

Mapping dynamic brain networks with MEG data using machine learning



Rukuang Huang

supervised by

Mark Woolrich

and

Chetan Gohil

Jesus College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

September 2024

Abstract

This thesis mainly concerns the analysis of non-invasive electrophysiological data, focusing on development of new methodologies for extracting more information from existing datasets. The primary object of interest is the temporal structure in brain network activity, in particular, time-varying functional connectivity (FC), which has been linked to cognition, demographics and disease states. Statistical machine learning and deep learning allows researchers to formulate generative models such as the HMM (Hidden Markov Model) and DyNeMo (Dynamic Network Modes) to give estimates of time-varying FC at the time scales of fast cognition, i.e. on the order of 100 milliseconds, which has been impossible with traditional methods like the sliding window approach. The main contribution of the work in this thesis lies in tackling two of the limitations of these generative models.

Firstly, time-varying estimates of power and FC are calculated under the assumption that they share the same dynamics. But there is no principled basis for this assumption. We propose Multi-dynamic Network Modes (M-DyNeMo), an extension to DyNeMo, that allows for the possibility that the power and the FC networks have different dynamics. Using this new method on magnetoencephalography (MEG) data, we show that the dynamics of the power and the FC networks are not strongly coupled. Using a visual perception task MEG dataset, we also show that the power and FC network dynamics are modulated by the task, and that the coupling in their dynamics changes significantly during task. This new method reveals novel insights into evoked responses and ongoing activity that previous methods fail to capture, challenging the assumption that power and FC share the same dynamics.

Secondly, existing methods assume the same network, or set of networks, are shared by all recording sessions, i.e. the networks are estimated at the group level. This is an unrealistic assumption as functional brain activity is known to possess significant heterogeneity and these methods do not allow for the discovery of, nor benefit from, subpopulation structure in the data. We propose the use of embedding vectors (c.f. word embedding in Natural Language Processing) to explicitly model individual sessions while inferring networks across a group. This vector is effectively a “fingerprint” for each session, which can cluster

sessions with similar functional networks together in a learnt embedding space. We apply this approach to estimate time-varying FC, using the HMM, to model individual sessions in neuroimaging data. We call this approach HIVE (HMM with Integrated Variability Estimation). Using simulated data, we show that HIVE can recover the true, underlying inter-session variability and show improved performance over existing approaches. Using real MEG data, we show the learnt embedding vectors (session fingerprints) reflect meaningful sources of variation across a population (demographics, scanner types, sites, etc). This work provides a powerful new technique for modelling individual sessions while leveraging information available across an entire group.

Overall, this thesis utilises techniques in statistical machine learning and deep generative models to extract more information from neuroimaging data. This body of work provides ways to gain extra insights in cortical network dynamics from large-scale and diverse datasets. In addition to those in methodology, significant contributions were made to the software package *osl-dynamics* that allows models to be trained and analysed efficiently and reproducibly.

Acknowledgements

I would like to express my deepest gratitude to my supervisor Mark Woolrich. Mark always gave guidance, encouragement, and unwavering support for my crazy (often unrealistic) ideas throughout my doctoral journey. I cannot imagine what this journey would have been like without Mark holding the torch while I was exploring in the tunnel.

I am forever indebted to my supervisor Chetan Gohil for the patience and rapid replies to my constant message bombardments. Chet's meticulous attention to detail and commitment to excellence have pushed me to reach new heights in my research (and to write clean codes).

I would also like to thank all members of OHBA for the insightful discussions and debates during coffee breaks and over a pint at the Star. A special thank you to the administrative staff, whose assistance made my academic journey smoother. Your efforts often go unnoticed but are deeply appreciated.

Finally, I would like to thank my parents and the rest of my family who have given their full support to my studies, without which my achievements would have been impossible.

Contents

1	Introduction	1
1.1	Recording signals from the brain	2
1.2	Networks and functional connectivity	3
1.3	The sliding window approach	4
1.4	Magnetoencephalography signal	4
1.4.1	MEG data preprocessing	5
1.4.2	Source reconstruction and parcellation	5
1.4.3	Sign ambiguity and leakage correction	6
1.4.4	Amplitude envelope data	7
1.4.5	Time-delay embedding	8
1.4.6	Frequency domain analysis	9
1.5	MEG datasets used in this thesis	9
1.5.1	Flowchart symbols	11
1.5.2	The Wakeman-Henson dataset	12
1.5.3	The Cam-CAN dataset	12
1.5.4	The MRC MEGUK dataset	14
1.6	Thesis outline	17
2	Modelling and learning techniques	19
2.1	Statistical learning	19
2.1.1	Supervised learning	19

2.1.2	Empirical risk minimisation	21
2.1.3	Linear regression as a probabilistic generative model	22
2.1.4	Unsupervised learning	22
2.1.5	Latent variable models (generative models)	24
2.1.6	Expectation-Maximisation (EM) algorithm	27
2.2	Deep learning	29
2.2.1	Examples of modules	30
2.2.2	Optimisation	33
2.2.3	Variational Autoencoder (VAE)	34
2.2.4	Reparameterisation trick	36
2.3	Statistical hypothesis testing	37
2.3.1	Permutation tests	39
2.3.2	Monte Carlo testing	39
2.3.3	The multiple testing problem	40
2.3.4	Maximum statistic	41
3	Existing models for estimating dynamic brain networks	43
3.1	Modelling brain dynamics with the Hidden Markov Model	43
3.1.1	Formulation of generative model	44
3.1.2	Model training	44
3.2	Extending HMM with Dynamic Network Modes	45
3.2.1	Formulation of generative model	46
3.2.2	Model training	47
3.2.3	Tricks for training DyNeMo	50
3.3	Comparing different generative models	51
3.3.1	Calculation of free energy	52
3.3.2	Calculating model evidence	53
3.4	Discussion	59

4	Multi-dynamic modelling of MEG data	61
4.1	Introduction	61
4.2	Structural equation modelling motivates multi-dynamic modelling. . .	62
4.3	Results	63
4.3.1	Sliding window approach reveals distinct power and FC dy- namics	63
4.3.2	Power and FC have similar spatial patterns but distinct dy- namics	65
4.3.3	Single dynamic DyNeMo is dominated by time-varying power	66
4.3.4	M-DyNeMo inferred networks are reproducible across subsets of data	68
4.3.5	Power and FC dynamics show different timings in evoked net- work response	71
4.3.6	The coupling between power and FC dynamics is modulated by task	72
4.4	Discussion	74
4.5	Conclusions	78
4.6	Methods	78
4.6.1	Dataset	78
4.6.2	Measure of power and FC	79
4.6.3	Sliding window approach for uncoupled dynamics	79
4.6.4	Generative model of M-DyNeMo	80
4.7	Orthogonalisation with PCA	81
4.8	Model training	84
5	Modelling variability in functional brain networks using embed- dings	85
5.1	Introduction	85
5.2	Dual estimation in the traditional HMM approach	87

5.3	The use of embedding vectors	87
5.4	Generating session-specific covariances	88
5.5	Training HIVE	91
5.5.1	Free energy given the state probabilities	93
5.5.2	Choosing the embedding dimension	94
5.5.3	Initialisation of variational parameters	96
5.5.4	Annealing during training	97
5.6	Datasets	98
5.6.1	Simulated datasets	98
5.6.2	Real MEG datasets	98
5.7	Simulation results	100
5.7.1	Simulation study 1	100
5.7.2	Simulation study 2	102
5.7.3	Simulation study 3	102
5.8	Real data results	104
5.8.1	Wakeman-Henson dataset	104
5.8.2	Combined dataset	105
5.8.3	Cam-CAN dataset	107
5.8.4	Choosing embedding dimension on the real datasets	110
5.9	Variability and model complexity	110
5.10	Constraining embedding vectors	113
5.10.1	Simulation example	115
5.11	Discussion	115
5.11.1	Future work	118
5.11.2	Hyper-parameters	119
5.12	Conclusions	119

6 Conclusion

121

A	Data simulation	143
A.1	Simulating the hidden time courses	143
A.2	Simulating the observed data	144
A.3	Simulating data with multiple dynamics	145
A.4	Simulate session-specific covariances	145
B	Supplementary material on multi-dynamic modelling	147
B.1	Simulation results	147
B.1.1	Simulated data	147
B.1.2	Simulation 1: M-DyNeMo performs more accurate inference on data with multiple dynamics	148
B.1.3	Simulation 2: M-DyNeMo can infer accurately on single dy- namics data	149
B.2	More results on real data	149
B.3	Model training details	156
B.3.1	Learning correlation matrices	156
B.3.2	Extra tricks for model training	156
B.4	Post-hoc analysis	157
B.4.1	Re-normalising mode time courses	157
B.4.2	Maximum statistic permutation test for spatial map similarity	157
B.5	Model specifications	158
C	Supplementary material on modelling variability	161
C.1	Dual estimation in HMM	161
C.2	Outlier in the dataset	161
C.3	Possible improvements on algorithm for choosing embedding dimensions	162
C.3.1	Improving the procedure	162
C.3.2	The one standard deviation rule	164
C.4	Model specifications	164

D DyNeMo with Integrated Variability Estimation	167
D.1 Training DIVE	168
D.2 Simulation results	168
E Modelling temporal dynamics with Gaussian Process	173
E.1 Generative model	174
E.2 Inference	175
E.3 Simulation result	177
E.4 Discussion	177

List of Figures

1.1	An example of extracting the amplitude envelope from an oscillation. The black line represents the raw signal, the red line represents the extracted amplitude envelope.	8
1.2	An example of time-delay embedding. The top panel shows a simulated two-channel signal. The black line shows the simulated signal of channel $x(t)$ and the red line shows the that of channel $y(t) = x(t + 0.07)$. The bottom panel shows the covariance matrix of the time-delay embedded data. The off-diagonal blocks of the covariance matrix include the auto-correlations of the original data. .	10
1.3	Example of transforming from time domain to frequency domain with Fourier transform. Left panel shows the simulated oscillation at 5 Hz in the time domain and the right panel show the magnitude (square norm) of the complex Fourier transformed signal in the frequency domain.	11
1.4	Flowchart of the preprocessing steps applied to the Wakeman-Henson dataset. The flowchart was produced by the OSL library. Firstly, a band-pass filter of 0.5-125Hz and a notch filter with 50Hz and 100 Hz were applied. Then the data was downsampled to 250Hz, after which bad segments and channels were removed. Then an ICA step with 40 components was used to remove noise and finally the bad channels are interpolated.	13

1.5	Flowchart of the preprocessing steps applied to the Cam-CAN dataset. The flowchart was produced by the OSL library. Firstly, a band-pass filter of 0.5-125Hz and a notch filter with 50Hz and 100 Hz were applied. Then the data was downsampled to 250Hz, after which bad segments and channels were removed. Then an ICA step with 64 components was used to remove noise and finally the bad channels are interpolated.	15
1.6	Flowchart of the preprocessing steps applied to the MEGUK dataset. The flowchart was produced by the OSL library. Firstly, a band-pass filter of 0.5-125Hz and a notch filter with 50Hz and 100 Hz were applied. Then the data was downsampled to 250Hz, after which bad segments and channels were removed. Then an ICA step with 64 components was used to remove noise and finally the bad channels are interpolated.	16
2.1	Graphical model of a latent variable model. Here $\mathbf{x}_i$ is the observed data, $\mathbf{z}_i$ is the latent variable and ϕ is the model parameters.	25
2.2	Graphical representation of a multi-layer perceptron. $\mathbf{x}$ is the input and $f(\mathbf{x})$ is the output. The dotted arrow represents multiple applications of dense layers and activations.	32
2.3	Graphical representation of a Recurrent Neural Network. Here $\mathbf{x}_1, \dots, \mathbf{x}_T$ are the inputs, $\mathbf{h}_0, \dots, \mathbf{h}_T$ are the hidden states, and $\mathbf{y}_1, \dots, \mathbf{y}_T$ are the outputs. g_1 and g_2 are neural networks themselves and the dotted arrow represents multiple applications of the same procedure.	32

- 2.4 Illustration a variational autoencoder with Gaussian variational distribution.** The encoder with parameters ϕ takes input $\mathbf{x}$, and gives the variational mean $\mu_\psi(\mathbf{x})$ and standard deviation $\sigma_\psi(\mathbf{x})$. Using the reparameterisation trick, a sample ϵ^s from the standard Gaussian distribution is drawn, which is then multiplied with σ_ψ and added to $\mu_\psi(\mathbf{x})$. The resulting sample from the variational distribution is passed through the decoder with parameters ϕ to get an estimate of the original input. Here the red boxes indicate different terms of the loss function. 37
- 3.1 Particle filter finds the correct number of states.** Exact model evidence (blue, covered by orange line), negative free energy (orange) and estimated model evidence with particle filter (green) of the trained HMM is plotted on the y-axis against the number of states. The shades show one standard deviation of 10 folds cross validation. . 59
- 4.1 Sliding window approach reveals distinct power and FC dynamics.** *Results from amplitude time courses (1-45Hz) of 38 brain regions obtained from the MEGUK-38 resting-state data.* a) K-Means state time courses for TV variance (i.e. power) (top row) and TVFC (i.e. correlation) (bottom row), using 2s sliding windows. Different colours indicate activation of different brain states and only the first 20s of the first subject is shown. b) Correlation between power (y-axis) and FC state time courses (x-axis). c) Power maps of each K-Means state are plotted on the top row. Red areas indicate brain areas with higher power (and blue areas lower power) than the average across states. FC maps are plotted on the bottom row. Red edges indicate positive correlations (and blue edges negative correlations) between brain areas. 64

4.2 Power and FC have similar spatial patterns but distinct dynamics. *Results of running M-DyNeMo on the amplitude time courses (1-45Hz) of 38 brain regions obtained from the MEGUK-38 resting-state data.* a) Inferred mode time courses (re-normalised) for power, α (top) and FC, β (bottom). b) Top row shows power spatial modes where red areas show higher (blue areas show lower) activation than average across modes. Middle row shows FC spatial modes where red edges show positive (and blue edges show negative) connectivity. Bottom row shows the FC degree maps where red areas show higher (blue areas show lower) node degree (i.e. centrality) than average across modes. c) Pairwise cosine similarity between power and FC degree spatial maps. Diagonal entries with significantly large values are annotated with asterisks depending on the p-values. d) Correlation profile of mode time courses for within power modes (left), within FC modes (middle) and between power and FC modes. Correlations significantly (at 5% significance level) differ from zero under a maximum statistic permutation test are marked with an asterisk. . 67

4.3 Single dynamic DyNeMo is dominated by time-varying power. *Results from amplitude time courses (1-45Hz) of 38 brain regions obtained from MEGUK-38 resting-state data.* a) Power (top row) and FC (covariance, bottom row) network maps given by re-calculating time varying covariances on M-DyNeMo inferred power mode time course α . b) Power (top row) and FC (covariance, bottom row) maps given by DyNeMo. c) Cosine similarity between re-calculated covariances on M-DyNeMo inferred power mode time course α and DyNeMo inferred covariances for each pair of modes. 69

4.4 **M-DyNeMo inferred networks are reproducible across halves.**
Results from amplitude time courses (1-45Hz) of 38 brain regions obtained from MEGUK-38 resting-state data. a) Power spatial modes given by training M-DyNeMo on subjects 1-32 (top row) and subjects 33-65 (bottom row) independently. The cosine similarity between power maps from each half is also shown for each pair of modes. b) FC spatial modes given by training M-DyNeMo on subjects 1-32 (top row) and subjects 33-65 (bottom row) independently. The ordering of both power and FC modes are matched with cosine similarities between networks from each half, which are shown in the figure for both power and FC networks. 70

4.5 **Evoked responses have different timings in different mode time courses.** *Results from amplitude time courses (1-45Hz) of 38 brain regions obtained from Wakeman-Henson data.* a) Power spatial modes are shown on the top row and FC spatial modes are shown on the bottom row. b) Evoked responses for re-normalised α (top row) and β (bottom row) for different contrasts on different columns, including the average visual response (left), the difference between faces and scrambled faces (middle), and the difference between famous and unfamiliar faces (right). Here different colours represent baseline corrected evoked responses for different modes as contribution to either the overall variance (for α) or the overall correlation (for β). Periods with significant (de-)activation are indicated with a bar for each mode. 73

- 4.6 **Coupling between α and β is modulated by task stimuli.** *Results from amplitude time courses (1-45Hz) of 52 brain regions obtained from MEGUK-52 data.* a) Power (top row) and FC (bottom row) spatial modes are plotted. b) Evoked responses (baseline corrected) for re-normalised α (left) and β (right) to visual stimuli. Different colours show evoked responses for different modes. c) Correlations between α and β during rest (left) and during task (middle) are shown. The right plot illustrates difference between the correlation profiles where significant differences are marked with asterisks. 75
- 4.7 **Generative model of M-DyNeMo.** Historic logits $\theta_{1:t}^1$ and $\theta_{1:t}^2$ are concatenated and passed to the LSTM cell. The outputs of the LSTM cell are the parameters of the distribution from which we sample the future logits θ_{t+1}^1 and θ_{t+1}^2 . The sampled logits are concatenated to the original logit sequences and the process is repeated. The generated sequences of logits are normalised with a softmax transformation and we get the mode time courses $\alpha_{1:T}, \beta_{1:T}$. Diagonal matrices $\mathbf{E}_j$ and correlation matrices $\mathbf{R}_j$ are linearly mixed with $\alpha_{1:T}$ and $\beta_{1:T}$ respectively to give the time-varying power $\mathbf{G}_t$ and FC $\mathbf{F}_t$. Finally the observed data are generated through the multivariate Gaussian distribution with time-varying covariance $\mathbf{G}_t \mathbf{F}_t \mathbf{G}_t$ 82
- 5.1 **Examples of embedding vectors.** a) Example of word embedding vectors. The x-axis encodes whether a word is for animals or non-animals and the y-axis encodes whether the object can fly or not. b) Illustration of possible variability captured by embedding vectors. The x-axis encodes the direction of increasing beta power in motor network whereas the y-axis encodes the direction of increasing peak alpha frequency in visual network. 88

- 5.2 Forward pass of the variability encoding block.** Session-specific information from the session embedding vectors ω^i and state-specific information from the group-level Cholesky vectors $\mathbf{l}_j$ are combined to generate session and state-specific Cholesky vectors $\mathbf{l}_j^i$. Here the **encoder** k is an affine transformation that condenses state-specific information to a lower dimensional space. The **decoder** h is a multi-layer perceptron (MLP) that decodes state and session-specific deviations from the concatenated embedding vectors ξ_j^i . Deviation magnitude d_j^i is generated by an exponential distribution with rate λ_j^i , which is generated by applying an affine transformation f followed by a softmax ρ to the output of h . The deviation map $\mathbf{e}_j^i$ is generated by applying an affine transformation g and normalisation η to the output of h . Finally we get the session-specific Cholesky vector $\mathbf{l}_j^i$ with $\mathbf{l}_j^i = \mathbf{l}_j + d_j^i \mathbf{e}_j^i$ 90
- 5.3 Generative model of HIVE.** State time courses are generated from the group-level transition probability matrix $\mathbf{A}$ through a Markov process. At the same time, session embedding vectors ω^i and group-level Cholesky vectors $\mathbf{l}_j$ are passed to the variability encoding block to give session-specific Cholesky vectors $\mathbf{l}_j^i$. Given the state s_t^i at each time, the observed data is generated through the multivariate Gaussian likelihood. 92
- 5.4 The full HIVE training framework.** The black solid arrows in the plot show connections where gradient flows through during back-propagation and the dashed black arrows show connections where gradient does not flow through. 95

- 5.5 Variability encoding block learns session-specific covariance deviations.** *Results obtained using Simulation study 1.* a) Simulated (top row), HMM-DE inferred (middle row) and HIVE inferred (bottom row) group-level covariances. The columns show covariances of different states. b) Inferred session embedding vectors. Sessions 1 and 2 are annotated. c) Simulated deviations (top row) and deviations from trained generative model of HIVE (bottom row) for state 1 for session 1 (left column) and for session 2 (right column). 101
- 5.6 HIVE infers the underlying subpopulation structure.** *Results obtained using Simulation study 2.* a) Simulated (left) and inferred (right) session embedding vectors. Each point is marked and coloured by the ground truth group assignment, and is annotated with the session number. b) Simulated (top), HMM-DE inferred (middle) and HIVE inferred (bottom) state time courses. c) Session-pairwise cosine distance of simulated (x-axis) against inferred (y-axis) covariances from HMM-DE (left) and HIVE (right). The black line shows the $y = x$ line, which corresponds to optimal performance, and the red line is a fitted line through the points, with the coefficient of determination r^2 displayed in the title. 103
- 5.7 HIVE performs more accurate inference than HMM-DE.** *Results obtained using Simulation study 3.* Accuracy of session-specific covariances (y-axis) are plotted against the number of sessions in the dataset (x-axis) for HMM-DE (blue) and HIVE (orange). Here the number of samples per session is always the same. Correlation between inferred and simulated session-specific covariances is used as a metric for accuracy. 104

- 5.8 HIVE reveals similarities and differences between MEG recordings.** *Results obtained on the Wakeman-Henson dataset.* a) Session-pairwise cosine distance of inferred embedding vectors. Embedding vectors for the same subject are grouped together and have smaller distance between them. b) Session-pairwise L2 distance of inferred covariances from HIVE (left) and HMM-DE (right). c) Clustering metrics - Silhouette score (left), negative Davies-Bouldin score (middle), Calinski-Harabasz score (right), based on subject labels for 10 independent runs of both approaches. Higher values for these metrics indicate better clustering. 106
- 5.9 Embedding vectors reflect meaningful variability regarding scanner/site.** *Results obtained with a combination of the Nottingham and Cam-CAN datasets.* a) PCA-projected embedding vectors coloured with different datasets (left, black stars indicate centroids of embedding vectors from the two datasets) and different age groups (right, older subjects are coloured with lighter colours). b) Subject-pairwise cosine distance of embedding vectors. c) Clustering metrics - Silhouette score (left), negative Davies-Bouldin score (middle), Calinski-Harabasz score (right), based on subject labels for 10 independent runs for both approaches. d) 6 states are inferred with HIVE. The first and second row shows the group level power (red areas show above average and blue areas show below average power across states) and FC maps (top 3% edges are plotted). The third row shows power difference (red areas show higher and blue areas show lower power in Nottingham subjects than Cam-CAN subjects) for each state between centroids of both datasets. The bottom row shows the difference in PSDs (solid lines show the means and shaded areas show one standard deviation across the channels) across frequencies between datasets. . . 108

5.10 **HIVE infers age structure in the population more accurately**

than dual estimation. *Results obtained with the Cam-CAN dataset.*

a) Inferred embedding vectors projected to 2 dimensions with PCA.

