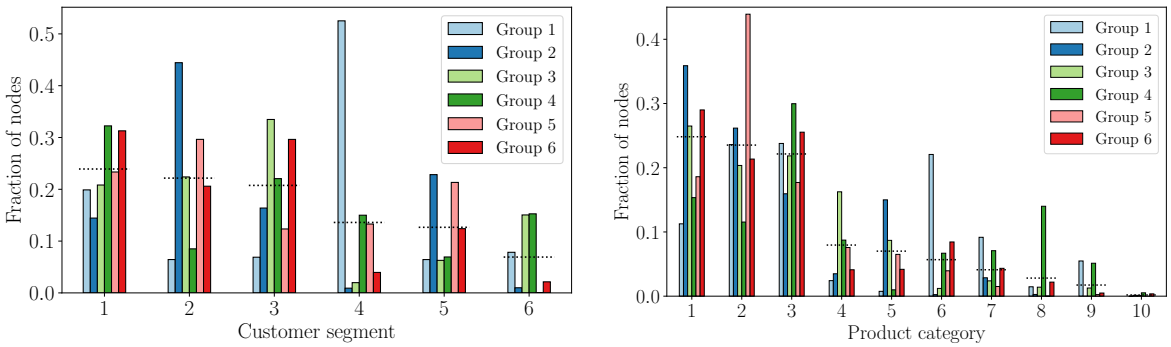
Figure A.1: Heatmap of $\boldsymbol{\Theta}$ values, summarising patterns of interaction among $K_C = 6$ customer blocks and $K_P = 6$ product blocks, using basket-penetration weighting and a 50^{th}-percentile threshold to generate the input unweighted network. This plot is qualitatively similar to the one from Figure 4.4(a), which is for a network with item-penetration weighting.



(a) Histogram of $\max_{r \in \mathcal{K}_C} q_r^c$ (customer nodes)

(b) Histogram of $\max_{s \in \mathcal{K}_P} q_s^p$ (product nodes)

Figure A.2: Distributions of the largest block-membership probabilities for (a) customers and (b) products. Similarly to Figure 4.5, these plots suggest that block assignments are more uncertain for products than for customers.



(a) Comparison to existing customer segments

(b) Comparison to existing product categories

Figure A.3: (a) Proportion of customer nodes in each block that fall in each `purchasing-medium` segment. Similarly to Figure 4.6(a), block 1 has many customers from segment 4 (an upscale segment) and block 2 has many customers from segment 2 (a price-sensitive segment). (b) Proportion of product nodes in each block that fall in each `L3` product category. The same product categories are over-represented in blocks 1 and 2 as in Figure 4.6(b).

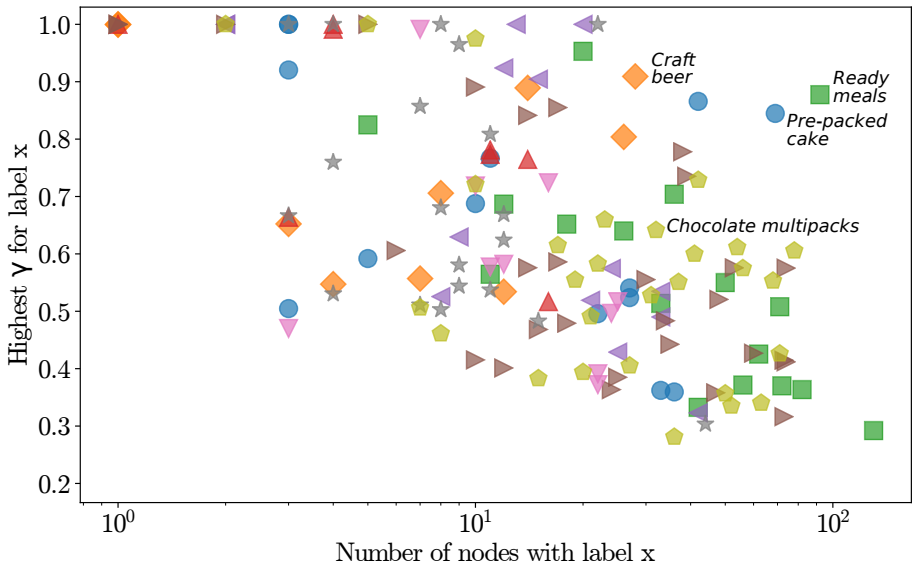


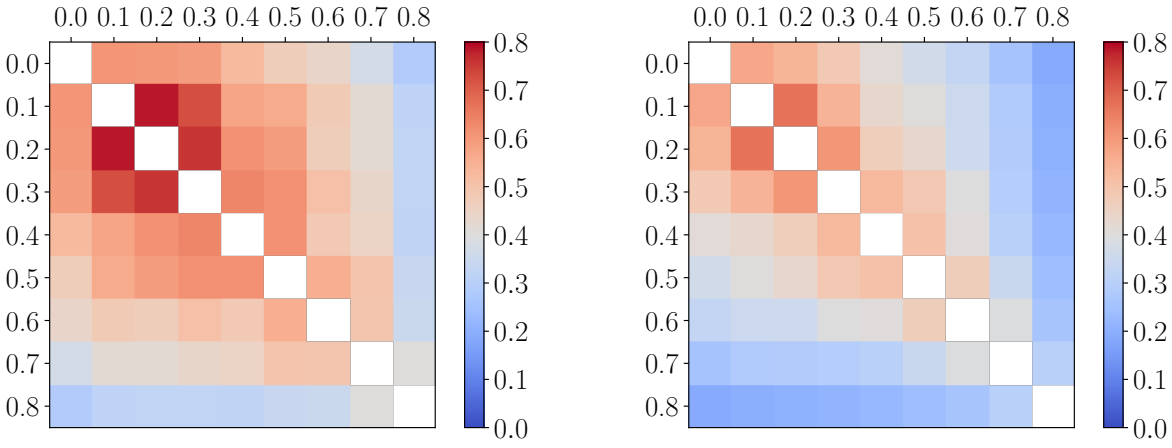
Figure A.4: Scatter plot of the largest γ_{sx} for each `L2` label x and block $s \in \mathcal{K}_P$. The horizontal axis indicates the number of products ($n_x = \sum_{p \in \mathcal{N}_P} \delta(x_p, x)$) with the respective label. In the upper-right corner of the plot, we highlight a subset of labels that are informative of the network's block structure. (These are the same four labels as those in Figure 4.7.)

Varying the threshold for generating unweighted networks

Our insights from Section 4.4.3 are robust to different choices of threshold. In particular, the composition of customer and product blocks by `purchasing- medium` segments and `L3` categories, respectively, is very similar across a wide range of thresholds to that from Figure 4.6. For example, block 1 includes many upscale customers and block 2 includes many price-sensitive customers. Because the resulting figures are similar to Figure 4.6 (and to Figure A.3 for basket-penetration weighting), we do not show them. Tables A.2–A.3 and Figures A.5–A.6 summarise changes in other outputs of the inference algorithm as we vary the percentile threshold.

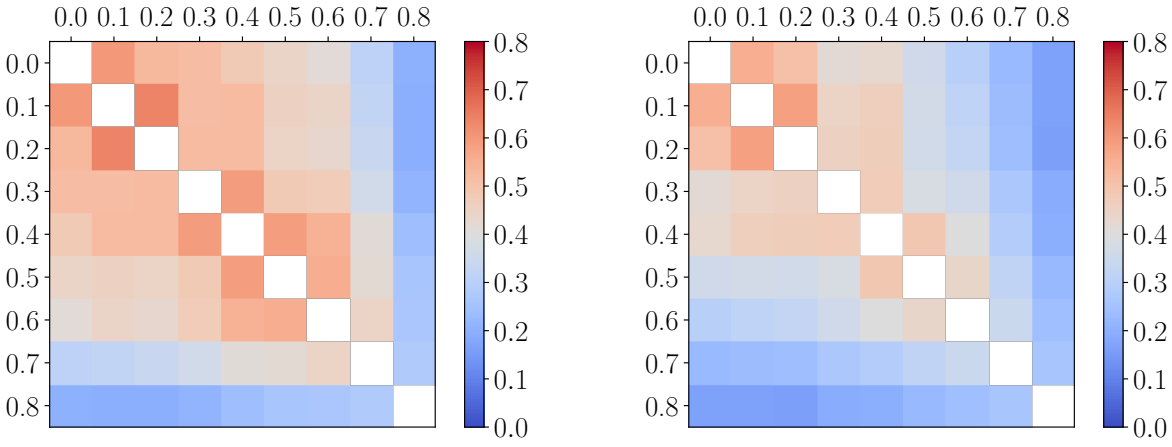
Table A.2: As we increase the percentile threshold that we use to generate unweighted networks, the distributions of the largest block-membership probabilities (i.e., the distributions of $q^c_{\max} = \max_{r \in \mathcal{K}_C} q^c_r$ for customers and $q^p_{\max} = \max_{s \in \mathcal{K}_P} q^p_s$ for products) shift to the left. This table indicates the means of these two distributions and the fraction of nodes in which the largest block-membership probability is at least 0.9.

	Threshold	*Customers*		*Products*	
		$\mathbb{E}[q^c_{\max}]$	$\mathbb{P}(q^c_{\max} \geq 0.9)$	$\mathbb{E}[q^p_{\max}]$	$\mathbb{P}(q^p_{\max} \geq 0.9)$
Item penetration	0.0	0.95	0.83	0.81	0.48
	0.1	0.94	0.83	0.82	0.48
	0.2	0.94	0.81	0.81	0.46
	0.3	0.94	0.80	0.79	0.42
	0.4	0.92	0.77	0.78	0.38
	0.5	0.91	0.70	0.77	0.37
	0.6	0.90	0.69	0.76	0.34
	0.7	0.87	0.62	0.76	0.34
	0.8	0.84	0.51	0.75	0.33
Basket penetration	0.0	0.95	0.85	0.82	0.49
	0.1	0.94	0.81	0.81	0.48
	0.2	0.92	0.77	0.80	0.45
	0.3	0.91	0.75	0.79	0.41
	0.4	0.90	0.70	0.79	0.41
	0.5	0.87	0.65	0.78	0.39
	0.6	0.86	0.61	0.76	0.34
	0.7	0.84	0.57	0.77	0.36
	0.8	0.78	0.40	0.77	0.35



(a) Customer nodes (b) Product nodes

Figure A.5: Pairwise NMI between hard partitions (obtained by setting $g_c = \text{argmax}_{r \in \mathcal{K}_C} q_r^c$ for customer nodes and $g_p = \text{argmax}_{s \in \mathcal{K}_P} q_s^p$ for product nodes) for different threshold values using item-penetration weighting. Partitions tend to become more dissimilar for larger threshold values than for smaller ones.