Darker-coloured points are younger participants and lighter-coloured ones are older participants. The two red stars show centroids of subjects with age < 36 and > 72 .

b) 20-fold cross validated age prediction

accuracy for HIVE with different embedding dimensions and HMM-DE. 109

5.11 **HIVE infers interpretable age patterns in networks.** *Results*

obtained with the Cam-CAN dataset. 8 states are inferred by HIVE.

Group level power (red areas show above average and blue areas show

below average power across states) and FC maps (top 3% edges are

plotted) are shown as well as the differences in power (red areas show

higher and blue areas show lower power in older subjects than younger

subjects) and PSDs (solid lines show the means and shaded areas show

one standard deviation across the channels) between old and young

subjects. 111

5.12 **Free energy is used to choose n_ω .** For each of the real datasets

results, different models with different candidate embedding dimen-

sions are trained 10 times independently. The free energy on training

data is plotted against embedding dimension. The p-values of the

t-tests are plotted between the boxplots. 112

5.13 **Variability in the data can cause ever-decreasing free energy**

in HMM. a) There is no subject variability in the data and the

variational free energy is plotted against number of states. b) There

is subject variability in the data and the variational free energy is

plotted against the number of states 112

5.14	Sex differences are not visible from PCA-projected embedding vectors. The same set of embedding vectors shown in Figure 5.10a are coloured with the sex of individual. Embedding vectors associated with male and female participants are coloured with blue and orange respectively.	113
5.15	The embedding vector of the masked session can be recovered. Ground truth simulated embedding vectors (left) and inferred embedding vectors are plotted (right). Each point is coloured according to the task ID and annotated by the subject ID. The circled session on the right indicate the masked session during training. . . .	116
5.16	The covariance matrices of the masked session can be recovered. Ground truth simulated (top row) and generated (bottom row) covariance matrices of the masked session are plotted. Each column shows the covariance matrices for each state.	116
B.1	Sliding window approach fails to infer ground truth state time courses. <i>Results on Simulation 1 dataset.</i> a) Dice coefficient of state time courses inferred by sliding window approach using different window lengths. b) The simulated (top row) and sliding window approach inferred (bottom row, with window size of 6 samples) time courses. A segment of 500 samples containing short state transitions is shown.	150

B.2 M-DyNeMo infers time courses more accurately than DyNeMo

when data has multiple dynamics. *Results on Simulation 1 dataset.* a) The simulated (top row), M-DyNeMo inferred (middle row) and DyNeMo inferred (bottom row) time courses for time-varying standard deviations (left column) and correlations (right column). Here different colours indicate activation of different states and only the first 500 samples are plotted. b) DICE coefficients of DyNeMo and M-DyNeMo inferred time courses compared to the simulated ground truth. c) The reconstruction error (Riemannian distance) of the first 500 samples of time varying covariances is plotted against sample on the x-axis. The reconstruction error of DyNeMo is plotted in orange and that of M-DyNeMo is plotted in blue. 151

B.3 M-DyNeMo infers accurately on data with single dynamic.

Results on Simulation 2 dataset. a) From top to bottom row are the ground truth simulated, the DyNeMo inferred, and the two “separate” M-DyNeMo inferred mode time courses. Here different colours indicate activation of different states and only the first 500 samples are plotted. b) The reconstruction error (Riemannian distance) of the first 5000 samples of time-varying covariances is plotted against sample on the x-axis. The reconstruction error of DyNeMo is plotted in orange and that of M-DyNeMo is plotted in blue. 152

B.4 Single dynamic DyNeMo is dominated by time-varying power.

Results from amplitude time courses (1-45Hz) of 38 brain regions obtained from MEGUK-38 resting-state data. Power (top row) and FC (covariance, bottom row) network maps given by re-calculated networks using M-DyNeMo inferred FC mode time course β . The cosine similarity is also shown for each pair of modes between the re-calculated covariances on M-DyNeMo inferred FC mode time course β and standard DyNeMo inferred covariances. 152

B.5 M-DyNeMo reconstructed time varying covariances show greater fluctuation over time. *Results from amplitude time courses (1-45Hz) of 38 brain regions obtained from MEGUK-38 resting-state data. For M-DyNeMo and DyNeMo, the y-axis shows the variability of model reconstructed time varying covariances in terms of Riemannian distance of time varying covariances from the mean covariance for each subject.* 153

B.6 Evoked responses given by DyNeMo. *Results from amplitude time courses (1-45Hz) of 38 brain regions obtained from Wakeman-Henson data. a) Power maps for different modes given by DyNeMo. b) Evoked response analysis on DyNeMo-inferred mode time course. Different colours show baseline corrected evoked response (contribution to overall variance) of different modes.* 153

- B.7 Sliding window approach reveals evidence of different correlation profiles during task versus during rest.** *Results from amplitude time courses (1-45Hz) of 52 brain regions obtained from MEGUK-52 data.* a) Power (top row) and FC (bottom row) maps for each mode of the two state time courses. b) Correlations within α (left) and within β (middle) are shown. The right plot shows the differences in correlations during task versus during rest. Correlations that significantly differ from zero are marked with asterisks depending on the p-values. 154
- B.8 Additional FC degree spatial maps.** *Results from amplitude time courses (1-45Hz) of 38 brain regions obtained from Wakeman-Henson data and of 52 brain regions obtained from MEGUK-52 data.* a) FC degree spatial maps on the Wakeman-Henson dataset (left) and the cosine similarity between power maps and FC degree maps for each pair of power and FC modes (right). b) FC degree spatial maps on the MEGUK-52 dataset (left) and the cosine similarity between power maps and FC degree maps for each pair of power and FC modes (right). 155
- C.1 Outlier in the Wakeman-Henson dataset.** Top: PSD for the prepared data used for training is plotted for different runs of subject 8. Bottom: The session-pairwise L2 distance of the static covariances is plotted for the prepared data used for training. The outlier session is highlighted by exceptionally high distance with all other sessions. . 163
- D.1 Generative model of DIVE vs HIVE.** The generative model of DIVE only differs from that of HIVE in the temporal model. Time courses are generated from either an RNN or a Markov process, which is then used to generate the time-varying observation model parameters for each session. 169

D.2 **Simulation results of DIVE.** Top: Simulated and inferred mode time courses. Different colours show the mode time courses for different modes. Bottom: Simulated and inferred embedding vectors. Here the points are coloured with the ground truth group assignment and are annotated with the session ID. 171

E.1 **DyNeMo with Gaussian Process fails to recover the time courses.** Simulated state time courses (top) and inferred mode time courses (bottom) are plotted. Different colours show activation of different modes 177

List of Tables

B.1	Model specifications of M-DyNeMo on different datasets.	158
B.2	Model specifications of DyNeMo on different datasets.	159
C.1	Model specifications for training HMM on simulated datasets.	164
C.2	Model specifications for training HMM on real datasets.	164
C.3	Model specifications for training HIVE on simulated datasets.	165
C.4	Model specifications for training HIVE on real datasets.	165

Notations and Nomenclatures

Notations used in this thesis are defined by this list unless otherwise stated:

$[n]$ The set $\{1, \dots, n\}$ for $n \in \mathbb{N}$.

$\mathbf{I}_n$ The $n \times n$ identity matrix.

$\mathbf{X}$ A matrix (uppercase in bold).

$\mathbf{x}$ A vector (lowercase in bold).

$\mathbf{X}^T$ The transpose of matrix $\mathbf{X}$.

$\mathbf{X}^{-1}$ The inverse of matrix $\mathbf{X}$.

$\mathbb{E}_q$ Expectation under the distribution q .

$\mathbb{N}$ The set of natural numbers.

$\mathbb{R}$ The set of real numbers.

$\mathbb{R}^n$ The set of n -dimensional real numbers.

$\mathbb{R}^{m \times n}$ The set of $m \times n$ matrices of real numbers.

$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ The multivariate normal distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$.

$f \circ g$ The composition of functions f and g .

$f : S_1 \rightarrow S_2$ A function from set S_1 to set S_2 .

$x \leftarrow y$ Assignment of value y to variable x .

x A scalar.

$x_{1:n}$ The set $\{x_1, \dots, x_n\}$.

$[\mathbf{x}, \mathbf{y}]$ The concatenation of vectors $\mathbf{x}$ and $\mathbf{y}$.

ANN Artificial Neural Network

CPU Central Processing Unit

DIVE DyNeMo with Integrated Variability Estimation

DyNeMo Dynamic Network Modes

EEG Electroencephalography

ELBO Evidence Lower Bound

EM Expectation-Maximization algorithm

ERM Empirical Risk Minimization

FC Functional Connectivity

fMRI Functional Magnetic Resonance Imaging

GMM Gaussian Mixture Model

GPU Graphics Processing Unit

HIVE HMM with Integrated Variability Estimation

HMM-DE HMM with Dual Estimation

HMM Hidden Markov Model

ICA Independent Component Analysis

KL divergence Kullback-Leibler divergence

LCMV Linearly Constrained Minimum Variance

LSTM Long Short-Term Memory

M-DyNeMo Multi-Dynamic Network Modes

MAP Maximum A Posteriori

MEG Magnetoencephalography

MLP Multi-Layer Perceptron

MRI Magnetic Resonance Imaging

NLP Natural Language Processing

PCA Principal Component Analysis

PC Principal Component

PROFUMO Probabilistic Functional Modes

PSD Power Spectral Density

ReLU Rectified Linear Unit

RNN Recurrent Neural Network

ROI Region of Interest

SGD Stochastic Gradient Descent

sMRI structural MRI

TDE Time Delay Embedding

TPU Tensor Processing Unit

TVFC Time-varying Functional Connectivity

TV Time-varying

UKB UK Biobank

VAE Variational Autoencoder

Chapter 1

Introduction

The brain is a vital organ that serves as the centre of the nervous system in most animals, including humans. The human brain, in particular, is a complex network of approximately 90 billion *neurons* - cells that receive and transmit electrochemical signals, and is responsible for a wide range of functions, including cognition ([Nestler et al., 2020, Diamond, 2013]), emotion ([Armony and Vuilleumier, 2013]), memory ([Baddeley, 2007, Eysenck, 2012]), sensation ([Hellier, 2014, Acevedo et al., 2018, Guyton and Hall, 2011]), and motor control ([Guyton and Hall, 2011]). Indeed, the brain is sometimes referred to as the last frontier of biology ([Squire et al., 2012]) and paradoxically, the study of the brain and nervous system (neuroscience) might be an ultimate challenge for the human brain: Does the brain have the capacity to fully understand something as complex as itself ([Pinel, 2009])?

This thesis endeavours to make an incremental, yet significant step in the area of computational neuroscience. Leveraging mathematical and statistical models, alongside recent advances in computer science, machine learning and deep learning, it seeks to understand the brain and its operations through the analysis of *functional neuroimaging data*.

1.1 Recording signals from the brain

The brain is a complex system and there are enormous amount of neurons and connections between them. As a results, it is impossible to directly observe its activity. However, it is possible to record the activity of the brain by measuring its electrical, magnetic, or metabolic activity. The resulting data are called functional neuroimaging data. Methods for recording these data can be broadly divided into two categories: invasive and non-invasive methods. Invasive methods include electrocorticography (ECoG), intracranial EEG (iEEG), etc. These are critical tools for studying the brain, especially in the context of epilepsy treatment and research ([Zhang et al., 2021, Kramer et al., 2010]). These techniques involve placing electrodes directly on or in the brain to record electrical activity ([Miller et al., 2009]) and can record data with high signal-to-noise ratio (SNR). However, these are invasive procedures that carry risks associated with surgery and hence are naturally constrained in quantity of data and diversity of population.

This thesis concerns the analysis of non-invasive modalities (e.g. electroencephalography - EEG, magnetoencephalography - MEG, functional magnetic resonance imaging - fMRI, etc) of brain signal recordings. Despite having a lower SNR, the non-invasiveness of this type of recordings allows scanning of a much wider population and thus enables massive datasets to be collected in diverse experimental setups. For instance, the UK Biobank (UKB) consists of fMRI recordings of around 40,000 individuals, and the Cam-CAN dataset ([Taylor et al., 2017]) has both MEG and fMRI recordings for approximately 700 individuals. This is particularly important for the study of brain disorders and diseases ([Mandal et al., 2018, Kohl et al., 2024]) because risk factors and biomarkers can be ignored due to lack of statistical power in small datasets, as well as for the development of brain-computer interfaces and neurofeedback systems ([McFarland and Wolpaw, 2017]), where model training is often extremely data-hungry.

1.2 Networks and functional connectivity

Neuroscientists nowadays believe that cognitive tasks are performed not by individual brain regions working in isolation, but by networks of several brain areas which are said to be functionally connected ([Fair et al., 2009, Park and Friston, 2013, Petersen and Sporns, 2015]). A number of studies have shown that the brain recruits functional networks during task ([Fries, 2015, Quinn et al., 2018, Luckhoo et al., 2012, Cole et al., 2014]), and at rest ([Beckmann et al., 2005, Brookes et al., 2011]). As opposed to structural connectivity, which refers to the physical connections between brain regions, *functional connectivity* (**FC**) is defined as any measure of statistical dependency between spatially remote regions ([Friston, 1994]). FC is typically measured in terms of correlation or coherence ([Bowyer, 2016]) between the time series of different brain regions and therefore is an empirical measure of the temporal relationship between regions, without indicating how the temporal co-variation is mediated ([Hutchison et al., 2013, Friston, 2011]). Although the neural underpinnings of FC are still unsolved, evidence suggests that FC is related to underlying neural activity (See for example [Nir et al., 2008]) and significant differences have been found in FC between healthy and diseased states ([Greicius, 2008, Heine et al., 2012, Menon, 2011]).

Traditionally, it was typically assumed that the strength of interactions between regions is constant over time (stationary) during resting-state scans, but given the dynamic nature of the brain ([Rabinovich et al., 2012]), evidence suggesting that FC changes with active tasks ([Esposito et al., 2006, Fornito et al., 2012]), and the fact that within-subject variation in FC has been shown to considerable ([Honey et al., 2009, Meindl et al., 2010, Van Dijk et al., 2010]), an increasing important perspective is how functional networks dynamically fluctuates over time. This is the main goal of this thesis: to develop and extend computational models that can capture the fast temporal dynamic nature of functional networks in the brain.

1.3 The sliding window approach

One of the most common approaches to compute time-varying FC (TVFC) is the sliding window approach ([Allen et al., 2014, Chang and Glover, 2010]). During this approach, a time window of a fixed number of time points (called the *window size*) is selected and FC is calculated for the selected window with, for example, correlation or coherence. Next the window is moved by a fixed number of time points (called the *step size*) and the process is repeated. This results in a time series of FC measures. Clustering methods such as the K-means clustering algorithm (See Example 2.2) are followed to identify recurring and transient patterns of FC states ([Allen et al., 2014]).

Despite its success in studying TVFC, the sliding window approach has its limitations. The choice of window size and step size are arbitrary and can have a significant impact on the interpretation of results. On one hand, a large window size gives more robust estimation of TVFC, but it reduces sensitivity to short-lived changes in FC (e.g. transient bursts). On the other hand, a small window size can be more sensitive to intrinsic noise in the data, resulting in spurious and unstable estimates.

1.4 Magnetoencephalography signal

Although the tools and methods introduced in this thesis can in theory be applied to other modalities of functional data, we focus on results from MEG datasets due to its excellent time resolution ($\sim 1\text{kHz}$), which allows temporal dynamics to be studied at the time scales of fast cognition (on the order of 100 milliseconds), and good data quality compared to EEG. In this section we give a very brief description of the MEG signal and the steps for processing data before they are analysed.

During synaptic transmission, electrical currents are generated in the brain which, according to Maxwell's equations, produce magnetic fields. These magnetic

fields can be measured by extremely sensitive sensors (called SQUIDS) and that record time series data of magnetic field measurements.

1.4.1 MEG data preprocessing

The strength of the magnetic fields induced by synchronised neuronal currents is considerably weaker than the ambient magnetic field and can be easily distorted (e.g. by a nearby vibration, power line, muscle movement, etc). Therefore, it is paramount for MEG data to be preprocessed in order to remove noise and artefacts before any analysis. Typical preprocessing steps include:

- **Band-pass filtering:** Only signal within a certain frequency range is kept to remove high-frequency noise and low-frequency drifts.
- **Notch filtering:** Signal at a specific frequency (e.g. 50 Hz) is discarded to remove known sources of noise, such as power lines.
- **Bad segment/channel removal:** Time segments or channels of data with excessive noise are removed.
- **Independent component analysis (ICA):** Signal is decomposed into independent components and components that are likely to be noise are removed.
- **Downsampling:** Data is downsampled to reduce computational load.

1.4.2 Source reconstruction and parcellation

MEG sensors measure the signal outside of the head (called *sensor space*), but in many cases the goal is to infer the underlying neural activity that generates the signal. This requires estimating the signals in the brain (called *source space*) that give rise to the measured signals on the scalp. This process is called *source reconstruction* (also called *source localisation*) and is a challenging problem due to the ill-posed nature of the inverse problem - as there are much more source locations than there

are sensors, see [Ou et al., 2009]. Several methods have been proposed to solve this problem, including the minimum-norm estimation ([Ou et al., 2009]) and the linearly constrained minimum variance (**LCMV**) beamformer ([Van Veen et al., 1997, Van Veen and Buckley, 1988]).

Before solving the inverse problem, a step called *coregistration* is done to align the MEG sensors with the structural MRI of the subject so that the estimated source signals can be projected back to the brain. In this thesis, structural data were coregistered with the MEG data using an iterative close-point algorithm, and digitised head points acquired with a Polhemus pen were matched to individual subject’s scalp surfaces extracted with FSL’s BET tool ([Jenkinson et al., 2005, Smith, 2002]). The nose was not included in the coregistration as the structural MRI images were defaced.

Source reconstruction often gives time series signal for thousands of dipoles (voxels) in the brain, which can result in a debilitating number of statistical tests, or can be computationally infeasible to handle, depending on the analysis being done. Therefore, a common next step is to parcellate the brain into regions of interest (ROIs) and aggregate the signal of all dipoles within each ROI. The aggregation process in MEG is to use the first principle component. The ROIs are typically defined using an anatomical (e.g. [Tzourio-Mazoyer et al., 2002]) or a functional (e.g. [Yeo et al., 2011]) atlas.

1.4.3 Sign ambiguity and leakage correction

Despite the regularisation and constraints used in source reconstruction techniques like the beamformer, the estimated source signals are still subject to sign ambiguity and leakage.

Sign ambiguity refers to the fact that the polarity of the estimated dipole source is arbitrary and can be flipped. This means that the signs of the estimated source signal are arbitrary across different recording sessions and subjects, which poses a

challenge for group level analysis. [Vidaurre et al., 2016] proposed a method to align the signs across sessions by greedily searching for the sign flipping that maximises the agreement between the estimated source signals from different sessions.

Leakage is again a result of the ill-posed nature of the inverse problem and refers to the fact that the estimated source signals are not spatially confined to the true source location, but can leak into neighbouring regions. This is particularly problematic when the goal is to estimate the FC between regions, as the leakage can lead to spurious correlations. Several methods have been proposed to correct for leakage. The symmetric multivariate leakage reduction method proposed by [Colclough et al., 2015] was used in this thesis.

1.4.4 Amplitude envelope data

The raw source reconstructed MEG signals are typically very noisy and difficult to analyse. Therefore, it is common to extract the *amplitude envelope* (or power envelope) of the MEG signal and use it as input for further analysis. Historically, variations in the amplitude of the MEG signal have been used to study brain activity ([Baker et al., 2014, O'Neill et al., 2015]) and the correlation between amplitude envelope signals has been used as a measure of FC. The amplitude envelope is typically extracted by applying a Hilbert transform ([Feldman, 2011]) to the signal and taking the modulus of the resulting complex signal.

In Figure 1.1 we show an example of extracting the amplitude envelope from an oscillation. The black line represents the raw signal

$$\begin{aligned} x(t) &= a(t) \sin(2\pi f_1 t) \\ a(t) &= \sin(2\pi f_2 t), \end{aligned} \tag{1.1}$$

where $f_1 = 2$ Hz, $f_2 = 0.1$ Hz and the red line represents the extracted amplitude envelope.

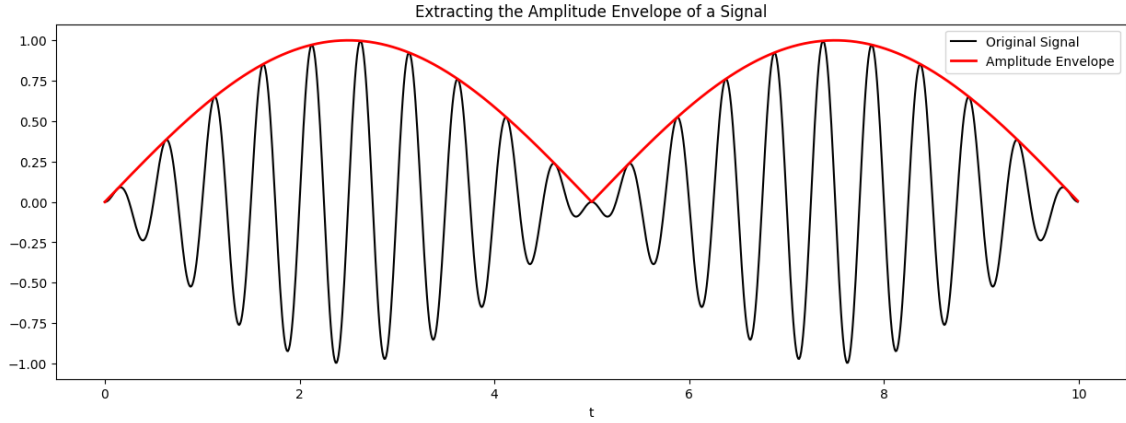


Figure 1.1: **An example of extracting the amplitude envelope from an oscillation.** The black line represents the raw signal, the red line represents the extracted amplitude envelope.