(a) Customer nodes (b) Product nodes

Figure A.6: Pairwise NMI between hard partitions (obtained by setting $g_c = \text{argmax}_{r \in \mathcal{K}_C} q_r^c$ for customer nodes and $g_p = \text{argmax}_{s \in \mathcal{K}_P} q_s^p$ for product nodes) for different threshold values using basket-penetration weighting. Partitions tend to become more dissimilar for larger threshold values than for smaller ones.

Table A.3: Increasing the threshold only slightly affects the distribution of $\gamma_x^{\max} = \max_{s \in \mathcal{K}_P} \gamma_{sx}$, which describes how informative different labels are about the given network's block structure.

Threshold	*Item penetration*		*Basket penetration*	
	$\mathbb{E}[\gamma_x^{\max}]$	$\mathbb{P}(\gamma_x^{\max} \geq 0.9)$	$\mathbb{E}[\gamma_x^{\max}]$	$\mathbb{P}(\gamma_x^{\max} \geq 0.9)$
0.0	0.62	0.19	0.64	0.20
0.1	0.64	0.20	0.63	0.23
0.2	0.63	0.19	0.63	0.21
0.3	0.61	0.17	0.62	0.14
0.4	0.62	0.18	0.62	0.17
0.5	0.63	0.19	0.64	0.19
0.6	0.63	0.16	0.63	0.15
0.7	0.66	0.22	0.67	0.21
0.8	0.69	0.27	0.73	0.31

A.3 Partitions with a Larger Number of Blocks

Table A.4: Summary of outputs that we obtain when setting K_C and K_P to the values from Table A.1 for a threshold value of 0.5. Compared to our results from Appendix A.2 (which correspond to $K_C = K_P = 6$), we obtain larger block-membership probabilities $q_{\max}^c$ and $q_{\max}^p$, and we obtain smaller probabilities $\gamma_x^{\max}$ that associate annotations to the inferred block structure. The NMI values between these partitions and those that we obtained for $K_C = K_P = 6$ suggest some agreement between these different divisions of the network into blocks.

Summary statistic	*Item penetration* $(K_C, K_P) = (24, 21)$	*Basket penetration* $(K_C, K_P) = (21, 18)$
$\mathbb{E}[q_{\max}^c]$	0.95	0.89
$\mathbb{P}(q_{\max}^c \geq 0.9)$	0.87	0.71
$\mathbb{E}[q_{\max}^p]$	0.77	0.78
$\mathbb{P}(q_{\max}^p \geq 0.9)$	0.43	0.42
$\mathbb{E}[\gamma_x^{\max}]$	0.51	0.55
$\mathbb{P}(\gamma_x^{\max} \geq 0.9)$	0.07	0.11
NMI with $(K_C, K_P) = (6, 6)$ (cust.)	0.44	0.45
NMI with $(K_C, K_P) = (6, 6)$ (prod.)	0.35	0.34

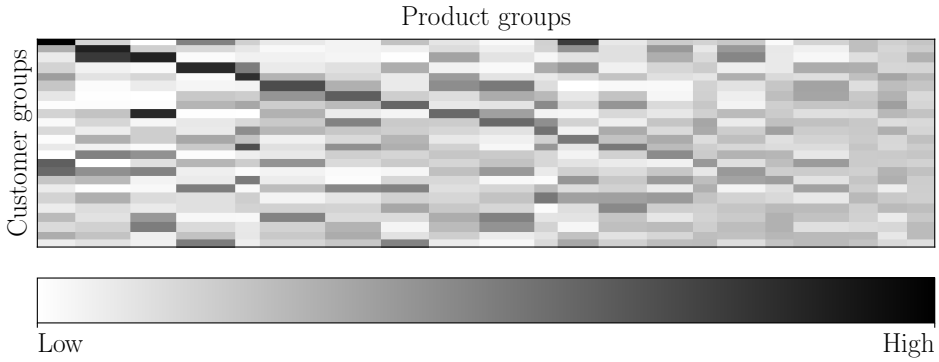


Figure A.7: Heatmap of $\boldsymbol{\Theta}$ values, summarising patterns of interaction among $K_C = 24$ customer blocks and $K_P = 21$ product blocks, using item-penetration weighting and a 50^{th}-percentile threshold to generate the input unweighted network.

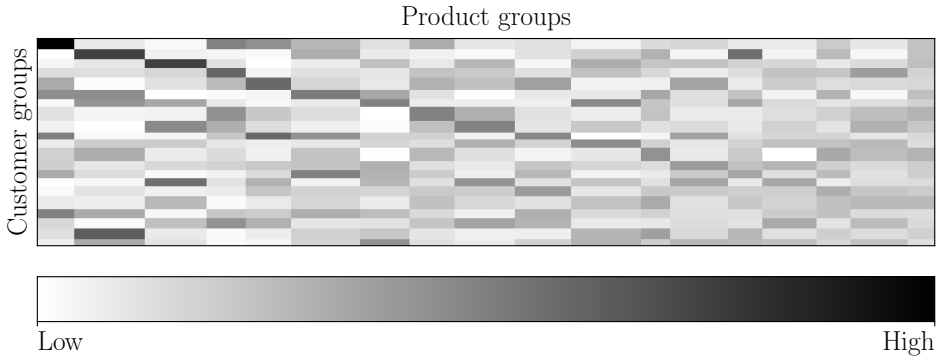
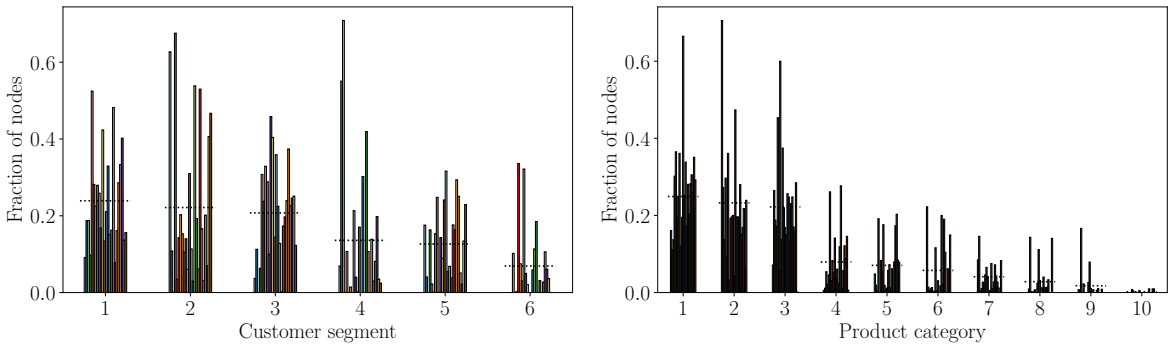


Figure A.8: Heatmap of $\boldsymbol{\Theta}$ values, summarising patterns of interaction among $K_C = 21$ customer blocks and $K_P = 18$ product blocks, using basket-penetration weighting and a 50^{th}-percentile threshold to generate the input unweighted network.



(a) Comparison to existing customer segments (b) Comparison to existing product categories

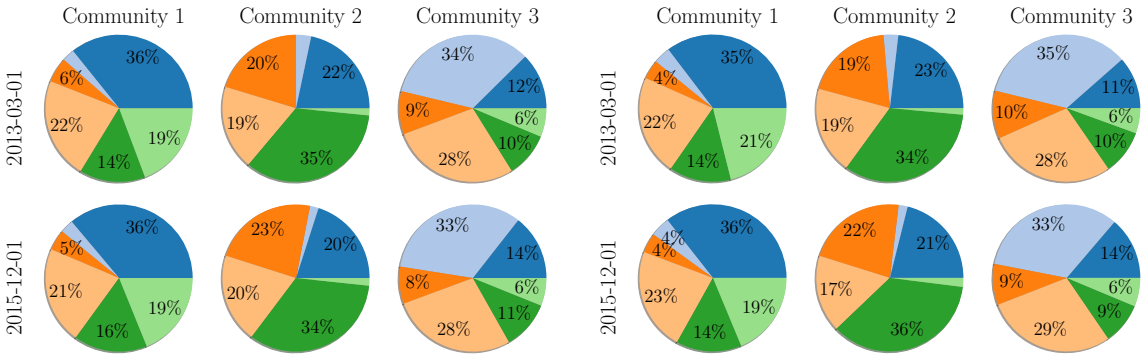
Figure A.9: Comparison of inferred blocks to metadata for $(K_C, K_P) = (24, 21)$ using item-penetration weighting. (Our results for basket-penetration weighting are similar.) Different customer segments and product categories are over-represented in different blocks.

Appendix B

Additional Results for Multilayer Networks

B.1 Temporal Networks

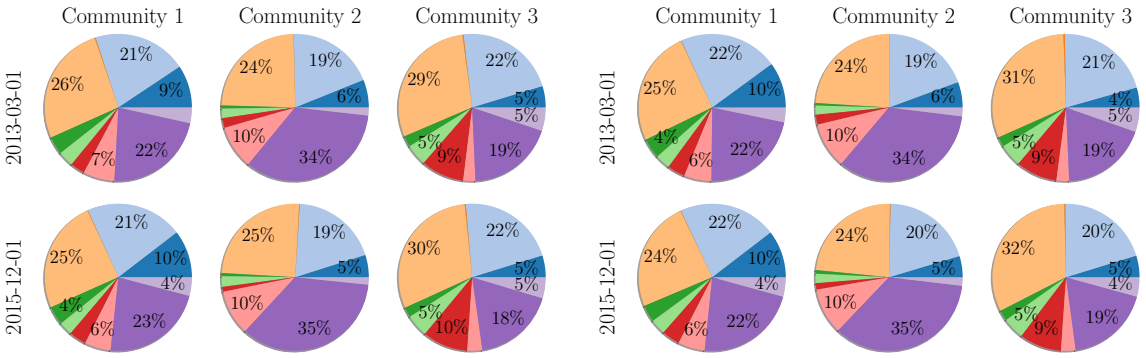
Community composition in the first and last temporal layers



(a) Item penetration

(b) Basket penetration

Figure B.1: Composition of customer communities by `purchasing-medium` segments in the first and last layers of the temporal network from Section 5.5.1 (using a 50^{th}-percentile threshold and $\gamma = 1.0$). The distribution of customers across these segments is approximately the same in the two time windows for each of the three communities.