1.4.5 Time-delay embedding

Instead of variations in the amplitude of the MEG signal, the spectral properties of the MEG signal are also of interest and have been widely used to study brain activity and diseases ([Cheyne, 2013, David et al., 2006, Fernández et al., 2006]). The spectral properties of the MEG signal can be extracted by the so-called *time-delayed embedding* (TDE) method ([Vidaurre et al., 2018b, Gohil et al., 2022]). This involves adding extra channels which are time-lagged versions of each channel in the original data. The covariance matrix of the time-delay embedded data includes cross/auto-correlations of the original data in the off/on-diagonal blocks. As the auto-correlations include information about the spectral properties of the data, this transformation allows downstream models to capture and focus on the spectral properties of the data.

In Figure 1.2 we show an example of the effect of time-delay embedding with 15 embeddings (± 7 lags) on a two-channel signal simulated from:

$$\begin{aligned} x(t) &= a(t) \sin(2\pi f_1 t), a(t) = \sin(2\pi f_2 t), \\ y(t) &= x(t + 0.07), \end{aligned} \tag{1.2}$$

where $f_1 = 2$ Hz, $f_2 = 0.1$ Hz and $t = \{10^{-2} \times (i - 1) : i \in [500]\}$. Notice $y(t)$ is a

time-lagged version of $x(t)$ by 7 time steps and we can clearly see that the covariance entry is the highest when the lag is 7, which is the time lag between $x(t)$ and $y(t)$. In this thesis, we typically use TDE to allow us to capture the spectral properties of the original data (e.g. Chapter 5). The exception to this is Chapter 4, where we take the alternative approach of modelling amplitude/Hilbert envelope time courses.

1.4.6 Frequency domain analysis

The raw MEG signals are time series data, from which we can observe the change of power (magnetic field) over time, hence we say the raw MEG signals are in the *time domain*. Rather than the magnitude at different time points, one may be interested in the frequency content, i.e. the magnitude of the signal at different frequency bands. This requires the transformation from the time domain to the *frequency domain*. This can be done with different transformations, an example of which is the Fourier transform, which converts a function in time into a complex-valued sum or integral of sine waves of different frequencies, with amplitude and phase information. Figure 1.3 shows an example of transforming a simulated oscillation at 5 Hz from time domain to frequency domain. Here the original oscillation is simulated by

$$x(t) = e^{-0.1\pi(t-5)^2} \sin(2\pi ft), \quad t \in \{10^{-3} \times (i - 1) : i \in [1000]\}, \quad (1.3)$$

where $f = 5$ Hz. Here we have omitted the phase information of the Fourier transformed signal and only plotted the square norm of the complex signal. A clear spike at 5 Hz is clearly picked up by the Fourier transformation.

1.5 MEG datasets used in this thesis

In this thesis we use several publicly available MEG datasets to validate the developed models as well as to demonstrate the utility and advantages of the models. In this section we give the details of dataset availability and the data preparation

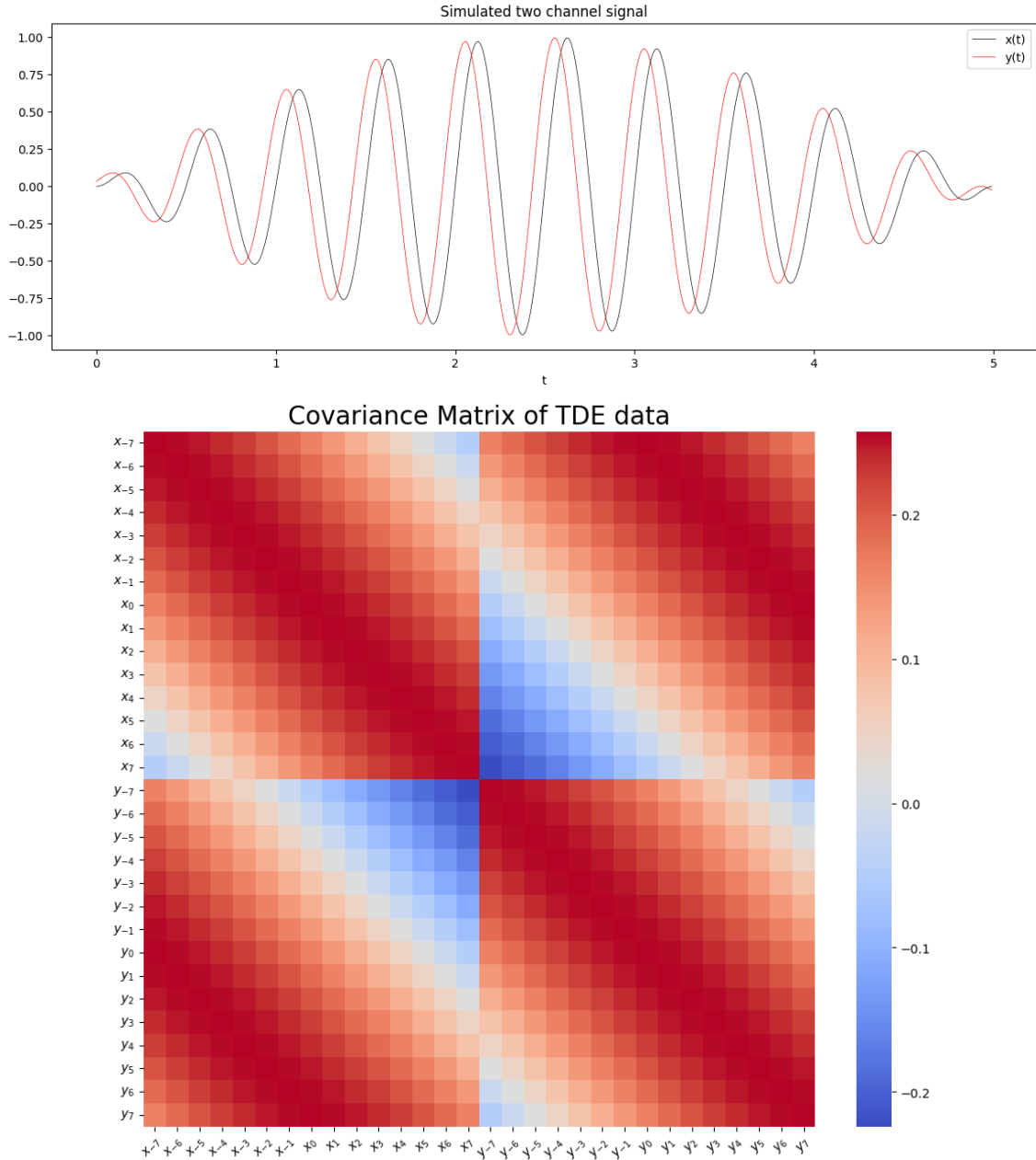


Figure 1.2: **An example of time-delay embedding.** The top panel shows a simulated two-channel signal. The black line shows the simulated signal of channel $x(t)$ and the red line shows the that of channel $y(t) = x(t + 0.07)$. The bottom panel shows the covariance matrix of the time-delay embedded data. The off-diagonal blocks of the covariance matrix include the auto-correlations of the original data.

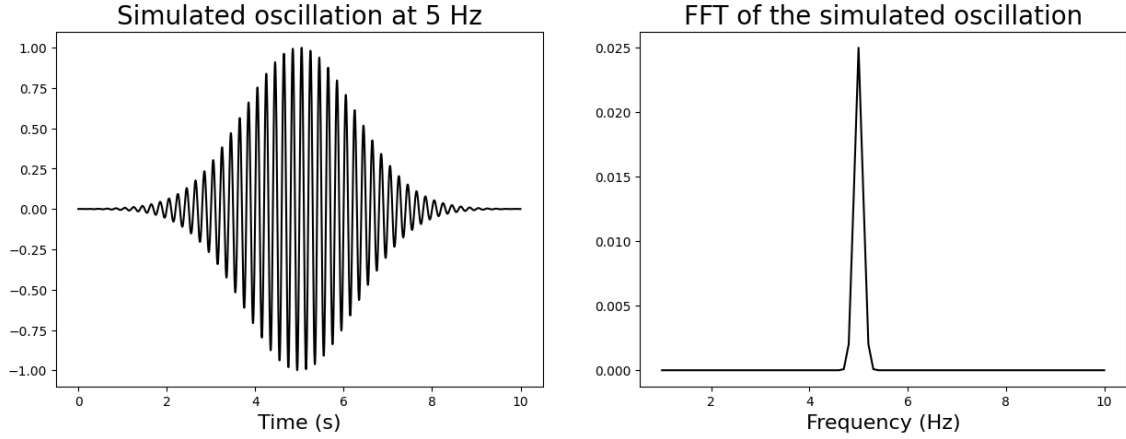


Figure 1.3: **Example of transforming from time domain to frequency domain with Fourier transform.** Left panel shows the simulated oscillation at 5 Hz in the time domain and the right panel show the magnitude (square norm) of the complex Fourier transformed signal in the frequency domain.

steps applied to the data before analysis. All the data preprocessing, source reconstruction and parcellation steps were done using the OHBA software library (OSL, [Quinn et al., 2022]).

1.5.1 Flowchart symbols

Here we introduce the symbols in the flowchart for preprocessing pipeline:

- **pick**: Only pick certain channels.
- **filter**: Band-pass filter the data at a frequency range.
- **notch_filter**: Notch filter the data at certain frequencies.
- **resample**: Resample the data at a different frequency.
- **bad_segments**: Annotate bad segments of the data.
- **bad_channels**: Annotate bad channels of the data.
- **ica_raw**: Perform ICA on the data.
- **ica_autoreject**: Reject ICA components automatically.

- **interpolate_bads**: Interpolate bad segments.

1.5.2 The Wakeman-Henson dataset

This is a visual task MEG dataset collected at 1kHz using an Elekta Neuromag Vectorview 306 scanner. In this dataset, each of the 19 participants (11 male, 8 female, aged 23-37 years) were scanned 6 times, during which they were presented with an image of famous, unfamiliar, or scrambled faces. In each scanning session, there were approximately 200 trials evenly split between the three conditions. The MaxFiltered dataset is publicly available and is referred to as the **Wakeman-Henson** dataset in this thesis. We refer the reader to the original paper ([Wakeman and Henson, 2015]) for more details on the experimental design and data collection.

Data preparation

The MaxFiltered data were preprocessed with the pipeline shown in Figure 1.4. Then the data were coregistered and source reconstructed onto an 8mm isotropic grid using an LCMV beamformer. Voxels were then parcellated into 38 anatomically defined ROIs, which is called the Giles parcellation, used in [Gohil et al., 2024a, Gohil et al., 2024b], before leakage reduction and sign flipping were applied.

1.5.3 The Cam-CAN dataset

This is an MEG dataset collected at University of Cambridge, UK as part of the Cam-CAN project ([Taylor et al., 2017]). This is a large scale dataset of 612 participants aged 18-88 years. The MEG data was collected using a 306-channel Elekta Neuromag Vectorview system at a sampling frequency of 1kHz with a highpass filter of 0.03Hz. In this thesis, we only used the resting-state recordings, each of which are approximately 9 minutes, and participants were instructed to keep their eyes closed.

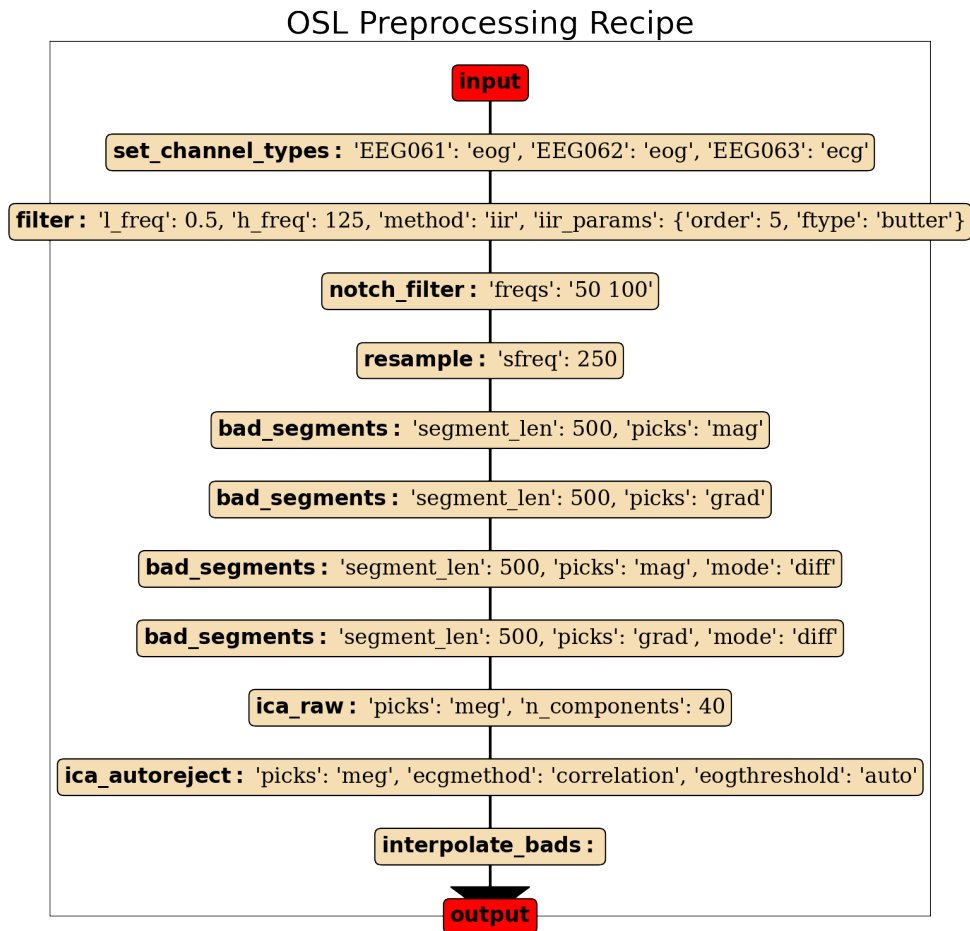


Figure 1.4: **Flowchart of the preprocessing steps applied to the Wakeman-Henson dataset.** The flowchart was produced by the OSL library. Firstly, a band-pass filter of 0.5-125Hz and a notch filter with 50Hz and 100 Hz were applied. Then the data was downsampled to 250Hz, after which bad segments and channels were removed. Then an ICA step with 40 components was used to remove noise and finally the bad channels are interpolated.

The dataset is publicly available and is referred to as the **Cam-CAN** dataset in this thesis.

Data preparation

Similar to the Wakeman-Henson dataset, MaxFiltered data are available for the Cam-CAN dataset. The preprocessing steps are summarised in Figure 1.5. Data were coregistered and source reconstructed onto an 8mm isotropic grid using an LCMV beamformer. Voxels were then parcellated into the same 38 anatomically defined ROIs as the Wakeman-Henson dataset, before leakage reduction and sign flipping were applied.

1.5.4 The MRC MEGUK dataset

In this thesis, only the Nottingham subset of this dataset was used, including eyes-open resting-state recordings of 65 subjects, eyes-closed resting-state recordings of 64 subjects and visual task recordings of 67 subjects. We refer to this dataset as the **MEGUK** dataset.

Data preparation

The preprocessing steps are summarised in Figure 1.6. Data were coregistered and source reconstructed onto an 8mm isotropic grid using an LCMV beamformer. Two atlases were used for parcellating the MEGUK dataset. One is the Giles parcellation with 38 ROIs used in Section 1.5.2 and 1.5.3, and the other is the Glasser parcellation with 52 ROIs ([Kohl et al., 2023]). The resulting data are referred to as **MEGUK-38** and **MEGUK-52** according to the number of parcels in the parcellation.

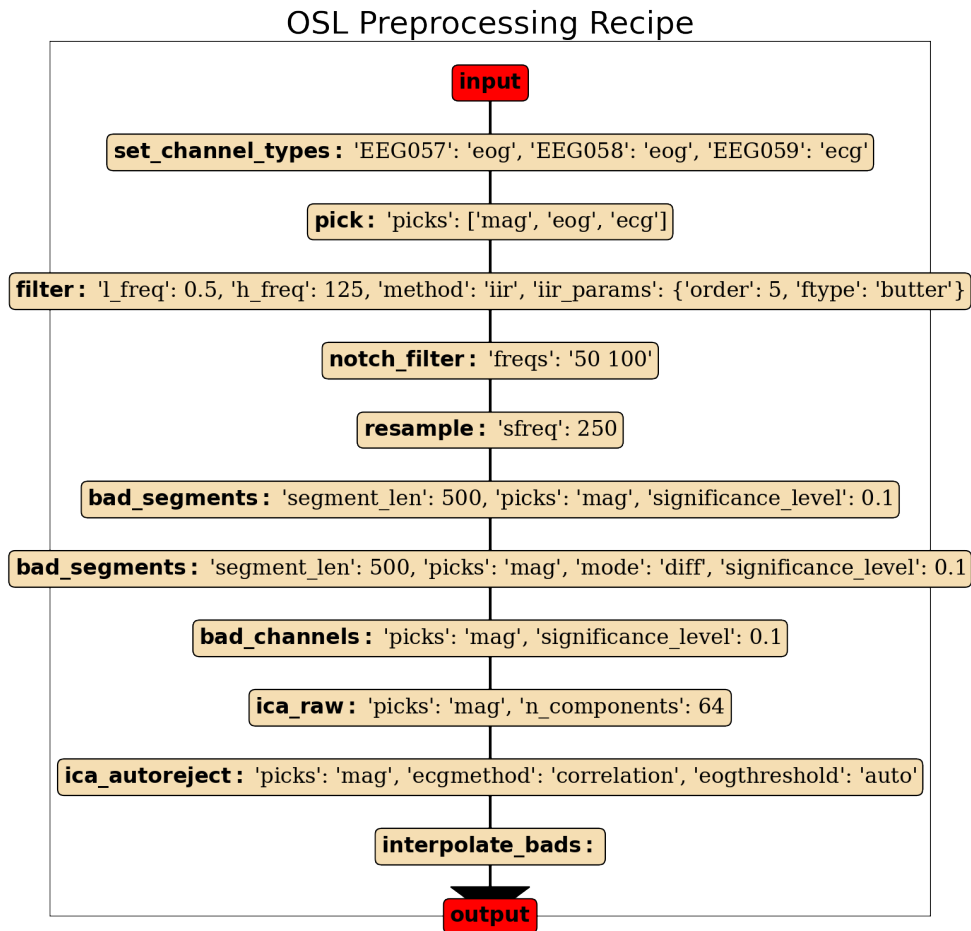


Figure 1.5: **Flowchart of the preprocessing steps applied to the Cam-CAN dataset.** The flowchart was produced by the OSL library. Firstly, a band-pass filter of 0.5-125Hz and a notch filter with 50Hz and 100 Hz were applied. Then the data was downsampled to 250Hz, after which bad segments and channels were removed. Then an ICA step with 64 components was used to remove noise and finally the bad channels are interpolated.

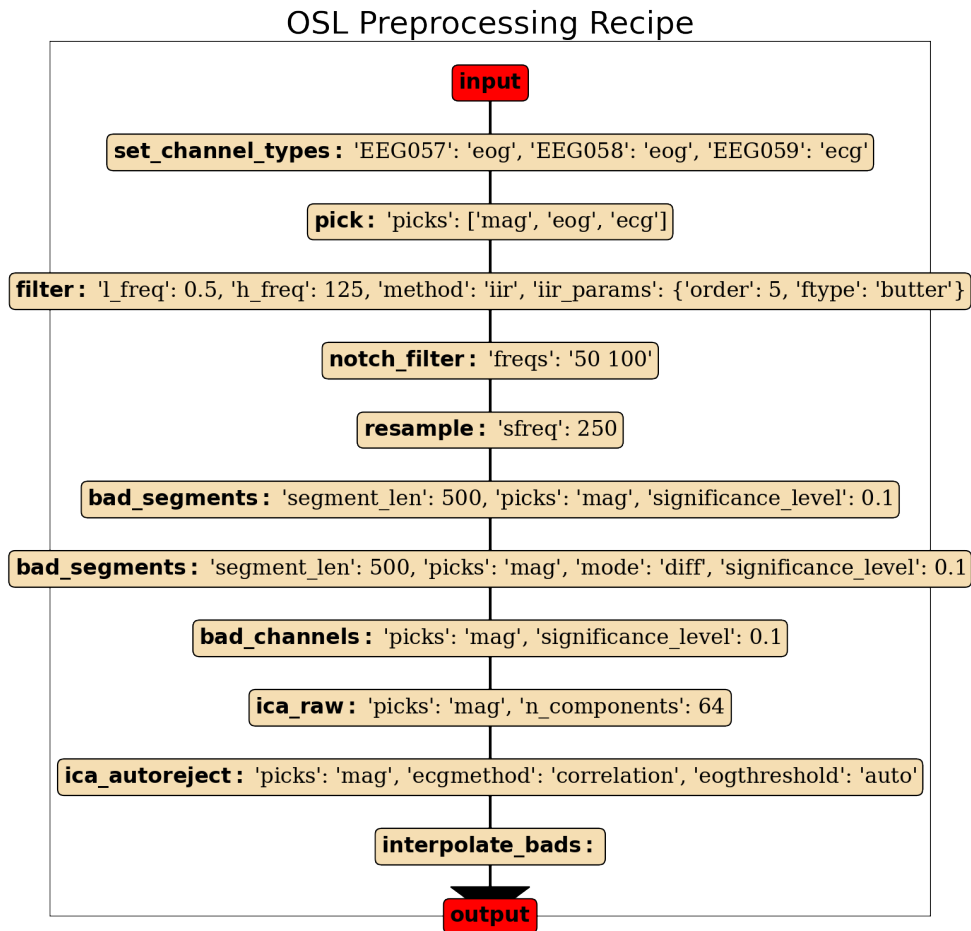


Figure 1.6: **Flowchart of the preprocessing steps applied to the MEGUK dataset.** The flowchart was produced by the OSL library. Firstly, a band-pass filter of 0.5-125Hz and a notch filter with 50Hz and 100 Hz were applied. Then the data was downsampled to 250Hz, after which bad segments and channels were removed. Then an ICA step with 64 components was used to remove noise and finally the bad channels are interpolated.