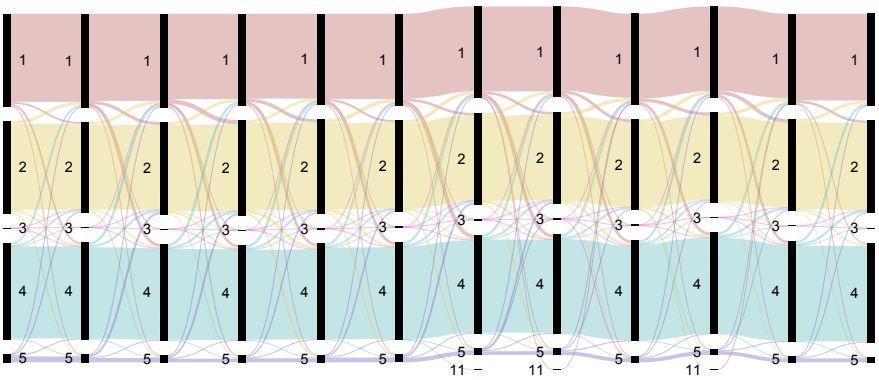
(a) Item penetration

(b) Basket penetration

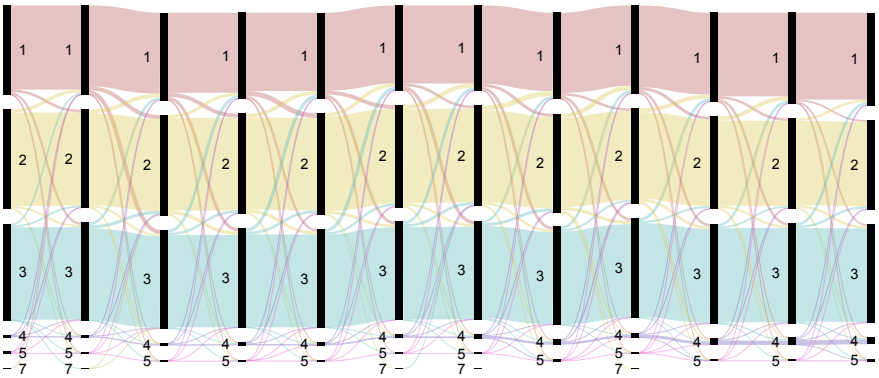
Figure B.2: Composition of product communities by L3 categories in the first and last layers of the temporal network from Section 5.5.1 (using a 50^{th}-percentile threshold and $\gamma = 1.0$). The distribution of products across these categories is approximately the same in the two time windows for each of the three communities.

Partitions for $\gamma = 1.1$

In Figures B.3–B.5, we show visualisations of the consensus partitions that we obtain for $\gamma = 1.1$. These are similar to the analogous plots in Section 5.5.1, which correspond to $\gamma = 1.0$. In particular, although we use a larger resolution value for the results in this appendix, the resulting consensus partitions still have just 3 communities of significant size.

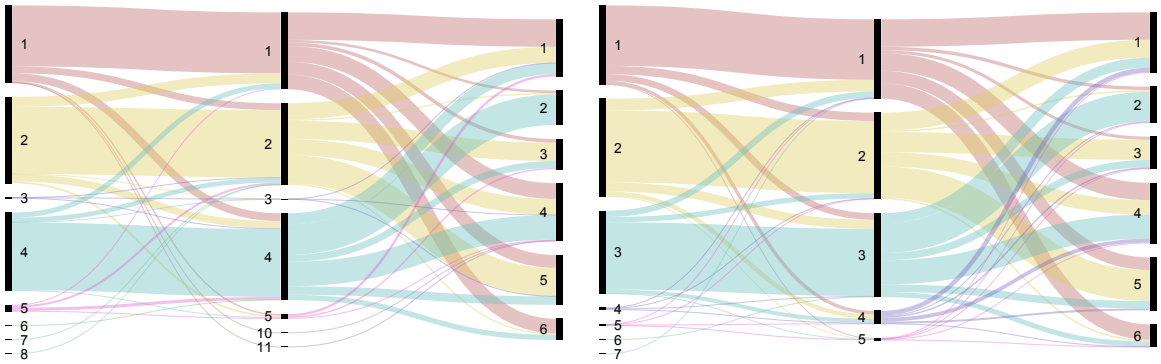


(a) Item penetration, $\gamma = 1.1$



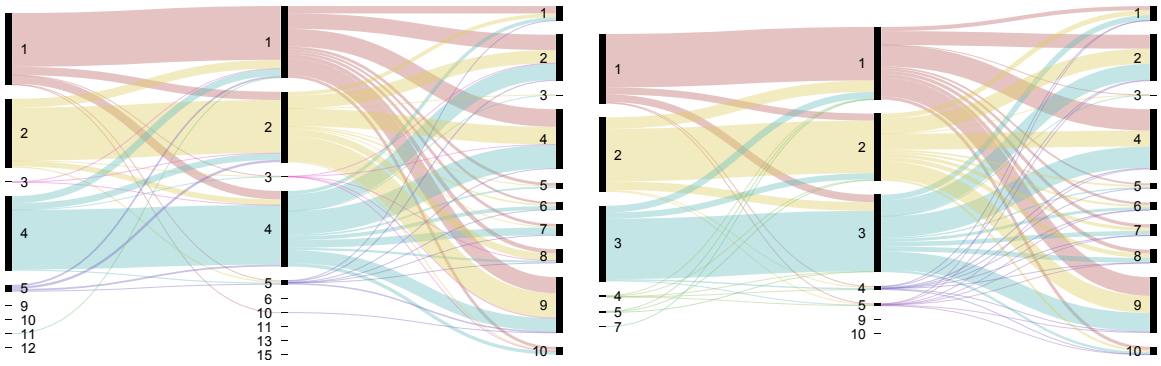
(b) Basket penetration, $\gamma = 1.1$

Figure B.3: Alluvial plots of the consensus partitions obtained for $\gamma = 1.1$ and either (a) item-penetration or (b) basket-penetration weighting (with a subsequent 50^{th}-percentile threshold applied to both). In each case, there are 3 large communities. Their sizes remain roughly constant through time, but on average about 8% of the nodes change communities between consecutive layers.