1.6 Thesis outline

In this thesis, there are two main questions we want to answer regarding dynamic FC:

- Do power and FC share the same dynamics? Specifically, do the amplitudes of oscillatory brain activity and the magnitude of interactions between regions of the brain fluctuate with the same temporal dynamics?
- How do we model the variability in datasets or even between datasets? For example, how can we model the way in which functional brain networks vary between different people?

We aim to build machine learning models to help us answer these two questions. In **Chapter 2**, we give a brief description of the tools and methodology behind the developed models in this thesis as well as the necessary statistical theory to help us draw conclusions and interpretations. In **Chapter 3**, we describe two existing model-based methods for estimating TVFC and discuss limitations associated with these methods. Furthermore, we introduce attempts in quantitatively comparing generative models. In **Chapter 4**, we tackle the first question by introducing a model that can infer potentially uncoupled power and FC dynamics. We show how this model can give additional insights on brain dynamics and how it relates to cognition. In **Chapter 5**, we focus on the second question. We show how the use of embedding vectors capture variability between sessions and how we can extract meaningful sources of variation with the proposed framework. Finally, in **Chapter 6**, we conclude and summarise the implications of the work in this thesis.

Chapter 2

Modelling and learning techniques

In this chapter, an overview of the underlying theories of the models discussed in later chapters is given. This includes statistical learning, deep learning and hypothesis tests.

2.1 Statistical learning

The goal of *statistical learning* (also used interchangeably with *machine learning*) is to build models and algorithms that can understand and make predictions from data. The data can be in the form of images, text, time series (which is the focus of this thesis), etc. Statistical learning falls into different categories, including but not limited to supervised learning and unsupervised learning, depending on the nature of the data and the learning task.

2.1.1 Supervised learning

From the perspective of statistical decision theory, it is assumed that the observed data from the real world are generated by some **unknown** probability distribution. In the context of supervised learning, the data of interest is the doublet $(\mathbf{x}, y) \in \mathcal{X} \times \mathcal{Y}$, where $\mathbf{x} \in \mathcal{X}$ denote a real-valued random input vector (also called *a feature*

or an independent variable) and $y \in \mathcal{Y}$ denote a real-valued random output variable (also called a *response* or a *dependent variable*), drawn from the unknown joint distribution $p(\mathbf{x}, y)$ on the space $\mathcal{X} \times \mathcal{Y}$. We aim to find a function $\hat{f} : \mathcal{X} \rightarrow \mathcal{Y}$ so that the *expected risk* is minimised:

$$\hat{f} = \arg \min_f \mathbb{E}_{p(\mathbf{x}, y)} L(y, f(\mathbf{x})), \quad (2.1)$$

where $L : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$ is the pre-defined *loss function* that penalises incorrect predictions made by f . The intuition is to find the function that performs the best on “average”. By the tower rule, Equation (2.1) can be written as

$$\hat{f} = \arg \min_f \mathbb{E}_{p(\mathbf{x})} \mathbb{E}_{p(y|\mathbf{x})} L(y, f(\mathbf{x})). \quad (2.2)$$

Hence it suffices to minimise the point-wise loss function:

$$\hat{f}(\mathbf{x}^*) = \arg \min_f \mathbb{E}_{p(y|\mathbf{x}=\mathbf{x}^*)} L(y, f(\mathbf{x})) \quad \forall \mathbf{x}^* \in \mathcal{X}, \quad (2.3)$$

which means some of the supervised learning algorithms (discriminative models) can be thought of as estimating the conditional data distribution $p(y|\mathbf{x})$.

For the purpose of avoiding overfitting, interpretability and computational efficiency, we often restrict the class of functions f to a parametric family of functions f_θ with parameters θ . Then the problem becomes finding the best parameters $\hat{\theta}$ that minimises the expected loss function. The simplest example is the ordinary least squares linear regression, shown in Example 2.1.

Example 2.1 (Linear regression) In linear regression ([Montgomery et al., 2021]), the function f is assumed to be linear and has the form:

$$f_\theta(\mathbf{x}) = \mathbf{x}^T \boldsymbol{\theta}, \quad (2.4)$$

and a common choice of the loss function is the squared loss:

$$L(y, f_{\boldsymbol{\theta}}(\mathbf{x})) = (y - f_{\boldsymbol{\theta}}(\mathbf{x}))^2. \quad (2.5)$$

We can show that the optimal parameter $\hat{\boldsymbol{\theta}}$ that minimises the expected loss function is given by

$$\hat{\boldsymbol{\theta}} = \arg \min_{\boldsymbol{\theta}} \mathbb{E}(y - \mathbf{x}^T \boldsymbol{\theta})^2 = [\mathbb{E}(\mathbf{x} \mathbf{x}^T)]^{-1} \mathbb{E}(\mathbf{x} y), \quad (2.6)$$

where the expectation is taken over the unknown joint distribution $p(\mathbf{x}, y)$.

2.1.2 Empirical risk minimisation

As aforementioned, the true data distribution $p(\mathbf{x}, y)$ is unknown and thus minimising the true expected risk is infeasible in practice. Instead, with observed data $\{\mathbf{x}_i, y_i : i \in [n]\}$, we can use the empirical distribution $\hat{p}(\mathbf{x}, y) = \frac{1}{n} \sum_{i=1}^n \delta_{(\mathbf{x}_i, y_i)}(\mathbf{x}, y)$ to approximate the true distribution, where $\delta_{(\mathbf{x}_i, y_i)}$ is the Dirac delta function. The *empirical risk minimisation* (**ERM**) problem is then

$$\hat{f} = \arg \min_f \mathbb{E}_{\hat{p}(\mathbf{x}, y)}[L(y, \mathbf{x})] = \arg \min_f \frac{1}{n} \sum_{i=1}^n L(y_i, f(\mathbf{x}_i)), \quad (2.7)$$

and if it is assumed $f_{\boldsymbol{\theta}}$ lies in a parametric family parameterised by $\boldsymbol{\theta}$:

$$\hat{\boldsymbol{\theta}} = \arg \min_{\boldsymbol{\theta}} \frac{1}{n} \sum_{i=1}^n L(y_i, f_{\boldsymbol{\theta}}(\mathbf{x}_i)). \quad (2.8)$$

It is worth mentioning that one can show, using the strong law of large numbers, the empirical distribution converges to the true unknown distribution almost surely as the number of observed samples $n \rightarrow \infty$ ([Sen and Singer, 1994] Theorem 2.3.10). This is the reason why performance of ERM generally improves with more data.

2.1.3 Linear regression as a probabilistic generative model

Under ERM, one can show that Equation (2.6) becomes

$$\hat{\boldsymbol{\theta}}_{OLS} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}, \quad (2.9)$$

where $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n]^T$ is an $n \times p$ matrix called the *design matrix* and $\mathbf{y} = [y_1, \dots, y_n]^T$ is the response vector. This is the well-known formula for the ordinary least squares (OLS) estimator in linear regression.

We can also view the problem of linear regression from a probabilistic perspective, by assuming the conditional data distribution $p(y|\mathbf{x})$ is a Gaussian distribution:

$$p(y|\mathbf{x}) = \mathcal{N}(y|\mathbf{x}^T \boldsymbol{\theta}, \sigma^2), \quad (2.10)$$

and data samples are drawn from the distribution independently. Then the likelihood of the data is given by

$$p(y_1, \dots, y_n | \mathbf{x}_1, \dots, \mathbf{x}_n, \boldsymbol{\theta}, \sigma^2) = \prod_{i=1}^n \mathcal{N}(y_i | \mathbf{x}_i^T \boldsymbol{\theta}, \sigma^2). \quad (2.11)$$

The maximum likelihood estimator (MLE) of $\boldsymbol{\theta}$ is then given by

$$\hat{\boldsymbol{\theta}}_{MLE} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}, \quad (2.12)$$

which is the same as the OLS estimator.

2.1.4 Unsupervised learning

Unlike supervised learning, the output variable y is not provided in unsupervised learning and the data of interest is $\mathbf{x} \in \mathcal{X}$, assumed to be drawn from the unknown distribution $p(\mathbf{x})$. The goal of unsupervised learning is to find the underlying structure of the data by analysing properties of the distribution $p(\mathbf{x})$. The main chal-

Challenges of unsupervised learning include typically much higher dimensionality of $\mathbf{x}$, the lack of a clear objective function to optimise and the difficulty in evaluating the performance of the model. One must often rely on domain knowledge and heuristics to guide the interpretation of the results. Based on different model assumptions, unsupervised learning generally aims to find a region in the space of $\mathbf{x}$ such that $p(\mathbf{x})$ is large. Similar to the supervised learning setting, the empirical distribution $\hat{p}(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^n \delta_{\mathbf{x}_i}(\mathbf{x})$ is used to approximate the true distribution.

Clustering Analysis

In clustering analysis, the goal is to partition the data $\{\mathbf{x}_i : i \in [n]\}$ into K groups or *clusters* $C = \{C_k : k \in [K]\}$ that satisfy

$$\bigcup_{k=1}^K C_k = \{\mathbf{x}_i : i \in [n]\}, \quad C_k \cap C_{k'} = \emptyset \quad \forall k \neq k', \quad (2.13)$$

so that data points are similar to those in the same cluster and dissimilar to those in other clusters. One of the challenges is to define a measure of similarity between data points.

Example 2.2 (K-means) A common approach is to use the Euclidean distance for quantitative variables. One of the most popular clustering methods is the K-means algorithm ([Jain, 2010]), which aims to find the partition that minimises the within-cluster sum of squares:

$$W(C) = \sum_{k=1}^K \sum_{\mathbf{x}_i \in C_k, \mathbf{x}_{i'} \in C_k} \|\mathbf{x}_i - \mathbf{x}_{i'}\|^2, \quad (2.14)$$

Dimensionality reduction

This class of methods attempts to identify low-dimensional manifolds within the space of data that represents high data density $p(\mathbf{x})$.

Example 2.3 (Principal Component Analysis - PCA) PCA finds a linear map

that projects the data onto a lower-dimensional space spanned by orthonormal basis vectors called *principal components* ([Jolliffe and Cadima, 2016]). The first principal component is the direction in which the data varies the most, and the second principal component is the direction that is orthogonal to the first principal component and in which the data varies the most, and so on.

Example 2.4 (Independent Component Analysis - ICA) Many multivariate data can be considered as measurements from multiple underlying sources, which cannot be directly measured. Electrophysiology data is a typical example. ICA ([Comon, 1994]) aims to find a linear transformation of the data such that the components are non-Gaussian and statistically independent.

2.1.5 Latent variable models (generative models)

K-means and PCA are unsupervised learning methods that are very established and widely used. However, they are non-probabilistic models that are based on heuristics and intuitive notion of (dis-)similarity and important information in the data. In K-means, the notion of similarity is defined using Euclidean distance, and in PCA, the important information is defined by the directions of maximum variance, which are strong assumptions and not necessarily true in practice. Latent variable models ([Loehlin, 2004]) are a more general class of models that aims to model the probabilistic generating process of the data itself through unobserved latent variables, which serves as representations and summary of the data. For instance, in mixture models, in particular Gaussian Mixture Models (GMM, [Reynolds et al., 2009]), the latent variables represent cluster assignments of the data points while in probabilistic PCA and variational autoencoders, the latent variables represent lower dimensional representations of the data. The latent variable models have several advantages and disadvantages:

- Distribution over the latent variables allows for uncertainty quantification.

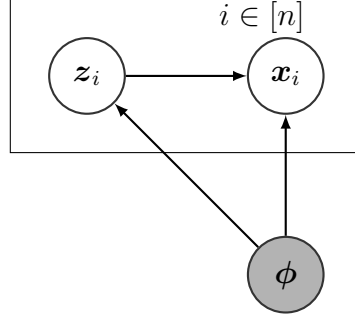


Figure 2.1: **Graphical model of a latent variable model.** Here $\mathbf{x}_i$ is the observed data, $\mathbf{z}_i$ is the latent variable and ϕ is the model parameters.

- The model can be naturally extended to Bayesian learning where the model parameters are assigned prior distributions.
- Prior knowledge and domain expertise can be easily incorporated into the model through the prior distributions.
- Posterior distributions of the latent variables are often intractable and need to be approximated.

Formulation

Consider the scenario that in addition to the observed data $\mathbf{x}_{1:n} = \{\mathbf{x}_i : i \in [n]\}$, there are unobserved latent variables $\mathbf{z}_{1:n} = \{\mathbf{z}_i : i \in [n]\}$. Here we assume, for simplicity, that the number of latent variables is the same as the number of observed data points, which is not necessarily true in a more general setting. The joint distribution of the probabilistic model is given by

$$p_{\phi}(\mathbf{x}_{1:n}, \mathbf{z}_{1:n}) = p_{\phi}(\mathbf{z}_{1:n}) \prod_{i=1}^n p_{\phi}(\mathbf{x}_i | \mathbf{z}_i), \quad (2.15)$$

where ϕ is the model parameters and we assume that the observed data are conditionally independent given the latent variables. The graphical representation of a latent variable model is shown in Figure 2.1.

Example 2.5 (Gaussian Mixture Model - GMM) In GMM, the latent vari-

ables represent the cluster assignments of the data points and are assumed to be drawn independently from a multinomial distribution, i.e.

$$\mathbb{P}(\mathbf{z}_i = k) = \pi_k, \quad (2.16)$$

where for $k \in [K]$, $\pi_k \geq 0$, such that $\sum_{k=1}^K \pi_k = 1$ are called the *mixing coefficients*. Given the latent variable, the observed data follows a Gaussian distribution:

$$\mathbf{x}_i | (\mathbf{z}_i = k) \sim \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k). \quad (2.17)$$

Hence the model parameters are $\boldsymbol{\phi} = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k : k = 1, \dots, K\}$. One can show that assuming $\boldsymbol{\Sigma}_k = \sigma^2 \mathbf{I}$, as $\sigma^2 \rightarrow 0$, the posterior probability $p_{\boldsymbol{\theta}}(\mathbf{z}_i | \mathbf{x})$ becomes binarised and GMM coincides with K-means.

Example 2.6 (Hidden Markov Model - HMM) In the HMM, the latent variables follow a discrete-time (homogeneous) *Markov process*, i.e.

$$p(\mathbf{z}) = p(z_1) \prod_{i=2}^n p(z_i | z_{i-1}). \quad (2.18)$$

Assuming that $\mathbf{z}$ takes values in a finite *state space* $[K]$, the *transition probability matrix* $\mathbf{A}$ is such that $\mathbf{A}_{kk'} = p(z_i = k' | z_{i-1} = k)$. The observed data is assumed to be conditionally independent given the latent variables and follows the *emission distribution* $p(\mathbf{x}_i | z_i)$. Hence the complete likelihood of the data is given by

$$p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}_1 | z_1) p(z_1) \prod_{i=2}^n p(\mathbf{x}_i | z_i) p(z_i | z_{i-1}), \quad (2.19)$$

where the dependence on model parameters $\boldsymbol{\phi}$ is omitted for readability.

Inference challenges

To obtain estimates of the model parameters ϕ , we would like to maximise the model likelihood. Notice that our probabilistic model is $p_\phi(\mathbf{x}, \mathbf{z})$ and we have no access to the latent variables $\mathbf{z}$. Therefore, we wish to maximise the marginal log-likelihood of the observed data

$$\log p_\phi(\mathbf{x}) = \log \int p_\phi(\mathbf{x}, \mathbf{z}) d\mathbf{z} \quad (2.20)$$

with respect to ϕ . However, marginalisation over the latent variables often results in intractable integrals. This poses a significant optimisation challenge and we need to resort to iterative algorithms and approximations.

2.1.6 Expectation-Maximisation (EM) algorithm

The EM algorithm ([Dempster et al., 1977]) is an iterative strategy for maximising the likelihood under missing data or unobserved latent variables. An approximation $q(\mathbf{z})$, called the *variational distribution*, to the posterior distribution $p(\mathbf{z}|\mathbf{x})$ is introduced and we can rewrite the marginal log-likelihood as (assuming $p_\phi(\mathbf{z}) = p(\mathbf{z})$)

$$\begin{aligned} \log \int p_\phi(\mathbf{x}, \mathbf{z}) d\mathbf{z} &= \log \int \frac{p_\phi(\mathbf{x}, \mathbf{z}) q(\mathbf{z})}{q(\mathbf{z})} d\mathbf{z} \\ \text{[Jensen's Inequality]} \quad &\geq \int q(\mathbf{z}) \log \frac{p_\phi(\mathbf{x}|\mathbf{z}) p(\mathbf{z})}{q(\mathbf{z})} d\mathbf{z} \\ &= \int q(\mathbf{z}) \log p_\phi(\mathbf{x}|\mathbf{z}) d\mathbf{z} - \int q(\mathbf{z}) \log \frac{q(\mathbf{z})}{p(\mathbf{z})} d\mathbf{z} \quad (2.21) \\ &= \mathbb{E}_{q(\mathbf{z})}[\log p_\phi(\mathbf{x}|\mathbf{z})] - D_{KL}(q(\mathbf{z})||p(\mathbf{z})) \\ &:= \mathcal{F}(\phi, q). \end{aligned}$$

Here $D_{KL}(\cdot||\cdot)$ is the Kullback-Leibler divergence ([Kullback and Leibler, 1951]) and $\mathcal{F}(\phi, q)$ is called the *evidence lower bound (ELBO)*, where evidence is an alternative term for the marginal log-likelihood. The first term in the ELBO $\mathbb{E}_{q(\mathbf{z})}[\log p_\phi(\mathbf{x}|\mathbf{z})]$ is the *expected conditional log likelihood*, which serves as a measure of reconstruction

error and how well the model fits to the data, whilst the second term $D_{KL}(q(\mathbf{z})||p(\mathbf{z}))$ acts as a model complexity penalisation term and punishes q if it deviates away from the prior distribution $p(\mathbf{z})$.

The EM algorithm, as the name suggests, consists of two steps and at each iteration t :

- **E-step:** Maximise the ELBO with respect to the variational distribution while holding the model parameters fixed. It can be shown that the optimal variational distribution is given by

$$q^{(t)}(\mathbf{z}) = \arg \max_q \mathcal{F}(\phi^{(t-1)}, q) = p_{\phi^{(t-1)}}(\mathbf{z}|\mathbf{x}). \quad (2.22)$$

- **M-step:** Maximise the ELBO with respect to the model parameters while holding the variational distribution fixed:

$$\phi^{(t)} = \arg \max_{\phi} \mathcal{F}(\phi, q^{(t)}) = \arg \max_{\phi} \mathbb{E}_{q^{(t)}(\mathbf{z})} [\log p_{\phi}(\mathbf{x}|\mathbf{z})]. \quad (2.23)$$

Depending on the parametric model $p_{\phi}(\mathbf{x}|\mathbf{z})$, the M-step in some cases cannot be solved exactly, in which case we can use gradient-based optimisation algorithms inside the M-step. This is called the *generalised EM algorithm* ([McLachlan and Krishnan, 2007]). Furthermore, during the E-step, the true posterior distribution $p_{\phi^{(t-1)}}(\mathbf{z}|\mathbf{x})$ is only available in a few special cases (e.g. GMM and probabilistic PCA) and in general, we need to restrict the class of variational distributions to a more tractable family of distributions. This approach is called *variational inference*.

Mean-field variational family

One of the most popular choices of variational family is the *mean-field variational family*, where we assume that the latent variables are independent of each other:

$$q(\mathbf{z}) = \prod_{i=1}^n q(\mathbf{z}_i). \quad (2.24)$$

Though this assumption greatly simplifies the computation of the E-step, it is a strong assumption and often result in underestimation of posterior variance in practice ([Blei et al., 2017]). Other variational families are possible, including the use of normalising flows that finds a transformation of a simple distribution to achieve richer variational families.

2.2 Deep learning

Deep learning is a sub-field of machine learning that utilises Artificial Neural Networks (ANNs), which are inspired by biological neural networks, to learn mappings from input data to output data. The word “deep” refers to the use of multiple layers in the neural network for parameterising a rich family of transformations. The use of deep learning has been very successful in a wide range of applications, including computer vision, natural language processing (NLP), signal recognition and generation, etc. The success of deep learning is due to the development of new architectures, optimisation algorithms, but more importantly, the availability of large-scale datasets and extremely powerful computational hardware. It has several advantages and disadvantages compared to traditional machine learning approaches, such as:

Advantages:

- There is no need for feature engineering. The neural network learns to extract high level and abstract features from the data automatically.
- Stochastic gradient descent (SGD, see Section 2.2.2) and its variants allow deep

learning algorithms to handle large-scale datasets efficiently, i.e. the methods are *scalable*.

- Due to the universal approximation theorem ([Hornik et al., 1989]), deep learning models are theoretically capable of approximating arbitrary functions.
- Advancement in computational hardware (e.g. GPU and TPU) enables fast and scalable training of these models.
- Software libraries such as TensorFlow and PyTorch provide ease of use and flexibility for building and training deep learning models.
- Deep learning models are fundamentally modularised. Complex models can be built by stacking simple and basic modules, which makes it easy to modify and extend models.

Disadvantages:

- The training of deep learning models is computationally expensive due to the large amount of parameters.
- Deep learning models are *data-hungry* and could be prone to overfitting when the amount of training data is limited.
- These models are often considered as *black boxes* and it is typically extremely difficult to interpret the decisions and predictions made by the models.

2.2.1 Examples of modules

As mentioned, deep learning models are fundamentally modularised and in this section, we give details of the basic modules that are used in this thesis.

Fully-connected layers

A *fully-connected layer* $f \sim \text{Dense}(q, p)$ is also called a *dense layer*, a *feed-forward layer* or a *linear layer*:

$$f(\mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{b}, \quad (2.25)$$

where $\mathbf{x} \in \mathbb{R}^p$ is the input, $\mathbf{W} \in \mathbb{R}^{q \times p}$ and $\mathbf{b} \in \mathbb{R}^q$ are parameters called *weights* and *biases* respectively. Here p is the input dimension and q is the output dimension, also called the *number of units*.