(a) Item penetration, $\gamma = 1.1$ (b) Basket penetration, $\gamma = 1.1$

Figure B.4: Customer communities in the first and last time windows and their mapping to the 6 `purchasing-medium` segments for $\gamma = 1.1$ with (a) item-penetration weighting and (b) basket-penetration weighting. We observe similar patterns to the analogous plots from Figure 5.19 for $\gamma = 1.0$. In particular, nearly all customers in segment 2 belong to one community.

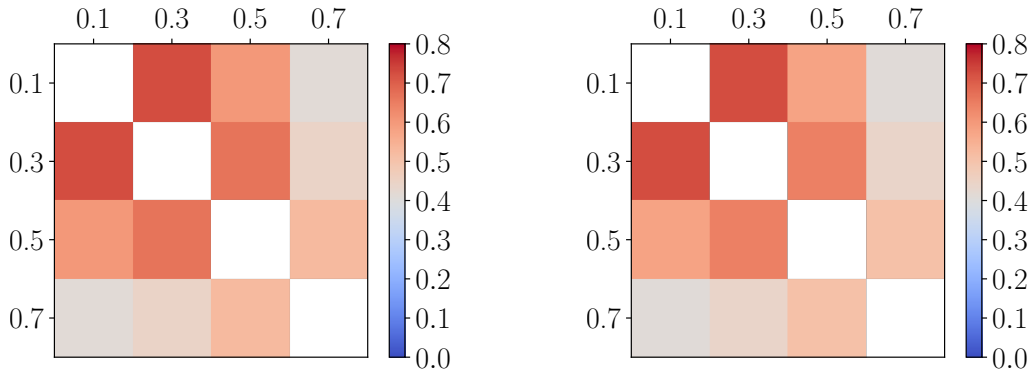


(a) Item penetration, $\gamma = 1.1$ (b) Basket penetration, $\gamma = 1.1$

Figure B.5: Product communities in the first and last time windows and their mapping to the 10 `L3` product categories for $\gamma = 1.1$ with either (a) item-penetration weighting or (b) basket-penetration weighting. We observe similar patterns to the analogous plots from Figure 5.20 for $\gamma = 1.0$.

Varying the threshold for generating unweighted networks

Figure B.6 and Tables B.1–B.2 show that detected communities are relatively robust to different choices of the percentile threshold. In particular, the IterModMax algorithm converges to similar values of the interlayer coupling ω and of the copying probability p for different thresholds. We used GenLouvain (rather than GenLouvainRand) for the results in this section; because of this, we see larger values of $\langle \omega_{\text{out}} \rangle$ and of $\langle p \rangle_t$ than in Section 5.5.1.



(a) Item penetration (b) Basket penetration

Figure B.6: NMI between consensus partitions that we obtained for percentile thresholds in the set $\{0.1, 0.3, 0.5, 0.7\}$. The large similarities between these partitions (especially for threshold values in the set $\{0.1, 0.3, 0.5\}$) suggest that these results are relatively robust to different choices of threshold.

Table B.1: $\langle \omega_{\text{out}} \rangle$ across 50 IterModMax trials with $\gamma = 1$ and different thresholds.

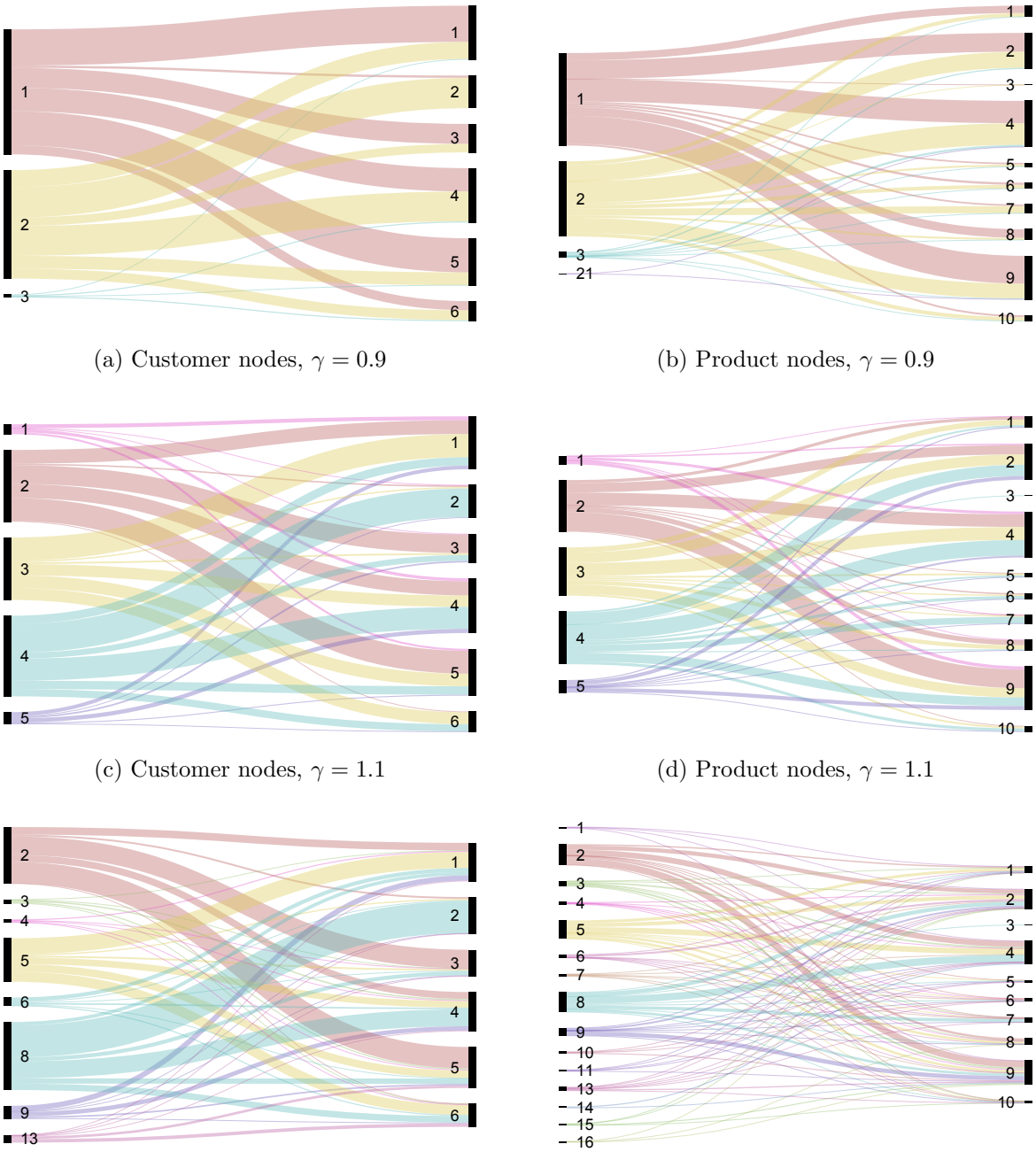
Threshold	Item penetration	Basket penetration
0.1	2.6	2.6
0.3	2.5	2.6
0.5	2.5	2.5
0.7	3.0	2.7