Nonlinearity

Element-wise nonlinear functions σ , called *activations*, are used to introduce non-linearity in the neural network, which is essential in ensuring expressiveness of the neural networks. Popular activation functions include but are not limited to:

- **Sigmoid.** $\sigma(x) = \frac{1}{(1+\exp(-x))}$.
- **Hyperbolic tangent.** $\sigma(x) = \tanh(x) = \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}$.
- **ReLU** (Rectified Linear Unit, [Nair and Hinton, 2010]). $\sigma(x) = \max(0, x)$.
- **Softplus.** $\sigma(x) = \log(1 + \exp(x))$.

Multi-layer perceptron (MLP)

An MLP $f \sim \text{MLP}(d_L, \dots, d_0, \sigma)$ ([Gardner and Dorling, 1998]) is formed by composing multiple dense layers with activations in-between:

$$f(\mathbf{x}) = f_L \circ \sigma \circ \dots \circ \sigma \circ f_1(\mathbf{x}), \quad f_l \sim \text{Dense}(d_l, d_{l-1}), \quad \text{for } l \in [L], \quad (2.26)$$

where L is the number of layers. See Figure 2.2 for a graphical representation of an MLP.

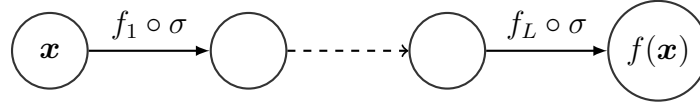


Figure 2.2: **Graphical representation of a multi-layer perceptron.** x is the input and $f(x)$ is the output. The dotted arrow represents multiple applications of dense layers and activations.

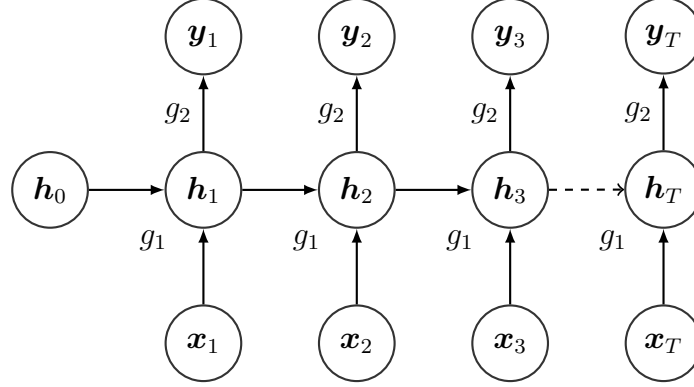


Figure 2.3: **Graphical representation of a Recurrent Neural Network.** Here $x_1, \dots, x_T$ are the inputs, $h_0, \dots, h_T$ are the hidden states, and $y_1, \dots, y_T$ are the outputs. g_1 and g_2 are neural networks themselves and the dotted arrow represents multiple applications of the same procedure.

Recurrent Neural Network (RNN)

A Recurrent Neural Network ([Rumelhart et al., 1986]) is typically applied to time series data $x_1, x_2, \dots, x_T$ and $y_1, y_2, \dots, y_T$ are the outputs. In an RNN, the output y_t is a function of the whole history $x_{1:t}$. This is achieved by computing a sequence of hidden representations h_t based on the history $x_{1:t}$ and decode h_t to get y_t . Hence starting from an initial state h_0 , for each $t = 1, \dots, T$

$$\begin{aligned} h_t &= g_1(h_{t-1}, x_t) \\ y_t &= g_2(h_t), \end{aligned} \tag{2.27}$$

where g_1, g_2 are neural networks and can be as simple as a dense layer followed by an activation. See Figure 2.3 for a graphical representation of an RNN.

2.2.2 Optimisation

Training the large amount of parameters in deep learning models is extremely challenging. Recall the ERM problem in Equation (2.7):

$$\min_{\phi} \mathcal{L}(\phi) = \min_{\phi} \frac{1}{n} \sum_{i=1}^n L(y_i, f_{\phi}(\mathbf{x}_i)). \quad (2.28)$$

Unlike linear regression, in most cases of deep learning, an analytic solution for the parameter ϕ is infeasible given the complexity of the neural network f_{ϕ} . Instead, numerical optimisation is used and the most common one used for training deep neural networks is *gradient descent*, i.e. starting with an initial value ϕ_0 , for each iteration t ,

$$\phi_t = \phi_{t-1} - \epsilon_t \frac{d\mathcal{L}}{d\phi}(\phi_{t-1}), \quad (2.29)$$

where ϵ_t is called the *learning rate*. Automatic differentiation tools ([Paszke et al., 2017]) allow efficient computation of the gradient $\frac{d\mathcal{L}}{d\phi}$ through chain rule and matrix multiplication but there are several challenges:

- Complexity of computing the gradient scales linearly with the number of samples n :

$$\frac{d\mathcal{L}}{d\phi} = \frac{1}{n} \sum_{i=1}^n \frac{d}{d\phi} L(y_i, f_{\phi}(\mathbf{x}_i)). \quad (2.30)$$

Many datasets for training deep learning models are huge in size hence computing the gradient is not practical even for modern computational hardware. Instead an unbiased stochastic estimate of the gradient:

$$\widehat{\frac{d\mathcal{L}}{d\phi}} = \frac{1}{|B|} \sum_{i \in B} \frac{d}{d\phi} L(y_i, f_{\phi}(\mathbf{x}_i)), \quad (2.31)$$

where B is a uniformly randomly sampled subset of $[n]$, called a *minibatch*, with $|B| \ll n$. This approach is called *stochastic gradient descent* (SGD).

Typically, the dataset is split into $n/|B|$ minibatches and each minibatch is

used in turn for updating the model parameters. After the whole dataset has been processed once, called an *epoch*, a different randomised set of minibatches is used for the next epoch. Although SGD uses a noisy estimate of the true gradient, it has been shown to converge almost surely for convex objective functions ([Bottou, 1998]). For non-convex optimisation problems, which is typical for highly complex and nonlinear f_ϕ , extensions of SGD, such as ADAM ([Kingma and Ba, 2014]), have been proposed to help cross over the ridges and valleys of the loss function.

- The computation of gradients involves a chain of matrix multiplications of the Jacobian matrices. If the matrices have eigenvalues that are greater than 1, then in some directions, the gradients as products can become very large. This is called the *exploding gradients* issue ([Ribeiro et al., 2020]). On the other hand, if some eigenvalues are small and close to 0, the gradients can *vanish*. This is problematic especially in RNNs where the same set of weights are applied multiple times. The LSTM cell ([Hochreiter and Schmidhuber, 1997]) is designed to mitigate this issue. Another solution is to clip gradients such that gradients larger than a threshold are set to the threshold.

2.2.3 Variational Autoencoder (VAE)

As mentioned in Section 2.1.5, the exact posterior distribution $p(\mathbf{z}|\mathbf{x})$ is intractable in most cases in latent variable models, especially when the likelihood $p_\phi(\mathbf{x}|\mathbf{z})$ and the prior distribution $p(\mathbf{z})$ are non-trivial. In the field of deep learning, the likelihood is also called a *decoder* and one of the solutions to the problem of approximating the posterior distribution is *amortised learning*. This involves considering a variational distribution $q_\psi(\mathbf{z}|\mathbf{x})$, which is typically parameterised by a neural network, called a *encoder*, with parameters ψ . The encoder serves as a function that maps data $\mathbf{x}$ to the variational distribution $q_\psi(\mathbf{z}|\mathbf{x})$, which is an approximation to the true posterior distribution $p(\mathbf{z}|\mathbf{x})$. This approach has two main advantages:

- When the number of latent variables is large and scales with the sample size, this avoids learning the variational parameter for each latent variable. Learning a map from data to the variational parameters allows training to be scalable with large datasets.
- When new data $\mathbf{x}^*$ comes in, one can directly get an approximation to the posterior distribution $p(\mathbf{z}|\mathbf{x}^*)$ by passing the new data through the encoder instead of training the whole model from the beginning.

This framework is called a *variational autoencoder* ([Kingma and Welling, 2013]) and the ELBO ((2.21)) becomes:

$$\mathcal{F}(\mathbf{x}, \phi, \psi) = \mathbb{E}_{q_\psi(\mathbf{z}|\mathbf{x})}[\log p_\phi(\mathbf{x}|\mathbf{z})] - D_{KL}(q_\psi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})). \quad (2.32)$$

The (scaled) ELBO for the whole set of observations $\{\mathbf{x}_i : i \in [n]\}$ is

$$\mathcal{F}(\phi, \psi) = \frac{1}{n} \sum_{i=1}^n \{ \mathbb{E}_{q_\psi(\mathbf{z}|\mathbf{x}_i)}[\log p_\phi(\mathbf{x}_i|\mathbf{z})] + D_{KL}(q_\psi(\mathbf{z}|\mathbf{x}_i)||p(\mathbf{z})) \}. \quad (2.33)$$

Assuming that $\{(\mathbf{x}_i, \mathbf{z}_i) : i \in [n]\}$ are drawn i.i.d. from $p_\phi(\mathbf{x}, \mathbf{z})$, $\mathcal{F}(\phi, \psi)$ is a lower bound of the scaled model evidence $\frac{1}{n} \log p_\phi(\mathbf{x}_1, \dots, \mathbf{x}_n) = \frac{1}{n} \sum_{i=1}^n \log p_\phi(\mathbf{x}_i)$, since $\mathcal{F}(\mathbf{x}_i, \phi, \psi) \leq \log p_\phi(\mathbf{x}_i)$ for all i . In practice, instead of maximising the ELBO, we minimise the negative ELBO as our loss function (also known as *variational free energy*) $\mathcal{L}(\phi, \psi) = -\mathcal{F}(\phi, \psi)$. Now as opposed to the two-step optimisation in EM algorithm, we use gradient descent to optimise the loss function with respect to ϕ and ψ jointly.

A common choice of the encoder is a Gaussian distribution with diagonal matrix and a standard Gaussian distribution for the prior:

$$\begin{aligned} q_\psi(\mathbf{z}|\mathbf{x}) &= \mathcal{N}(\mathbf{z}|\mu_\psi(\mathbf{x}), \sigma_\psi^2(\mathbf{x})\mathbf{I}), \\ p(\mathbf{z}) &= \mathcal{N}(0, \mathbf{I}), \end{aligned} \quad (2.34)$$

where μ_ψ, σ_ψ are neural networks. With this particular choice, the KL divergence has an analytic solution and it is easy to compute its gradient with respect to the weights of the encoder. However, the same cannot be said about the expected conditional log-likelihood, the analytic form of which we do not often have access to. The solution to this is to use a Monte-Carlo estimate of the gradient.

2.2.4 Reparameterisation trick

In order to optimise the loss function, it remains to estimate the gradients of the term $\mathbb{E}_{q_\psi(\mathbf{z}|\mathbf{x})}[\log p_\phi(\mathbf{x}|\mathbf{z})]$. We could get a single sample Monte Carlo estimate of this quantity by drawing $\mathbf{z}^s \sim q_\psi(\mathbf{z}|\mathbf{x})$ and estimating

$$\hat{\mathbb{E}}_{q_\psi(\mathbf{z}|\mathbf{x})}[\log p_\phi(\mathbf{x}|\mathbf{z})] = \log p_\phi(\mathbf{x}|\mathbf{z}^s). \quad (2.35)$$

But by doing this we lose the dependence on ψ and we cannot compute the gradient with respect to it. The solution to this is the reparameterisation trick ([Kingma and Welling, 2013]).

We know for a fact that if

$$\begin{aligned} \mathbf{z}^s &\sim \mathcal{N}(\mu, \sigma^2), \\ (\mathbf{z}')^s &= \mu + \sigma\epsilon, \quad \epsilon \sim \mathcal{N}(0, \mathbf{I}) \end{aligned}, \quad (2.36)$$

Then $\mathbf{z}^s$ and $(\mathbf{z}')^s$ have the same distribution. Hence by the law of the unconscious statistician ([DeGroot and Schervish, 2012]), if we use a Gaussian variational distribution, the expected conditional log-likelihood can be rewritten as

$$\mathbb{E}_{q_\psi(\mathbf{z}|\mathbf{x})}[\log p_\phi(\mathbf{x}|\mathbf{z})] = \mathbb{E}_\epsilon[\log p_\phi(\mathbf{x}|\mu_\psi(\mathbf{x}) + \sigma_\psi^2(\mathbf{x})\epsilon)], \quad (2.37)$$

and use the estimate $\log p_\phi(\mathbf{x}|\mu_\psi(\mathbf{x}) + \sigma_\psi^2(\mathbf{x})\epsilon^s)$ with $\epsilon^s \sim \mathcal{N}(0, \mathbf{I})$. Notice now we are able to compute the gradient of this estimate with respect to all the parameters and this estimator is an unbiased estimator of the true gradient of the expected

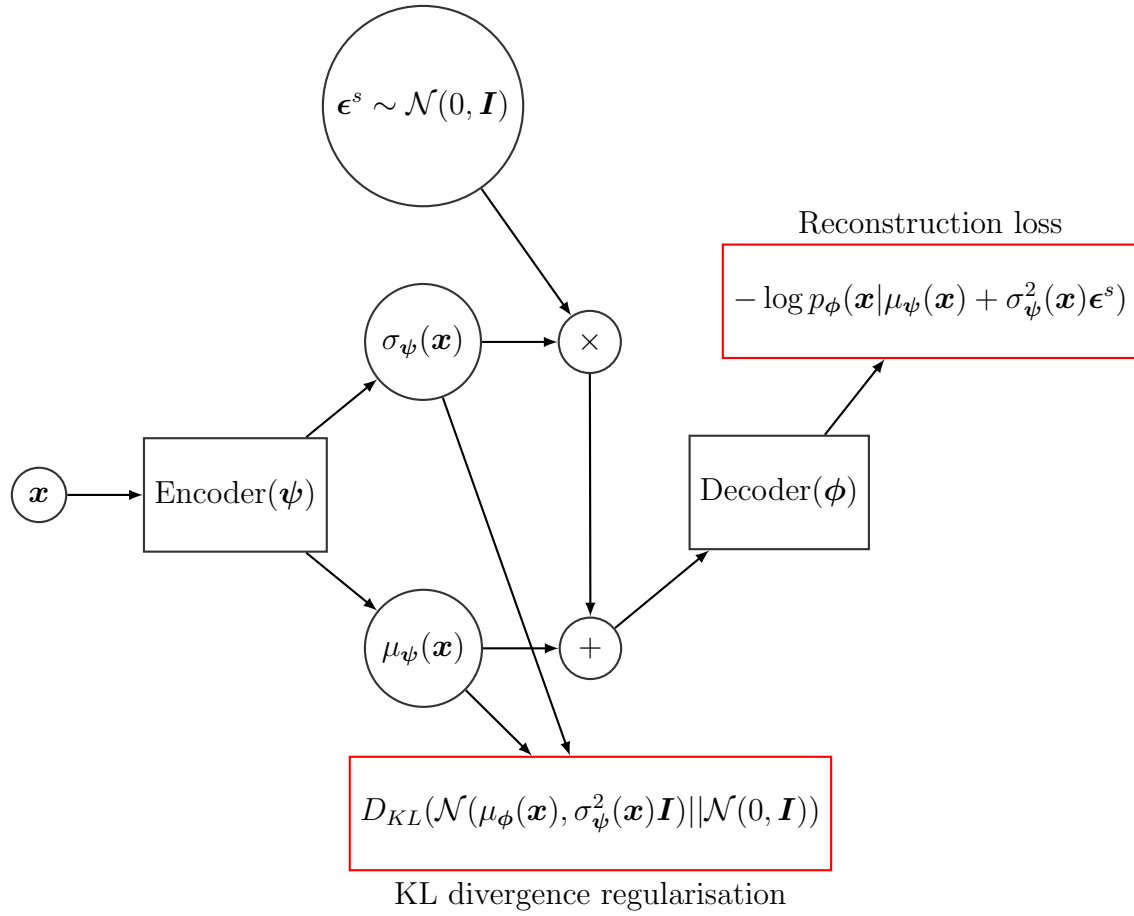


Figure 2.4: **Illustration a variational autoencoder with Gaussian variational distribution.** The encoder with parameters ψ takes input $\mathbf{x}$, and gives the variational mean $\mu_\psi(\mathbf{x})$ and standard deviation $\sigma_\psi(\mathbf{x})$. Using the reparameterisation trick, a sample ϵ^s from the standard Gaussian distribution is drawn, which is then multiplied with σ_ψ and added to $\mu_\psi(\mathbf{x})$. The resulting sample from the variational distribution is passed through the decoder with parameters ϕ to get an estimate of the original input. Here the red boxes indicate different terms of the loss function.

conditional log-likelihood. See Figure 2.4 for an illustration of this special case of VAE.

2.3 Statistical hypothesis testing

In many cases we want to make claims about some data $\mathbf{x} \in \mathcal{X}$ which are assumed to be drawn from a (possibly unknown) distribution $p(\mathbf{x})$. Statistical hypothesis tests are methods that are used to decide whether there is sufficient evidence in the data that supports some sort of “abnormality” in a rigorous manner. The general

procedure is as follows. Firstly, a claim that the effect being studied does not exist is defined. This is called the *null hypothesis*. Then we are interested in the question “Under the null hypothesis (assuming there is no effect), what is the probability that the data can be as extreme as my observed data?” If the probability is below a threshold, called the *significance level*, then we can conclude that there is evidence of an effect and the null hypothesis is rejected by the hypothesis test with a certain significance level ([Lehmann et al., 1986a]).

Formally, we would want to summarise the data with a function $S : \mathcal{X} \rightarrow \mathbb{R}$, called a *statistic* that maps the data into a scalar, so that there is an order imposed on the data and the “as extreme as” statement is well-defined. The probability of question is called the Type I error (also more widely known as the *p-value*) and is defined as

$$p_{>} = \mathbb{P}(S(\mathbf{x}) > S(\mathbf{x}_{obs}) | H_0), \quad (2.38)$$

where $\mathbf{x}_{obs}$ is the observed data and the probability is taken with respect to the conditional data distribution $p(\cdot | H_0)$ given the null hypothesis.

With different study objectives, different *alternative hypotheses* can be defined, upon which the following definitions of p-value can also be used:

$$\begin{aligned} p_{<} &= \mathbb{P}(S(\mathbf{x}) < S(\mathbf{x}_{obs}) | H_0) \\ p_{\neq} &= \mathbb{P}(|S(\mathbf{x})| > |S(\mathbf{x}_{obs})| | H_0), \end{aligned} \quad (2.39)$$

where $p_{>}, p_{<}$ are called p-values of *one-sided* tests whilst $p_{\neq}$ is called a p-value of a *two-sided* test.

In many cases, the Central Limit Theorem (CLT) tells us as the number of sample tends to infinity, the distribution of the mean of samples converges in distribution to a normal distribution, which allows one to approximate the p-value. However this is not always the case as we may be interested in test statistics other than the sample mean or there is simply a lack of data. Non-parametric tests solves this

by not placing any assumption on the null distribution and sometimes exploiting structure (exchangeability) within the data.

2.3.1 Permutation tests

An example of non-parametric tests used in this thesis is the *permutation test* ([Lehmann et al., 1986b, Pesarin and Salmaso, 2010]), which uses the idea of proof by contradiction. Suppose we have observations $\mathbf{x}, \mathbf{y}$ from two groups and we want to test if they come from the same distribution, which is the null hypothesis. Under the assumption that the group labels are exchangeable, permuting the group assignment of the observations gives the same joint distribution under the null hypothesis. Hence the test involves calculating the test statistic for every permutation of group assignment. The null hypothesis is rejected if the observed test statistic is among the extremes of the test statistics obtained by permutations.

The permutation test is an exact test and does not put any parametric assumption on the underlying data distribution. However, this comes with the downside of extremely high computational cost, due to the fact that the number of permutations grows exponentially with the number of samples. This is solved by a modification of the test with Monte Carlo sampling.

2.3.2 Monte Carlo testing

The important quantity p-value $p_{>}$ can be rewritten as

$$\begin{aligned} p_{>} &= \mathbb{P}(S(\mathbf{x}) > S(\mathbf{x}_{obs}) | H_0) \\ &= \mathbb{E}[\mathbb{1}\{S(\mathbf{x}) > S(\mathbf{x}_{obs})\} | H_0]. \end{aligned} \tag{2.40}$$

Hence we can consider using the Monte Carlo estimate

$$\hat{p}_{>} = \frac{1}{N_s} \sum_{s=1}^{N_s} \mathbb{1}\{S(\mathbf{x}^s) > S(\mathbf{x}_{obs})\}, \tag{2.41}$$

where $\mathbf{x}^s$ are samples from the null distribution and N_s is the number of samples. This idea can be applied to permutation tests as a random assignment of groups is a sampling from the null distribution and thus reducing the computational cost of permutation test by using a Monte Carlo estimate of the p-value ([Pesarin and Salmaso, 2010, Walter and Barkema, 2015]).

2.3.3 The multiple testing problem

In some cases it is useful to test multiple hypotheses simultaneously. However, this raises the problem that as the number of tests increases, there is an increasing chance that one of the tests will show significant results even the ground truth is all of the null hypotheses are true ([Hochberg, 1987]).

Formally, suppose we have a family of null hypotheses and $\{\mathbf{x}_i : i \in [M]\}$ are data from the null distributions, where M is the number of tests. Then the family-wise error rate (FWER), defined as the probability that there is at least one false positives in the family of tests, is

$$\mathbf{FWER} = \mathbb{P} \left(\bigcup_{i=1}^M \{S(\mathbf{x}_i) > S(\mathbf{x}_{obs,i})\} \right) \geq \mathbb{P}(S(\mathbf{x}_i) > S(\mathbf{x}_{obs,i})) \forall i \in [M], \quad (2.42)$$

where $\mathbf{x}_{obs,i}$ is the observed data for test i . This can be thought of as the Type I error of the test with null hypothesis that there is no effect anywhere (all the individual null hypotheses are true). If the tests are independent, then it is trivial to show

$$\mathbf{FWER} = 1 - \prod_{i=1}^M (1 - \mathbb{P}(S(\mathbf{x}_i) > S(\mathbf{x}_{obs,i}))), \quad (2.43)$$

which is typically much larger than the individual Type I errors and is non-decreasing with M . A number of methods have been proposed to correct the individual significance levels to assure the FWER is controlled, e.g. the Bonferroni correction, but these methods can result in very conservative tests, which has low statistical power.