Table B.2: $\langle p \rangle_t$ for the consensus partitions that we obtained from 50 IterModMax trials with $\gamma = 1.0$ and different thresholds.

Threshold	Item penetration	Basket penetration
0.1	0.94	0.94
0.3	0.94	0.94
0.5	0.95	0.94
0.7	0.97	0.97

B.2 Multilevel Networks

Varying the resolution



(a) Customer nodes, $\gamma = 0.9$

(b) Product nodes, $\gamma = 0.9$

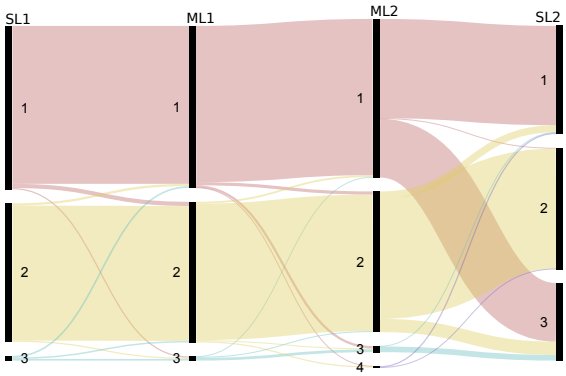
(c) Customer nodes, $\gamma = 1.1$

(d) Product nodes, $\gamma = 1.1$

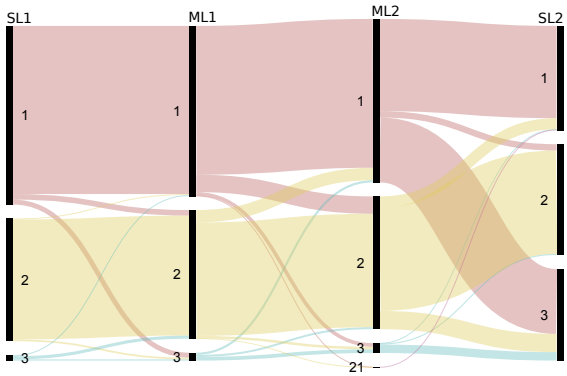
(e) Customer nodes, $\gamma = 1.6$

(f) Product nodes, $\gamma = 1.6$

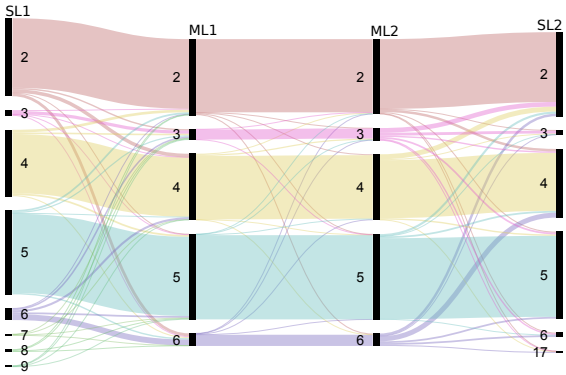
Figure B.7: Comparison of communities in the consensus partitions to the `purchasing-medium` customer segmentation and to `L3` product categories. (Figures 5.24 and 5.25 in Section 5.5.2 show analogous visualisations for $\gamma = 1.0$.)