2.3.4 Maximum statistic

The use of maximum statistic ([Nichols and Hayasaka, 2003]) is one of the ways to mitigate the issue of multiple testing. Suppose all statistics $S(\mathbf{x}_i)$ have identical marginal distribution under the null hypotheses, then

$$\mathbb{P} \left(\bigcup_{i=1}^M \{S(\mathbf{x}_i) > T\} \right), \quad (2.44)$$

which is the probability that at least one of $S(\mathbf{x}_i)$ is greater than T for some $T \in \mathbb{R}$, can be re-expressed as

$$\begin{aligned} \mathbb{P} \left(\bigcup_{i=1}^M \{S(\mathbf{x}_i) > T\} \right) &= 1 - \mathbb{P} \left(\bigcap_{i=1}^M \{S(\mathbf{x}_i) \leq T\} \right) \\ &= 1 - \mathbb{P} \left(\max_{i \in [M]} S(\mathbf{x}_i) \leq T \right) \\ &= \mathbb{P} \left(\max_{i \in [M]} S(\mathbf{x}_i) > T \right). \end{aligned} \quad (2.45)$$

Hence if we choose T to be the $(1 - \alpha)$ quantile of the distribution of the maximum statistic $\max_{i \in [M]} S(\mathbf{x}_i)$, where α is the significance level, and reject the i -th null hypothesis if and only if $S(\mathbf{x}_{obs,i}) > T$, the FWER is controlled at significance level of α .

The distribution of the maximum statistic is complicated and most likely there is no closed-form solution for its quantiles. Hence Monte Carlo tests (together with permutation tests) are often used to approximate the quantile.

Chapter 3

Existing models for estimating dynamic brain networks

In Section 1.3 we discussed the sliding window approach for computing dynamic functional connectivity and the limitations of this approach. In this chapter, we give an overview of two recently developed techniques that use generative models to infer brain network dynamics in a principled, model-based and data-driven way. In addition, we describe attempts to quantitatively compare different generative models and contributions to the *osl-dynamics* toolbox ([Gohil et al., 2024a]).

3.1 Modelling brain dynamics with the Hidden Markov Model

[Baker et al., 2014] proposed using the HMM (Hidden Markov Model) to model the dynamics of functional brain networks. The basic assumption is that temporally, the brain undergoes transitions between brain states which have different spatiotemporal characteristics. The generative model of HMM describes how the time series data is generated given the model parameters and through training, we find parameters that best generate the observed data and partitions the time dimension of the data

into a finite number of mutually exclusive states.

3.1.1 Formulation of generative model

The general HMM framework is defined in Example 2.6 and in what follows we delve into how to use the HMM to describe the generating process of functional neuroimaging data. Suppose the data we have is $\mathbf{x}_{1:T} = \{\mathbf{x}_t : t \in [T]\}$, where T is the number of samples and $\mathbf{x}_t$ is the functional data we observe at time t , which is a vector of length N_c - the number of channels. The hidden states $s_{1:T} = \{s_t : t \in [T]\}$ of the HMM represent brain state labels taking values in the finite set $[J]$, with J being the number of states. The emission distributions are assumed to be multivariate Gaussian distributions:

$$p(\mathbf{x}_t | s_t) = \mathcal{N}(\mathbf{x}_t | \boldsymbol{\mu}_{s_t}, \boldsymbol{\Sigma}_{s_t}). \quad (3.1)$$

The collection $\boldsymbol{\phi} = \{\mathbf{A}, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j : j \in [J]\}$, where $\mathbf{A}$ is the transition probability matrix, includes all parameters to infer during model training.

3.1.2 Model training

In *osl-dynamics* ([Gohil et al., 2024a]), model training is done with the EM algorithm (see Section 2.1.6). More specifically, we employ a modified Baum-Welch algorithm ([Baum and Petrie, 1966, Baum and Eagon, 1967]). At each iteration i , we perform one E-step and one M-step. During the E-step, the forward-backward algorithm is used to compute the posterior probability (state probability $q(s_{1:T})$) of the hidden states.

During the M-step, the parameters $\boldsymbol{\phi}$ are updated. The transition probability matrix $\mathbf{A}$ is updated by the stochastic update technique used in [Vidaurre et al., 2018a]:

$$\mathbf{A}_{new} = (1 - \zeta)\mathbf{A}_{old} + \zeta\mathbf{A}_{interim}, \quad (3.2)$$

where $\mathbf{A}_{old}$ and $\mathbf{A}_{new}$ are the transition probability matrices before and after the update, $\mathbf{A}_{interim}$ is the interim update of the transition probability matrix given by the Baum-Welch algorithm, and ζ is the update weight that decreases with training epoch:

$$\zeta(i) = (i + delay)^{-forget}, \quad (3.3)$$

where $delay = 5$ and $forget = 0.7$. The rest of the parameters $\{\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j : j \in [J]\}$ are updated by minimising the negative expected conditional log-likelihood of the data through gradient descent:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\phi}) &= -\mathbb{E}_{q(s_{1:T})} [\log p_{\boldsymbol{\phi}}(\mathbf{x}_{1:T} | s_{1:T})] \\ &= -\sum_{t=1}^T \mathbb{E}_{q(s_t)} [\log p_{\boldsymbol{\phi}}(\mathbf{x}_t | s_t)] \\ &= -\sum_{t=1}^T \sum_{j=1}^J \gamma_{jt} \log \mathcal{N}(\mathbf{x}_t | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j), \end{aligned} \quad (3.4)$$

where $\gamma_{jt} = q(s_t = j)$ is the posterior probability that the state at time t is j .

3.2 Extending HMM with Dynamic Network Modes

Dynamic Network Modes (DyNeMo, [Gohil et al., 2022]) attempts to tackle two of the limitations of the HMM. Firstly, the temporal model of the HMM is a first-order Markov chain, which means that the state at time t only depends on the state at time $t - 1$. This is a strong prior assumption on the data that hinders the model's ability to capture long range temporal dependency. DyNeMo replaces the Markov chain with an RNN (See Section 2.2.1) to model the temporal dependency. Secondly, the time series data is partitioned into a finite number of mutually exclusive states in the HMM, i.e. data at each time point can only be generated by one state. This is a strong assumption on the data and might not be true in practice. DyNeMo introduces a soft-mixture of *modes* of networks, which is a much more flexible model

for modelling the potentially continuous nature of the dynamics of the brain. It has been shown that DyNeMo could infer TVFC more accurately than the HMM ([Gohil et al., 2022]), and can identify networks that are useful for studying task data ([Gohil et al., 2024b]).

3.2.1 Formulation of generative model

Conceptually, DyNeMo learns a set of modes, where each mode corresponds to a distinct spatial/spectral configuration of network activity. The actual network activity at each time point is then modelled as a linear combination of these modes. The time series of mixing coefficients characterises the dynamics of the modes and is referred to as the *mode time courses*. Formally, similar to the HMM, DyNeMo assumes that the data is generated by

$$\mathbf{x}_t \sim \mathcal{N}(\mathbf{m}_t, \mathbf{C}_t) \quad \forall t \in [T], \quad (3.5)$$

where $\mathbf{m}_t, \mathbf{C}_t$ are the instantaneous time-varying mean and covariance of the data at time t , which are generated by a mixture of finite number of modes of means $\boldsymbol{\mu}_j$ and covariances $\boldsymbol{\Sigma}_j$:

$$\begin{aligned} \mathbf{m}_t &= \sum_{j=1}^J \alpha_{jt} \boldsymbol{\mu}_j, \\ \mathbf{C}_t &= \sum_{j=1}^J \alpha_{jt} \boldsymbol{\Sigma}_j, \end{aligned} \quad (3.6)$$

where α_{jt} is the mixing coefficient for mode j at time t . The mixing coefficients are generated through a softmax transformation of latent probabilistic variables $\boldsymbol{\theta}_{1:T}$ (where $\boldsymbol{\theta}_t = \{\theta_{jt} : j \in [J]\} \forall t \in [T]$), referred to as the *logits*:

$$\alpha_{jt} = \frac{\exp(\theta_{jt}/\tau)}{\sum_{j'=1}^J \exp(\theta_{j't}/\tau)}, \quad (3.7)$$

in order to ensure $\sum_{j=1}^J \alpha_{jt} = 1, \forall t \in [T]$. Here τ is the temperature parameter that controls the sharpness of the mixing coefficients. The prior distribution (which defines the probabilistic generating process) of the logits, which are the latent variables in this generative model, is a product of the conditional distributions:

$$p(\boldsymbol{\theta}_{1:T}) = \prod_{t=2}^T p(\boldsymbol{\theta}_t | \boldsymbol{\theta}_{1:t-1}) = \prod_{t=2}^T \mathcal{N}(\boldsymbol{\theta}_t | \boldsymbol{\mu}_{\boldsymbol{\theta}_t}(\boldsymbol{\theta}_{1:t-1}), \boldsymbol{\sigma}_{\boldsymbol{\theta}_t}^2(\boldsymbol{\theta}_{1:t-1}) \mathbf{I}), \quad (3.8)$$

where the parameters $\boldsymbol{\mu}_{\boldsymbol{\theta}_t}(\boldsymbol{\theta}_{1:t-1})$ and $\boldsymbol{\sigma}_{\boldsymbol{\theta}_t}(\boldsymbol{\theta}_{1:t-1})$ are parameterised by an RNN:

$$\begin{aligned} \boldsymbol{\mu}_{\boldsymbol{\theta}_t}(\boldsymbol{\theta}_{1:t-1}) &= g_{gen}(\text{RNN}(\boldsymbol{\theta}_{1:t-1})), \\ \boldsymbol{\sigma}_{\boldsymbol{\theta}_t}(\boldsymbol{\theta}_{1:t-1}) &= \rho(h_{gen}(\text{RNN}(\boldsymbol{\theta}_{1:t-1}))), \end{aligned} \quad (3.9)$$

where g_{gen}, h_{gen} are learnable affine transformations and ρ is the softplus function to ensure positivity of standard deviations.

Remark 3.1 Through out this thesis, RNNs, and in particular the vanilla LSTMs, are used over other sequence models such as the Transformers ([Vaswani et al., 2017]) and the S4 model ([Gu et al., 2021]). The main reason was that empirically we found LSTMs work reasonably well for our purpose (modelling FC dynamics) and preliminary experiments with Transformers did not seem to have a huge impact on the results.

3.2.2 Model training

During model training, we aim to infer model parameters $\boldsymbol{\phi}$ that include $\{\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j : j \in [J]\}$ as well as parameters of the RNN, and affine transformations g_{gen}, h_{gen} . Due to the highly intractable nature of the prior distribution parameterised by an RNN, it is impossible to compute the exact posterior distribution $p(\boldsymbol{\theta} | \mathbf{x})$ analytically. In *osl-dynamics*, we use variational inference to infer an approximation to the posterior distribution as well as the observation model parameters. The loss function to be

minimised is the variational free energy (see Section 2.2.3 for derivation of the general framework):

$$\mathcal{L} = -\mathbb{E}_{q(\boldsymbol{\theta}_{1:T})}[\log p_{\boldsymbol{\phi}}(\mathbf{x}_{1:T}|\boldsymbol{\theta}_{1:T})] + D_{KL}(q(\boldsymbol{\theta}_{1:T})||p_{\boldsymbol{\phi}}(\boldsymbol{\theta}_{1:T})), \quad (3.10)$$

where $q(\boldsymbol{\theta}_{1:T})$ is the variational distribution, on which we employ the mean-field approximation (Section 2.1.6):

$$q(\boldsymbol{\theta}_{1:T}) = \prod_{t=1}^T q(\boldsymbol{\theta}_t) = \prod_{t=1}^T \mathcal{N}(\boldsymbol{\theta}_t | \mathbf{m}_{\boldsymbol{\theta}_t}, \mathbf{s}_{\boldsymbol{\theta}_t}^2 \mathbf{I}_J). \quad (3.11)$$

As mentioned in Section 2.2.3, when the number of samples T is large, it is not feasible to learn all the variational parameters $\mathbf{m}_{\boldsymbol{\theta}_t}, \mathbf{s}_{\boldsymbol{\theta}_t}$ separately. Instead we use amortised inference, where a function that maps the observed data to the variational parameters is learned. This has the additional benefit of straightforwardly applying a trained model to unseen data. More specifically, we use a bidirectional RNN:

$$\begin{aligned} \mathbf{m}_{\boldsymbol{\theta}_{1:T}} &= g_{inf}(\text{BiRNN}(\mathbf{x}_{1:T})), \\ \mathbf{s}_{\boldsymbol{\theta}_{1:T}} &= \rho(h_{inf}(\text{BiRNN}(\mathbf{x}_{1:T}))), \end{aligned} \quad (3.12)$$

where again g_{inf}, h_{inf} are learnable affine transformations. Hence the parameters of the encoder $\boldsymbol{\psi}$ include parameters of the bidirectional RNN, and affine transformations g_{inf}, h_{inf} . The loss function in (3.10) becomes

$$\mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\psi}) = - \underbrace{\sum_{t=1}^T \mathbb{E}_{q_{\boldsymbol{\psi}}(\boldsymbol{\theta}_t)}[\log p_{\boldsymbol{\phi}}(\mathbf{x}_t|\boldsymbol{\theta}_t)]}_{\mathbf{LL}} + \underbrace{\sum_{t=2}^T D_{KL}(q_{\boldsymbol{\psi}}(\boldsymbol{\theta}_t)||p_{\boldsymbol{\phi}}(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{1:t-1}))}_{\mathbf{KL}}. \quad (3.13)$$

The first term **LL** measures how well the model is fitted to the observed data and the second term **KL** is a temporal regularisation term that encourages the model to learn a sensible temporal structure. During training, gradient descent type algorithm was used to minimise the loss function with respect to $\boldsymbol{\phi}, \boldsymbol{\psi}$ and the reparameterisation

trick was used to estimate the gradient of $\mathbf{LL}$ (Section 2.2.4).

Breaking data into sequences

In practice, we are able to feed the whole sequence directly to a very long RNN sequence due to memory restrictions. More importantly, although the use of long short-term memory cell (LSTM, [Hochreiter and Schmidhuber, 1997]) mitigates the issue of gradient explosion and vanishing to some extent, it is still an issue when the sequence is long ([Bengio et al., 1994]). Typically, it is advised that the sequence length cannot exceed a few hundreds ([Jozefowicz et al., 2015]). Hence we further separate the data into sequences to feed the RNNs. Let T_{seq} be the sequence length and N_{seq} be the number of sequences such that $\lfloor T/T_{seq} \rfloor = N_{seq}$, then the loss for sequence n is

$$\mathcal{L}_n(\phi, \psi) = - \underbrace{\sum_{t=1}^{T_{seq}} \mathbb{E}_{q_\psi(\theta_t)} [\log p_\phi(\mathbf{x}_t | \theta_t)]}_{\mathbf{LL}} + \underbrace{\sum_{t=2}^{T_{seq}} D_{KL}(q_\psi(\theta_t) || p_\phi(\theta_t | \theta_{1:t-1}))}_{\mathbf{KL}}, \quad (3.14)$$

where $\mathbf{x}_t, \theta_t$ are the t -th element in sequence n of data and latent logit, respectively. The (average) loss for a randomly drawn batch $B \subseteq [N_{seq}]$ is

$$\mathcal{L}(\phi, \psi) = \frac{1}{|B|} \sum_{n \in B} \mathcal{L}_n(\phi, \psi). \quad (3.15)$$

Notice here each sequence can be seen as a data point and $\mathcal{L}_n(\phi, \psi)$ is the loss associated with this data point.

The need to break data into sequences presents a limitation of DyNeMo, being that it cannot capture super long range temporal dependencies (e.g. more than 10 seconds), which have been shown to exist and play an important role in sleep. We attempted to tackle this challenge by using Gaussian processes instead of RNNs to model the temporal generative process (see Appendix E).

3.2.3 Tricks for training DyNeMo

Annealing the **KL** term

Optimising the parameters of a VAE with RNN is challenging. In most runs, the **KL** term dominates and the model consistently sets the variational distribution to be close to the prior distribution of the latent variables, yielding a very small **KL** term ([Bowman et al., 2015]). The solution to this is surprisingly simple and effective: annealing the **KL** term. The idea is to train the model with a modified loss function where the **KL** term is multiplied by an annealing factor κ that starts at 0 and gradually increases to 1 as the training progresses, i.e.

$$\mathcal{L}_n = -\mathbf{LL} + \kappa \mathbf{KL}. \quad (3.16)$$

This allows the model to first learn a good encoder before tuning the variational distribution to match the prior distribution. In this thesis, we use a tanh scheme for the annealing factor κ :

$$\kappa = \begin{cases} 0 & \text{if } epoch = 1 \\ 1 & \text{if } epoch \geq N_{anneal} \\ \frac{1}{2} \tanh \left(sharpness \frac{[(epoch-1) \bmod N_{anneal}] - N_{anneal}/2}{N_{anneal}} \right) + \frac{1}{2} & \text{otherwise,} \end{cases} \quad (3.17)$$

where N_{anneal} is the number of annealing epochs beyond which $\kappa = 1$ and *sharpness* is a hyper-parameter that controls the steepness of how κ increases.

The trick of annealing the **KL** term during training involves tuning many hyper-parameters. Recently there have been more principled ways for training VAEs (e.g. [Rezende and Viola, 2018b, Rezende and Viola, 2018a]), and these methods can be explored for better training of the model.

Multi-start initialisation

The DyNeMo loss is highly non-convex and the model can end up in different local minima depending on the initialisation. To mitigate this, a multi-start initialisation strategy is used. At each initialisation, the model parameters are initialised randomly and the model is trained for a small number of epochs. This is repeated multiple times and the model with the lowest loss is chosen for further training.

Learning covariance matrices

A square matrix $\Sigma \in \mathbb{R}^{n \times n}$ is a legitimate covariance matrix if and only if it is

- Symmetric. $\Sigma = \Sigma^T$.
- Strictly positive definite. $\mathbf{v}^T \Sigma \mathbf{v} > 0$, $\forall \mathbf{v} \in \mathbb{R}^n$.

Hence we cannot simply learn all the entries of the covariances Σ_j as there are too many degrees of freedom and there is no guarantee that we learn a legitimate covariance matrix. Instead, we use the fact that there is a bijection between the space of strictly positive definite matrices and the space of lower-triangular matrices with strictly positive diagonals through Cholesky decomposition. In *osl-dynamics*, a vector of length $N_c(N_c + 1)/2$ is learned before it is filled as the entries of a lower triangular matrix whose diagonal is softplus transformed. The resulting matrix is multiplied with its own transpose to get the covariance matrix.

3.3 Comparing different generative models

Qualitatively, DyNeMo gives a description of the observed data under more relaxed assumptions than the HMM, and it has been shown in [Gohil et al., 2022] that DyNeMo is able to infer binarised mode time courses if the data is indeed generated from the HMM generative model, meaning when fitted on the real data, there is nothing in the generative model or model training that forces soft-mixing of mode

time courses. It is the solely the data itself that determines the soft-mixing nature of the inferred mode time courses. However, we ask the question if there is a **quantitative** way to compare HMM and DyNeMo? This is also closely related to the classic problem of choosing the number of states in HMM and the number of modes in DyNeMo.

3.3.1 Calculation of free energy

A natural choice of metric for goodness of fit is the loss function during training, i.e. the variational free energy.

Free energy of HMM-based models

For the HMM, the free energy can be written as

$$\begin{aligned}
\mathcal{F} &= \int q(z_{1:T}) \log \left[\frac{q(z_{1:T})}{p(\mathbf{x}_{1:T}, z_{1:T})} \right] dz_{1:T}, \\
&= \int q(z_{1:T}) \log \left[\frac{q(z_{1:T})}{p(\mathbf{x}_1|z_1)p(s_1) \prod_{t=2}^T p(\mathbf{x}_t|z_t)p(z_t|z_{t-1})} \right] dz_{1:T}, \\
&= - \int q(z_{1:T}) \log \left[\prod_{t=1}^T p(\mathbf{x}_t|z_t) \right] dz_{1:T} + \int q(z_{1:T}) \log \left[\frac{q(z_{1:T})}{p(z_1) \prod_{t=2}^T p(z_t|z_{t-1})} \right] dz_{1:T}, \\
&= \underbrace{- \int q(z_{1:T}) \log \left[\prod_{t=1}^T p(\mathbf{x}_t|z_t) \right] dz_{1:T}}_{\mathbf{LL}} \\
&\quad + \underbrace{\int q(z_{1:T}) \log q(z_{1:T}) dz_{1:T}}_{\mathbf{E}} \\
&\quad - \underbrace{\int q(z_{1:T}) \log \left[p(z_1) \prod_{t=2}^T p(z_t|z_{t-1}) \right] dz_{1:T}}_{\mathbf{P}},
\end{aligned} \tag{3.18}$$

where $\mathbf{LL}$ is the posterior expected log-likelihood, $\mathbf{E}$ is the posterior entropy and $\mathbf{P}$ is the posterior expected prior probability. Notice $\mathbf{E} - \mathbf{P}$ is the usual KL divergence

term. Each of the terms can be simplified and it can be shown that

$$\mathbf{LL} = \sum_{t=1}^T \sum_{k=1}^K \gamma_k(t) \log \mathcal{N}(\mathbf{x}_t | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \quad (3.19)$$

$$\mathbf{E} = \sum_{t=1}^{T-1} \sum_{k=1}^K \sum_{l=1}^K \xi_{kl}(t) \log \xi_{kl}(t) - \sum_{t=2}^{T-1} \sum_{k=1}^K \gamma_k(t) \log \gamma_k(t), \quad (3.20)$$

$$\mathbf{P} = \sum_{k=1}^K \gamma_k(1) \log \pi_k + \sum_{t=1}^{T-1} \sum_{k=1}^K \sum_{l=1}^K \xi_{kl}(t) \log A_{kl}, \quad (3.21)$$

where $\gamma(t) = q(z_t)$, $\xi(t) = q(z_t, z_{t+1})$. For details of derivation, we refer the readers to <https://github.com/OHBA-analysis/osl-dynamics/blob/main/doc/models/images/hmm-cost-function.pdf>.

For HIVE (Chapter 5), we are not able to get the exact free energy due to the fact that the posterior expected log-likelihood involves an intractable integral with respect to the variational distribution of deviation magnitudes. We can estimate it by plugging in the MAP estimate of deviation magnitudes.

Free energy of DyNeMo-based models

Training of DyNeMo-based models, including DyNeMo, M-DyNeMo (Chapter 4) and DIVE (Appendix D), directly optimises the variational free energy. When calculating the free energy, one can choose to plug in the MAP estimate of hidden variables or use a Monte-Carlo estimate. In practice, it is found that although there are differences between the two approaches, they follow the general trend, i.e. models with lower plug-in estimate generally have lower MC estimate of free energy, and vice versa.

3.3.2 Calculating model evidence

Some preliminary work was done using free energy as a metric of determining the number of states in HMM and the number of modes in DyNeMo. Although the metric is behaving itself in simulated datasets, meaning the trace of free energy flat-

tens out when going beyond the ground truth number of states (see Figure 5.13a), its performance on real datasets has been sub-optimal as the free energy kept decreasing with the number of states/modes ([Baker et al., 2014, Vidaurre et al., 2016, Gohil et al., 2022]). One hypothesis is this is a result of variability within datasets (Section 5.11), another hypothesis is that the free energy is not a good metric to use. As discussed in Section 2.2.3, the ELBO (negative free energy) is a lower bound of the model evidence or the marginal log-likelihood of the data, which is what we want to optimise initially.

Exact model evidence of the HMM

In the HMM, although we cannot optimise the model evidence directly, it is possible to calculate the exact model evidence of a trained HMM with an iterative approach. The HMM model evidence $\log p(\mathbf{x}_{1:T})$ can be written as

$$\log p(\mathbf{x}_{1:T}) = \log p(\mathbf{x}_1) + \sum_{t=2}^T \log p(\mathbf{x}_t | \mathbf{x}_{1:t-1}). \quad (3.22)$$

Thus it suffices to calculate each of the conditional probabilities $p(\mathbf{x}_t | \mathbf{x}_{1:t-1})$. Now consider the following algorithm iterating between the *prediction step* and *update step*. For $t \in [T]$,

- **Prediction step:**

$$p(s_t | \mathbf{x}_{1:t-1}) = \int p(s_t | s_{t-1}) p(s_{t-1} | \mathbf{x}_{1:t-1}) ds_{t-1} \quad (3.23)$$

- **Update step:**

$$\begin{aligned} p(\mathbf{x}_t | \mathbf{x}_{1:t-1}) &= \int p(\mathbf{x}_t | s_t) p(s_t | \mathbf{x}_{1:t}) ds_t, \\ p(s_t | \mathbf{x}_{1:t}) &= \frac{p(\mathbf{x}_t | s_t) p(s_t | \mathbf{x}_{1:t-1})}{p(\mathbf{x}_t | \mathbf{x}_{1:t-1})}, \end{aligned} \quad (3.24)$$

where $p(s_t | \mathbf{x}_{1:t-1}) := p(s_1)$ if $t = 1$. In our case, due to the fact that we have a simple finite state space HMM, the integrals become sums. To improve numerical

stability, we perform the calculations in the log space, resulting in the algorithm summarised in Algorithm 1.

Algorithm 1: HMM exact evidence

Input: Initial distribution $p(s_1)$, transition probabilities $p(s_t|s_{t-1})$, conditional likelihood $p(\mathbf{x}_t|s_t)$

Output: Model evidence l

Procedure:

$l \leftarrow 0$

for $t = 1, \dots, T$ **do**

Prediction step:

if $t = 1$ **then**

$l_j^p \leftarrow \log p(s_1 = j), j \in [J]$

else

$l_j^p \leftarrow \log \sum_i \exp[\log p(s_t = j|s_{t-1} = i) + l_i^u], j \in [J]$

Update step:

$l_j^u \leftarrow \log p(\mathbf{x}_t|s_t = j) + l_j^p, j \in [J]$

$l^* \leftarrow \log \sum_i \exp(l_i^u)$

$l_j^u \leftarrow l_j^u - l^*, j \in [J]$

$l \leftarrow l + l^*$

return l

Estimating model evidence of DyNeMo-based models

There is no easy way to compute the exact model evidence of DyNeMo given the complexity of the generative model and neural networks involved. Notice that the marginal likelihood is the normalising constant in for the posterior distribution:

$$p(\boldsymbol{\theta}_{1:T}|\mathbf{x}_{1:T}) = \frac{p(\mathbf{x}_{1:T}|\boldsymbol{\theta}_{1:T})p(\boldsymbol{\theta}_{1:T})}{p(\mathbf{x}_{1:T})} \quad (3.25)$$

This motivates the use of importance sampling ([Tokdar and Kass, 2010]) targeting the posterior distribution to estimate the normalising constant. First consider the simple case where we want to target the posterior distribution with $t = 1$:

$$p(\boldsymbol{\theta}_1|\mathbf{x}_1) \propto p(\mathbf{x}_1|\boldsymbol{\theta}_1)p(\boldsymbol{\theta}_1). \quad (3.26)$$

Let q_1 be our proposal distribution and we have the normalised importance weight function as:

$$w_1(\boldsymbol{\theta}_1) = \frac{p(\boldsymbol{\theta}_1)p(\mathbf{x}_1|\boldsymbol{\theta}_1)}{q_1(\boldsymbol{\theta}_1)}. \quad (3.27)$$

Now we sample M i.i.d. samples from the proposal

$$\boldsymbol{\theta}_1^{(1)}, \dots, \boldsymbol{\theta}_1^{(M)} \stackrel{i.i.d.}{\sim} q_1(\boldsymbol{\theta}_1), \quad (3.28)$$

and get the importance weights as

$$w_1^{(m)} = \frac{p(\boldsymbol{\theta}_1^{(m)})p(\mathbf{x}_1|\boldsymbol{\theta}_1^{(m)})}{q_1(\boldsymbol{\theta}_1^{(m)})} \quad \forall m \in [M]. \quad (3.29)$$

Then it is easy to show the normalised importance sampling estimate for $p(\mathbf{x}_1)$

$$p^M(\mathbf{x}_1) = \frac{1}{M} \sum_{m=1}^M w_1^{(m)} \xrightarrow{a.s.} p(\mathbf{x}_1) \quad (3.30)$$

as $M \rightarrow \infty$. Now we want to consider targeting the more general posterior distribution. Suppose, at time $t-1$, we have already approximated the posterior distribution $p(\boldsymbol{\theta}_{1:t-1}|\mathbf{x}_{1:t-1})$ with the proposal distribution q_{t-1} with importance weight function

$$w_{t-1}(\boldsymbol{\theta}_{1:t-1}) \propto \frac{p(\boldsymbol{\theta}_{1:t-1}|\mathbf{x}_{1:t-1})}{q_{t-1}(\boldsymbol{\theta}_{1:t-1})}. \quad (3.31)$$

Assume further that we have M i.i.d. sample paths from the proposal distribution:

$$\boldsymbol{\theta}_{1:t-1}^{(1)}, \dots, \boldsymbol{\theta}_{1:t-1}^{(M)} \stackrel{i.i.d.}{\sim} q_{t-1}(\boldsymbol{\theta}_{1:t-1}), \quad (3.32)$$

where $\boldsymbol{\theta}_{1:t-1}^{(m)}$ is the m -th sample, and importance weights

$$w_{t-1}^{(m)} \propto \frac{p(\boldsymbol{\theta}_{1:t-1}^{(m)}|\mathbf{x}_{1:t-1})}{q_{t-1}(\boldsymbol{\theta}_{1:t-1}^{(m)})}. \quad (3.33)$$

The idea is to approximate the posterior distribution at time t by updating the importance weights from the previous step. We introduce the conditional proposal $q_{t|t-1}(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{1:t-1})$ and draw samples with

$$\boldsymbol{\theta}_t^{(m)} \sim q_{t|t-1}(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{1:t-1}^{(m)}). \quad (3.34)$$

Then the sampled trajectories/paths follow the distribution

$$\boldsymbol{\theta}_{1:t}^{(m)} \sim q_{t-1}(\boldsymbol{\theta}_{1:t-1})q_{t|t-1}(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{1:t-1}) := q_t(\boldsymbol{\theta}_{1:t}). \quad (3.35)$$

Notice that with the proposal distribution constructed in this conditional way, the importance weight function targeting $p(\boldsymbol{\theta}_{1:t}|\mathbf{x}_{1:t})$ with proposal $q_t(\boldsymbol{\theta}_{1:t})$ is

$$\begin{aligned} w_t(\boldsymbol{\theta}_{1:t}) &\propto \frac{p(\boldsymbol{\theta}_{1:t}|\mathbf{x}_{1:t})}{q_t(\boldsymbol{\theta}_{1:t})} \\ &\propto \frac{p(\boldsymbol{\theta}_{1:t}, \mathbf{x}_{1:t})}{q_t(\boldsymbol{\theta}_{1:t})} \\ &\propto \frac{p(\mathbf{x}_t|\boldsymbol{\theta}_t)p(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{1:t-1})p(\boldsymbol{\theta}_{1:t-1}|\mathbf{x}_{1:t-1})}{q_{t|t-1}(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{1:t-1})q_{t-1}(\boldsymbol{\theta}_{1:t-1})} \\ &= \frac{p(\mathbf{x}_t|\boldsymbol{\theta}_t)p(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{1:t-1})}{q_{t|t-1}(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{1:t-1})}w_{t-1}(\boldsymbol{\theta}_{1:t-1}). \end{aligned} \quad (3.36)$$

Hence we have found a way to update the importance weights with previous calculated weights without sampling the whole path again (which is why a Markov Chain Monte Carlo (MCMC, [Hastings, 1970]) targeting the posterior is not ideal in our case):

$$w_t^{(m)} = \frac{p(\mathbf{x}_t|\boldsymbol{\theta}_t^{(m)})p(\boldsymbol{\theta}_t^{(m)}|\boldsymbol{\theta}_{1:t-1}^{(m)})}{q_{t|t-1}(\boldsymbol{\theta}_t^{(m)}|\boldsymbol{\theta}_{1:t-1}^{(m)})}w_{t-1}^{(m)}. \quad (3.37)$$

Now with these weights, we can approximate the predictive likelihood with

$$p^M(\mathbf{x}_t|\mathbf{x}_{1:t-1}) = \frac{\sum_{m=1}^M w_t^{(m)}}{\sum_{m=1}^M w_{t-1}^{(m)}}, \quad (3.38)$$

and the model evidence can be approximated by

$$\log p(\mathbf{x}_{1:T}) \approx \log p^M(\mathbf{x}_{1:T}) := \sum_{t=2}^T \log p^M(\mathbf{x}_t | \mathbf{x}_{1:t-1}). \quad (3.39)$$

This algorithm is generally called *sequential importance sampling* (SIS).

Resampling

The SIS algorithm provides a way to update weights when new observations are obtained, but there is a pitfall. As the sequence length gets longer, it is often the case that all but a few weights are non-zero, resulting in low effective sample size (ESS) and large variance in the estimators. A solution to this is through resampling of particles $\{(w_t^{(m)}, \boldsymbol{\theta}_{1:t}^{(m)}) : m \in [M]\}$. During resampling, an “ancestry” variable $A^{(m)} \sim \text{Multi}(w_t^{(1)}, \dots, w_t^{(M)})$ is drawn and the particle $\overline{\boldsymbol{\theta}_t^{(m)}}$ is set to $\boldsymbol{\theta}_t^{(A^{(m)})}$ for all $m \in [M]$ and all of the weights are set to $1/M$. It is easy to show that this resampling scheme is consistent in the sense that it keeps the expectation with respect to the empirical distribution of the particles unchanged. In practice, the resampling is performed whenever the ESS goes below a threshold. The SIS algorithm with resampling is often called a *particle filter* or *sequential Monte Carlo* ([Del Moral et al., 2006]).

Simulation result

The particle filter is also applicable for estimating the model evidence of the HMM, whose exact model evidence can be computed (Algorithm 1). We validate the particle filter on simulated data generated with the HMM generative model ($J = 5, T = 25, 600$, see Appendix A.2). Here we perform 10 folds cross validation and train the HMM with increasing number of states ($2 - 10$) and calculate the exact model evidence, the negative free energy and the particle filter estimated model evidence of the trained HMM with each number of states on left out data of each fold. We can

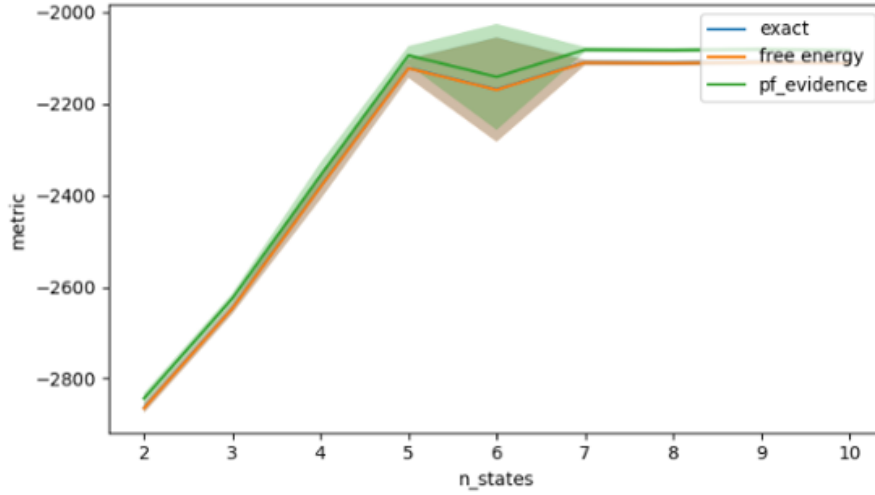


Figure 3.1: **Particle filter finds the correct number of states.** Exact model evidence (blue, covered by orange line), negative free energy (orange) and estimated model evidence with particle filter (green) of the trained HMM is plotted on the y-axis against the number of states. The shades show one standard deviation of 10 folds cross validation.

see in Figure 3.1 that all three metrics finds the correct number of states (plateau after 5 states) and while the particle filter overestimates model evidence, the error is reasonably small.

The above preliminary results of using the particle filter was shown with simulated data and the algorithm seems to behave well. However, there is no ground truth and it is hard to quantify the variance of the estimator, which makes it hard to justify to use the particle filter estimate of model evidence as a way to compare models. We decided it is not worth the time to dig deeper in this path.

3.4 Discussion

In this chapter, we gave a description of popular generative models, the HMM and DyNeMo, for inferring dynamic brain networks. Although work has been done to qualitatively show advantages of DyNeMo over the HMM, it is still paramount to have a concrete way to compare them qualitatively, given the observation that the objective function of both methods keeps decreasing with increasing model com-

plexity (number of state/modes).

We attempted to use model evidence, which is the original object we were trying to optimise, instead of its lower bound, the free energy. However, it turns out the model evidence has the similar behaviour of ever-decreasing with model complexity. Another route is by using the so-called bi-cross validation method for determining the optimal number of clusters in K-Means clustering ([Fu and Perry, 2020]) and there has been some promising results on determining the optimal number of states of HMM in fMRI data.

Apart from the new models developed and described in Chapter 4 and Chapter 5, improvements and additional features to the HMM and DyNeMo are also introduced in the *osl-dynamics* toolbox, including parameter regularisation, better model initialisation for more stable training, different ways of estimating individual-level networks from a trained group-level model, etc. Furthermore, there is the option to utilise GPU for faster and parallelised training of the HMM, and a new data pipeline that allows efficient distributed training on enormous datasets, which was not possible before.

Chapter 4

Multi-dynamic modelling of MEG data

4.1 Introduction

Generative models for inferring dynamic brain networks mentioned so far, including the HMM (Section 3.1) and DyNeMo (Section 3.2), make the underlying assumption that the dynamics of time-varying power and FC are the same (i.e. coupled). But there is no principled reason to assume fluctuation in connections between brain regions should be coupled with that in power of individual brain regions. This motivates us to present Multi-dynamic Network Modes (M-DyNeMo), an extension to DyNeMo, to infer potentially uncoupled power and FC dynamics.

In this chapter, with resting-state MEG data, we present evidence of separate power and FC dynamics given by the traditional sliding window approach, and show that plausible networks of separate dynamics can be inferred by M-DyNeMo. We further demonstrate that the networks are reproducible across different subjects, datasets, and parcellations, validating the robustness of M-DyNeMo on MEG data. Using visual task data, M-DyNeMo reveals distinct evoked network responses of power and FC dynamics to visual stimuli. We also show evidence for task-induced

coupling between power and FC dynamics.

Source code for M-DyNeMo is available in the *osl-dynamics* toolbox and scripts to reproduce results in this manuscript are available at:

https://github.com/OHBA-analysis/Huang2024_EvidenceForUncoupledDynamics.

4.2 Structural equation modelling motivates multi-dynamic modelling.

We can motivate multi-dynamic modelling by considering a dynamic version of structural equation modelling (SEM, [Penny et al., 2004]):

$$\mathbf{x}_t = \mathbf{M}_t \mathbf{x}_t + \mathbf{e}_t, \quad (4.1)$$

where $\mathbf{x}_t$ is the vector of brain signal at time t for N_c channels, $\mathbf{M}_t$ is an $N_c \times N_c$ matrix representing the connectivity information at time t , with $\text{diag}(\mathbf{M}_t) = 0$, and $\mathbf{e}_t \sim \mathcal{N}(0, \boldsymbol{\sigma}_t^2 \mathbf{I}_{N_c})$ is the noise at time t .

Rearranging Equation (4.1) gives

$$\mathbf{x}_t = (\mathbf{I}_{N_c} - \mathbf{M}_t)^{-1} \mathbf{e}_t, \quad (4.2)$$

from which we can see $\mathbf{x}_t$ is normally distributed with mean 0 and covariance matrix $(\mathbf{I}_{N_c} - \mathbf{M}_t)^{-1} (\boldsymbol{\sigma}_t^2 \mathbf{I}_{N_c}) (\mathbf{I}_{N_c} - \mathbf{M}_t)^{-T}$. Notice that the dynamics of the standard deviation of $\mathbf{x}_t$ depends only on the dynamics of $\boldsymbol{\sigma}_t$ and the dynamics of the correlation matrix of $\mathbf{x}_t$ depends only on the dynamics of $\mathbf{M}_t$. In other words, if $\boldsymbol{\sigma}_t$ and $\mathbf{M}_t$ follow distinct dynamics, we should expect the dynamics of the standard deviation and the correlation of $\mathbf{x}_t$ to be uncoupled.

4.3 Results

In this section, we study three real MEG datasets (Section 4.6.1): MEGUK-38 (resting-state), MEGUK-52, and Wakeman-Henson. For choices of hyper-parameters see Appendix B.5.

4.3.1 Sliding window approach reveals distinct power and FC dynamics

Firstly, we apply the well-established sliding window approach followed by K-Means clustering to identify two separate dynamics according to time-varying (TV) power (variance) and TVFC (correlation) respectively (See Section 4.6.3, [Allen et al., 2014]) from amplitude time courses (1-45Hz) of 38 brain regions obtained from MEGUK-38 resting-state data. Figure 4.1a illustrates the first 20 seconds of the state time courses (K-Means cluster assignments) for the first subject in the MEGUK-38 resting-state data. Qualitatively, we can see that TV power and TVFC does not follow the same dynamics, and the correlation between the two time courses is low, as shown in Figure 4.1b. Furthermore, each of the two state time courses corresponds to reasonable power and FC maps respectively, demonstrated in Figure 4.1c.

While Figure 4.1 provides preliminary evidence that power and FC have distinct dynamics, we do not know if this is an artefact of the choice of arbitrary hyperparameters (e.g. window size) or the inability of sliding window approaches to automatically adapt over time to fast dynamics (Section B.1.2 demonstrates this on simulated data). We instead need a method that can infer multiple dynamics in an adaptive, data-driven way. This motivates a new approach, Multi-dynamic Network Modes (M-DyNeMo).

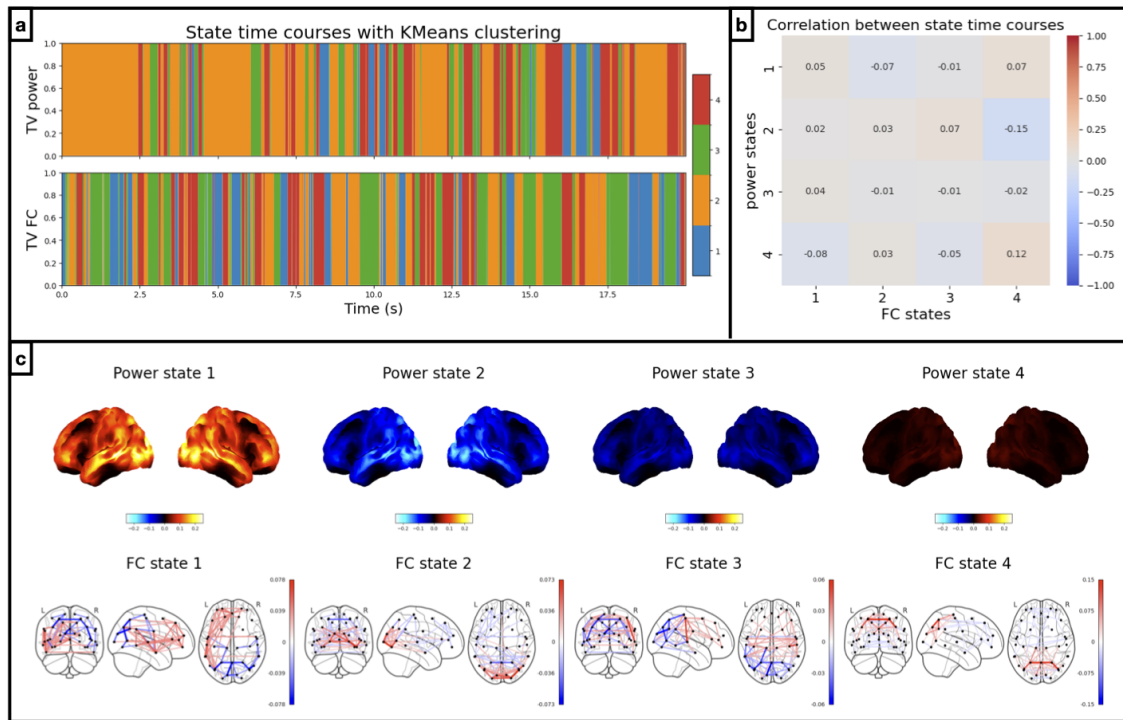


Figure 4.1: **Sliding window approach reveals distinct power and FC dynamics.** Results from amplitude time courses (1-45Hz) of 38 brain regions obtained from the MEGUK-38 resting-state data. a) K-Means state time courses for TV variance (i.e. power) (top row) and TVFC (i.e. correlation) (bottom row), using 2s sliding windows. Different colours indicate activation of different brain states and only the first 20s of the first subject is shown. b) Correlation between power (y-axis) and FC state time courses (x-axis). c) Power maps of each K-Means state are plotted on the top row. Red areas indicate brain areas with higher power (and blue areas lower power) than the average across states. FC maps are plotted on the bottom row. Red edges indicate positive correlations (and blue edges negative correlations) between brain areas.