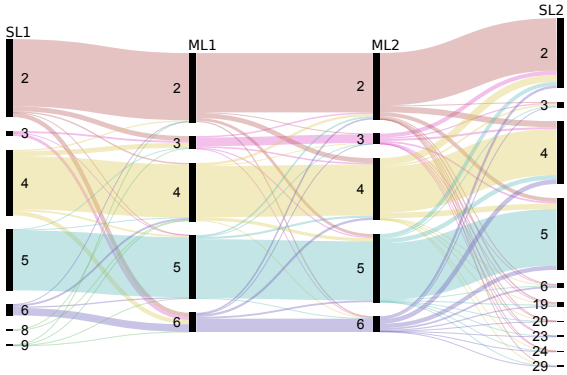
(a) Customer nodes, $\gamma = 0.9$



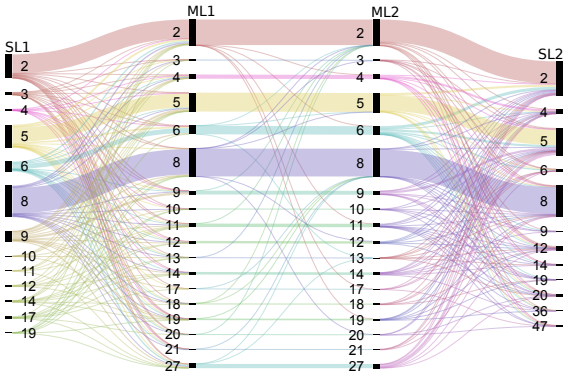
(b) Product nodes, $\gamma = 0.9$



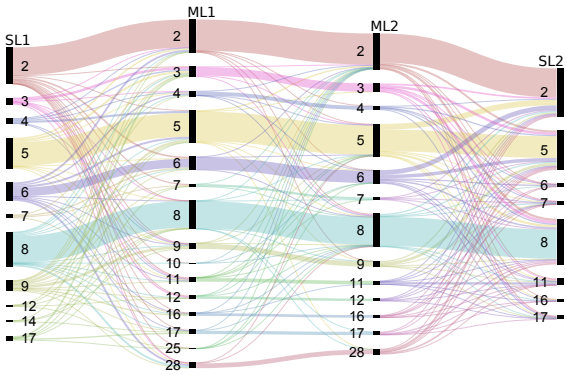
(c) Customer nodes, $\gamma = 1.1$



(d) Product nodes, $\gamma = 1.1$



(e) Customer nodes, $\gamma = 1.6$

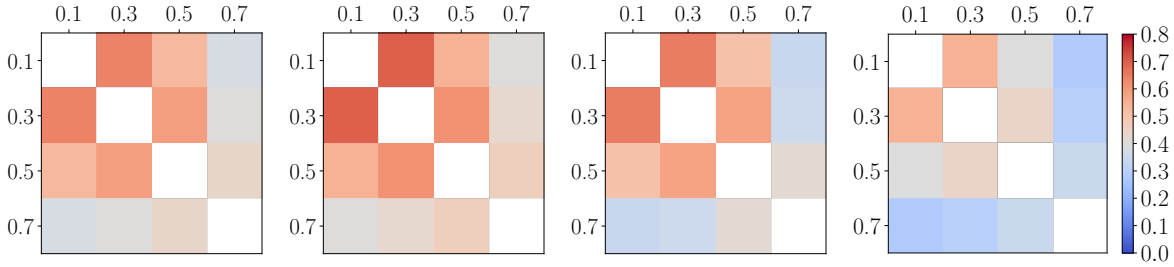


(f) Product nodes, $\gamma = 1.6$

Figure B.8: Alluvial diagrams of consensus communities for $\gamma \in \{0.9, 1.1, 1.6\}$, separately for customer nodes and product nodes. (Figure 5.23 in Section 5.5.2 shows analogous visualisations for $\gamma = 1.0$.) In both panels, the two middle layers correspond to the multilevel consensus partition, and the outer layers correspond to single-layer consensus partitions. (The category level is on the far left, and the product level is on the far right.) For clarity, we do not show communities with fewer than 10 nodes.

Varying the threshold for generating unweighted networks

Figure B.9 and Tables B.3–B.5 show that detected communities are relatively robust to different choices of the percentile threshold.



(a) $\gamma = 0.9$ (b) $\gamma = 1.0$ (c) $\gamma = 1.1$ (d) $\gamma = 1.6$

Figure B.9: NMI between consensus partitions that we obtained for different choices for percentile thresholds in the set $\{0.1, 0.3, 0.5, 0.7\}$. The large similarities between these partitions (especially for threshold values in the set $\{0.1, 0.3, 0.5\}$) suggest that these results are relatively robust to different choices of threshold.

Table B.3: $\langle \omega_{\text{out}} \rangle$ across 50 IterModMax trials using different percentile thresholds and resolution values. For each value of γ, our algorithm converges to similar values of ω when the threshold is in the set $\{0.1, 0.3, 0.5\}$. When the percentile threshold is 0.7, detected multilayer communities have larger persistence (see also Table B.4), and thus so is the value of ω that we calculate.

Threshold	$\gamma = 0.9$	$\gamma = 1.0$	$\gamma = 1.1$	$\gamma = 1.6$
0.1	1.7	1.9	2.4	2.5
0.3	1.8	1.9	2.3	2.4
0.5	1.8	1.9	2.3	2.3
0.7	4.6	3.9	3.6	2.2

Table B.4: Normalised persistence across layers for the consensus partitions that we obtain for different percentile thresholds and resolution values.

Threshold	$\gamma = 0.9$	$\gamma = 1.0$	$\gamma = 1.1$	$\gamma = 1.6$
0.1	0.87	0.85	0.85	0.86
0.3	0.86	0.85	0.86	0.88
0.5	0.89	0.87	0.88	0.91
0.7	0.99	0.97	0.97	0.97

Table B.5: Proportion of customers in segment 2 of the `purchasing-medium` customer segmentation (which is the most upscale of the six segments) that belong to a single community. These results show that this customer segment continues to emerge as one that correlates strongly to detected communities (see also Figure 5.19 in Section 5.5.1).

Threshold	$\gamma = 0.9$	$\gamma = 1.0$	$\gamma = 1.1$	$\gamma = 1.6$
0.1	0.91	0.88	0.87	0.68
0.3	0.92	0.90	0.85	0.68
0.5	0.93	0.90	0.86	0.72
0.7	0.87	0.85	0.82	0.70