4.3.2 Power and FC have similar spatial patterns but distinct dynamics

A formal description of M-DyNeMo is available in the Methods section (Section 4.6), including the generative model and the inference framework to fit the model to data. Conceptually, M-DyNeMo learns two basis sets of spatial configurations of networks (called *modes*), one for power (where the network is a spatial map over brain regions) and another for the FC (where the network is an edge-wise connectome), in contrast to DyNeMo where there is only one basis set of networks. The underlying assumption is that fluctuations in power and FC can be described independently via dynamics in the mixing of their respective basis sets. We can show using simulated data that M-DyNeMo performs more accurate inference than standard DyNeMo, if the ground truth contains distinct dynamics for power and FC (see Appendix B.1.2).

First, we trained M-DyNeMo on the amplitude time courses (1-45Hz) of 38 brain regions obtained from MEGUK-38 resting-state data. Figure 4.2a shows the re-normalised mode time courses for the first 8 seconds of the first subject in this dataset (See Appendix B.4.1 for details on re-normalisation). We see qualitatively that mode time courses for power (α) and for FC (β) possess very different characteristics, in the sense that β is much more binarised than α .

Power and FC networks for each mode are shown in Figure 4.2b. These networks resemble those found in multiple previous studies ([Baker et al., 2014, Vidaurre et al., 2016, Gohil et al., 2022, Gohil et al., 2024b]). For a clearer visual comparison, we also show the FC degree maps, each of which plot the sum of correlations for each node, relative to the average across modes. Note, the ordering of spatial modes found by the model is arbitrary, hence the network maps shown in this figure have been re-ordered post-hoc to match the spatial patterns of activity in power and FC for each pair of modes. Even though there is no requirement for them to be similar, there is good correspondence between the spatial patterns of the power and the FC spatial

maps. Specifically, there is significantly high cosine similarities between power and FC degree maps (0.98, 0.65 and 0.74 for mode pairs 1, 2 and 4 respectively) under a maximum statistic permutation test (see Appendix B.4.2 for details), illustrated by the diagonal entries in Figure 4.2c.

Although we identify power and FC modes that show activity in the same regions, their corresponding mode time courses (dynamics, α vs β) are not significantly correlated (under a maximum statistic permutation test across subjects), as demonstrated in Figure 4.2d, plotted on the right, suggesting time points with high power in a particular region may not necessarily also have high FC in the same region. The correlations within the power mode dynamics (α , left) and within the FC mode dynamics (β , middle) are mostly negative, indicating when a mode activates the others deactivate.

In summary, Figure 4.2 shows that power and FC have similar spatial patterns but distinct dynamics. However, one concern is that the distinct dynamics might be enforced by the assumptions made by M-DyNeMo. To mitigate this concern, we used simulations in Appendix B.1.3 to show that if the data is generated using a single, shared dynamic for power and FC, then M-DyNeMo does not incorrectly infer distinct dynamics.

4.3.3 Single dynamic DyNeMo is dominated by time-varying power

Given the evidence that there are distinct dynamics, we were interested to learn which of the dynamics dominate the decomposition when only a single dynamic is allowed, as is the case in standard DyNeMo. To investigate this we assumed that the power and FC share a single dynamic given by the power mode time course α , extracted from a trained M-DyNeMo, and then re-calculated the power and FC for each mode. The resulting networks are shown in Figure 4.3a. Comparing this with the networks inferred by standard DyNeMo on the same amplitude time

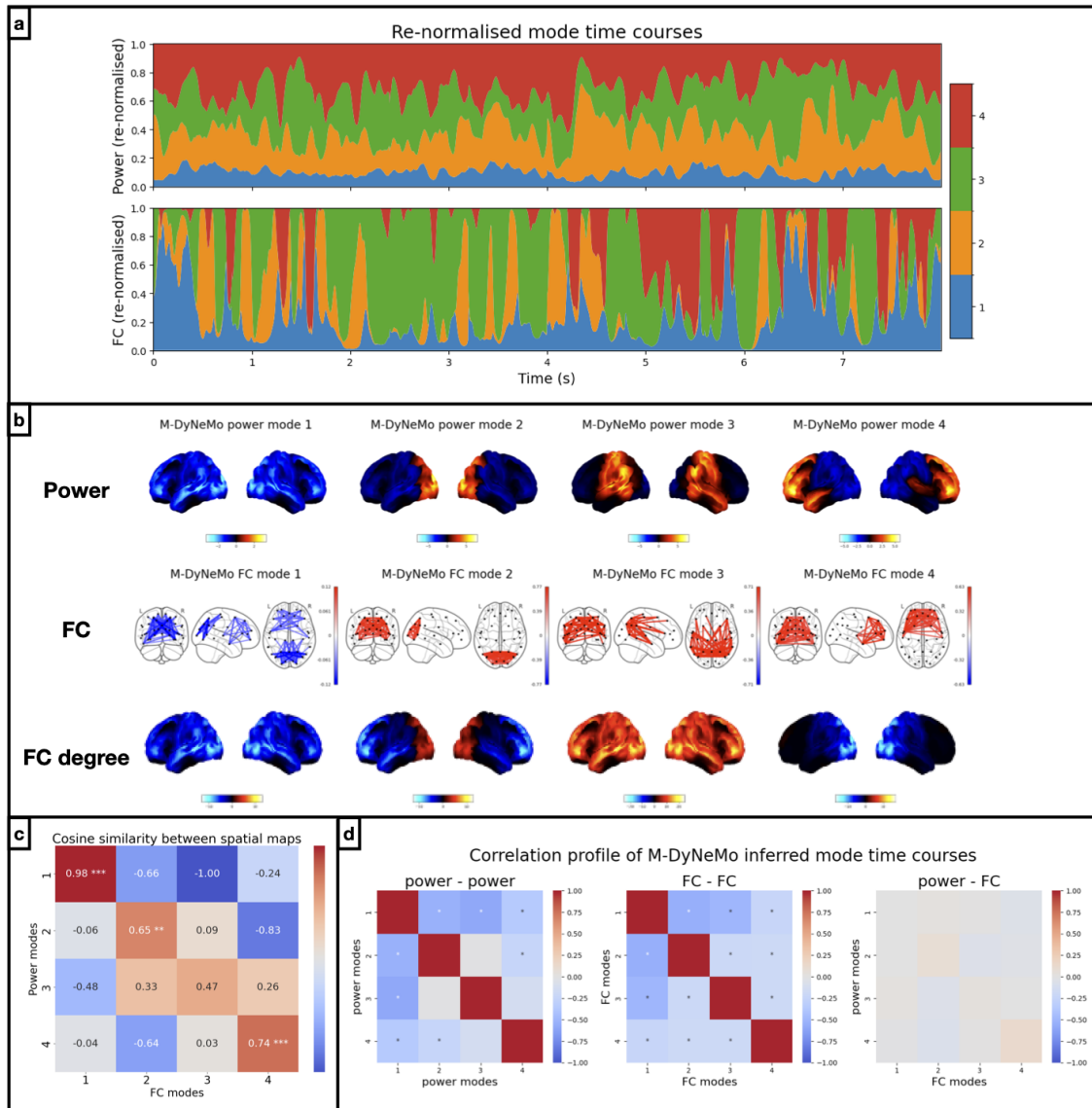


Figure 4.2: Power and FC have similar spatial patterns but distinct dynamics. Results of running M-DyNeMo on the amplitude time courses (1-45Hz) of 38 brain regions obtained from the MEGUK-38 resting-state data. a) Inferred mode time courses (re-normalised) for power, α (top) and FC, β (bottom). b) Top row shows power spatial modes where red areas show higher (blue areas show lower) activation than average across modes. Middle row shows FC spatial modes where red edges show positive (and blue edges show negative) connectivity. Bottom row shows the FC degree maps where red areas show higher (blue areas show lower) node degree (i.e. centrality) than average across modes. c) Pairwise cosine similarity between power and FC degree spatial maps. Diagonal entries with significantly large values are annotated with asterisks depending on the p-values. d) Correlation profile of mode time courses for within power modes (left), within FC modes (middle) and between power and FC modes. Correlations significantly (at 5% significance level) differ from zero under a maximum statistic permutation test are marked with an asterisk.

courses (1-45Hz) of 38 brain regions from the MEGUK-38 resting-state data, shown in Figure 4.3b, we can see that the power mode time course (α) inferred by M-DyNeMo contains similar information as the mode time course inferred by standard DyNeMo. Quantitatively, for each pair of modes, there is a significantly large cosine similarity (p-values all less than 0.001 under a maximum statistic permutation test) between the re-calculated networks using M-DyNeMo inferred power mode time course α and the inferred networks given by standard DyNeMo. This suggests the description provided by standard DyNeMo is dominated by the dynamics in power rather than FC. Furthermore, by re-calculating networks using M-DyNeMo inferred FC mode time course β , we show in Appendix Figure B.4 that the FC dynamics given by M-DyNeMo contains different information as the mode time course inferred by standard DyNeMo.

4.3.4 M-DyNeMo inferred networks are reproducible across subsets of data

To evaluate the reproducibility and robustness of the model, we train M-DyNeMo on different subsets of subjects of the MEGUK-38 resting-state dataset. We split 65 subjects of this dataset into two halves where the first half contains subjects 1 to 32 and the second half contains subjects 33 to 65. M-DyNeMo was trained independently on each half. Figure 4.4a and b shows that network maps inferred by M-DyNeMo on each half, with modes matched using cosine similarity, which is also shown in the figure for each pair of modes. In general, we see the same networks in each half with significantly large cosine similarity (p-values all less than 0.001) under a maximum statistic permutation test. There are only slight differences in the spatial pattern of activity in power modes 2 and 3.

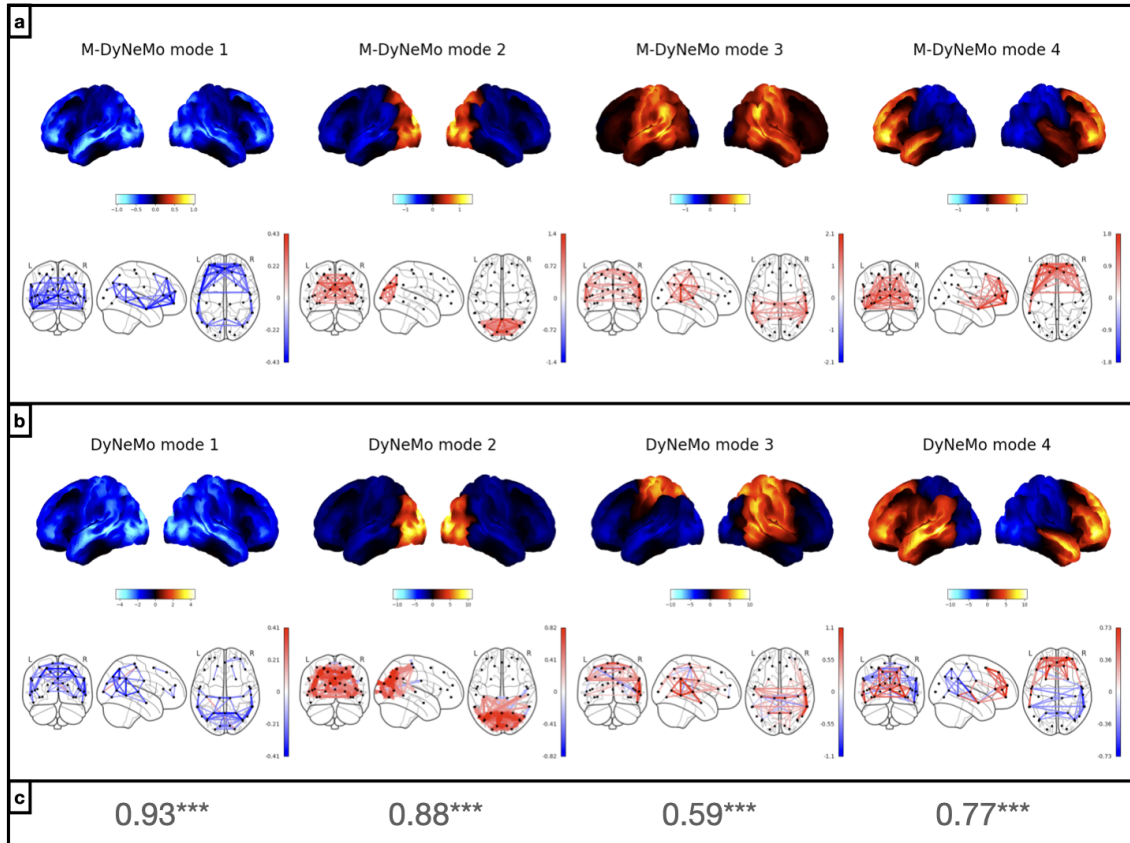


Figure 4.3: **Single dynamic DyNeMo is dominated by time-varying power.** Results from amplitude time courses (1-45Hz) of 38 brain regions obtained from MEGUK-38 resting-state data. a) Power (top row) and FC (covariance, bottom row) network maps given by re-calculating time varying covariances on M-DyNeMo inferred power mode time course α . b) Power (top row) and FC (covariance, bottom row) maps given by DyNeMo. c) Cosine similarity between re-calculated covariances on M-DyNeMo inferred power mode time course α and DyNeMo inferred covariances for each pair of modes.

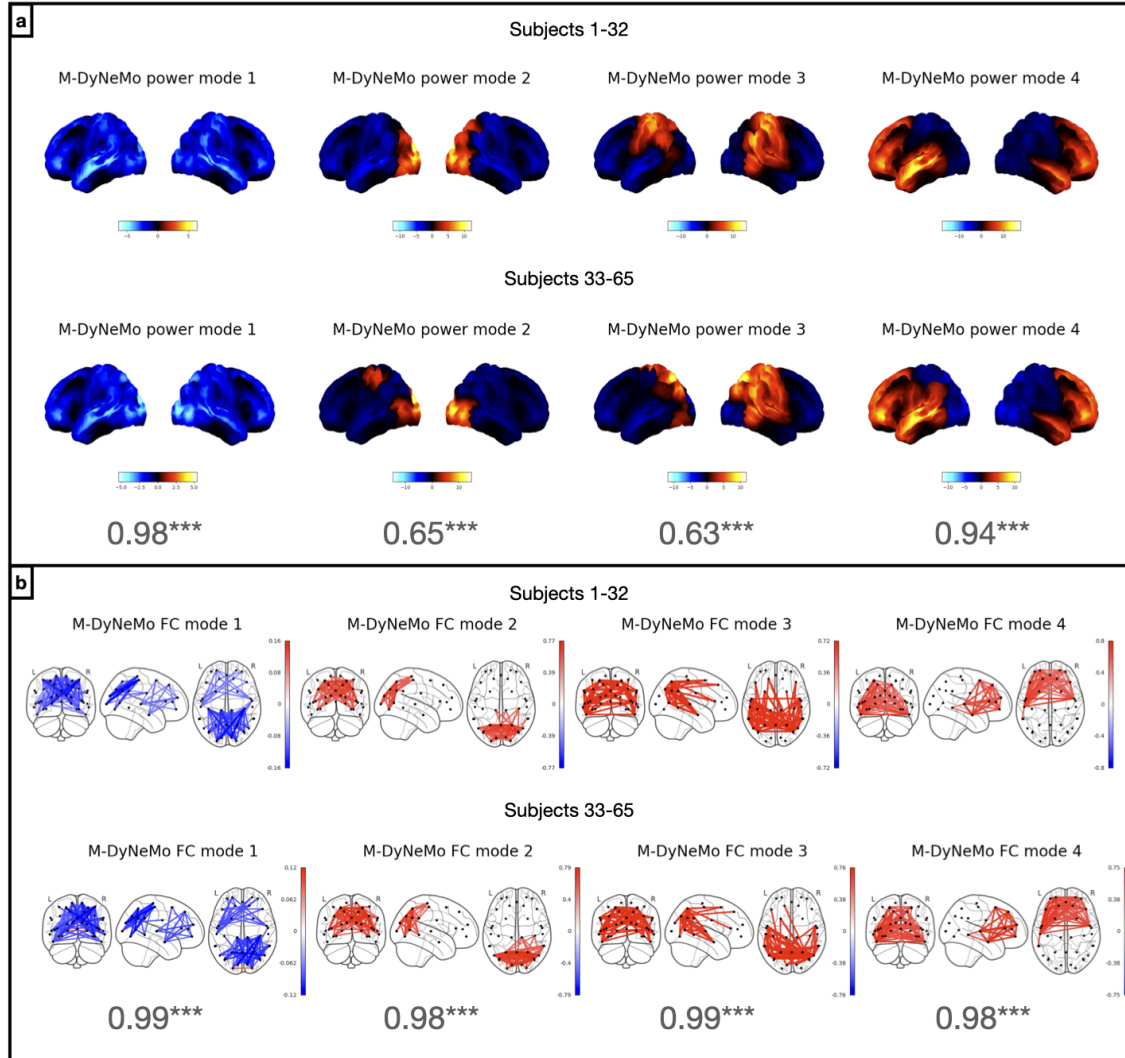


Figure 4.4: **M-DyNeMo inferred networks are reproducible across halves.** Results from amplitude time courses (1-45Hz) of 38 brain regions obtained from MEGUK-38 resting-state data. a) Power spatial modes given by training M-DyNeMo on subjects 1-32 (top row) and subjects 33-65 (bottom row) independently. The cosine similarity between power maps from each half is also shown for each pair of modes. b) FC spatial modes given by training M-DyNeMo on subjects 1-32 (top row) and subjects 33-65 (bottom row) independently. The ordering of both power and FC modes are matched with cosine similarities between networks from each half, which are shown in the figure for both power and FC networks.

4.3.5 Power and FC dynamics show different timings in evoked network response

In Section 4.3.2, we saw M-DyNeMo infers distinct dynamics (i.e. uncorrelated mode time courses) for power and FC using the amplitude time courses (1-45Hz) from resting-state MEG data. We now turn to task MEG data and ask the question: Do α and β respond differently to tasks? We perform this investigation with the Wakeman-Henson dataset, where participants are presented with images of famous, unfamiliar, or scrambled faces. This allows us to study how the mode time courses respond to visual stimuli as well as the difference in responses to different visual stimuli.

The network plots are shown in Figure 4.5a, which are comparable to those inferred from MEGUK-38 resting-state data, shown in Figure 4.2b. In Figure 4.5b, the evoked network responses are shown for different contrasts, including the average of all visual stimuli, the difference between faces (including both famous and unfamiliar faces) and scrambled faces, and the difference between famous and unfamiliar faces.

Power dynamics

We focus on the top row of Figure 4.5b first. Similar to the analysis performed in [Gohil et al., 2024b], a two-level General Linear Model (GLM, [Winkler et al., 2014]) is used for computing evoked network responses, where an evoked response analysis is carried out on each of the task epoch M-DyNeMo mode time courses. Note, that this is carried out *after* M-DyNeMo is trained, i.e. M-DyNeMo has no knowledge of the task timings. A maximum statistic permutation test is used to identify periods of significant responses.

We see in the top row of Figure 4.5b that evoked network responses for power, α , are broadly consistent with those given by training standard DyNeMo on the same data (see Appendix Figure B.6). For example, in the average visual response,

there is an immediate activation of the visual network (α mode 2) followed by an activation of the frontal network (α mode 4) and a deactivation of the visual network. In the difference between unscrambled faces (famous plus unfamiliar) and scrambled faces, there is an activation of the visual network and a delayed deactivation of the frontotemporal network. The only difference is that there is no activation of the suppressed network (α mode 1) at around 600ms after the onset like in the analysis of standard DyNeMo. There are no significant differences when we look at the responses to famous and unfamiliar faces.

FC dynamics

We now draw our attention to the bottom row of Figure 4.5b where the evoked responses of the FC network mode dynamics, β , are illustrated. For the average visual response (left), we see an immediate activation of mode 1 and deactivation of mode 3, followed by a delayed and persistent activation of mode 2 as well as a deactivation of the mode 1 and mode 4. Looking at the differences in response to faces and scrambled faces, there is only a delayed activation of mode 3 and a deactivation of the mode 1. Finally, there is a short-lived activation of the mode 3 when comparing responses to famous and unfamiliar faces.

4.3.6 The coupling between power and FC dynamics is modulated by task

As shown in Figure 4.5, there is evidence of different timings in evoked network responses for power and FC dynamics. This motivated us to investigate whether the extent to which the coupling between power dynamics and FC dynamics is different between task and rest. To this end, we trained M-DyNeMo on a new dataset, MEGUK-52, which contains 63 subjects that have both resting-state and visual task recordings, totalling 126 sessions.

Figure 4.6a shows the inferred networks, which are broadly consistent with those

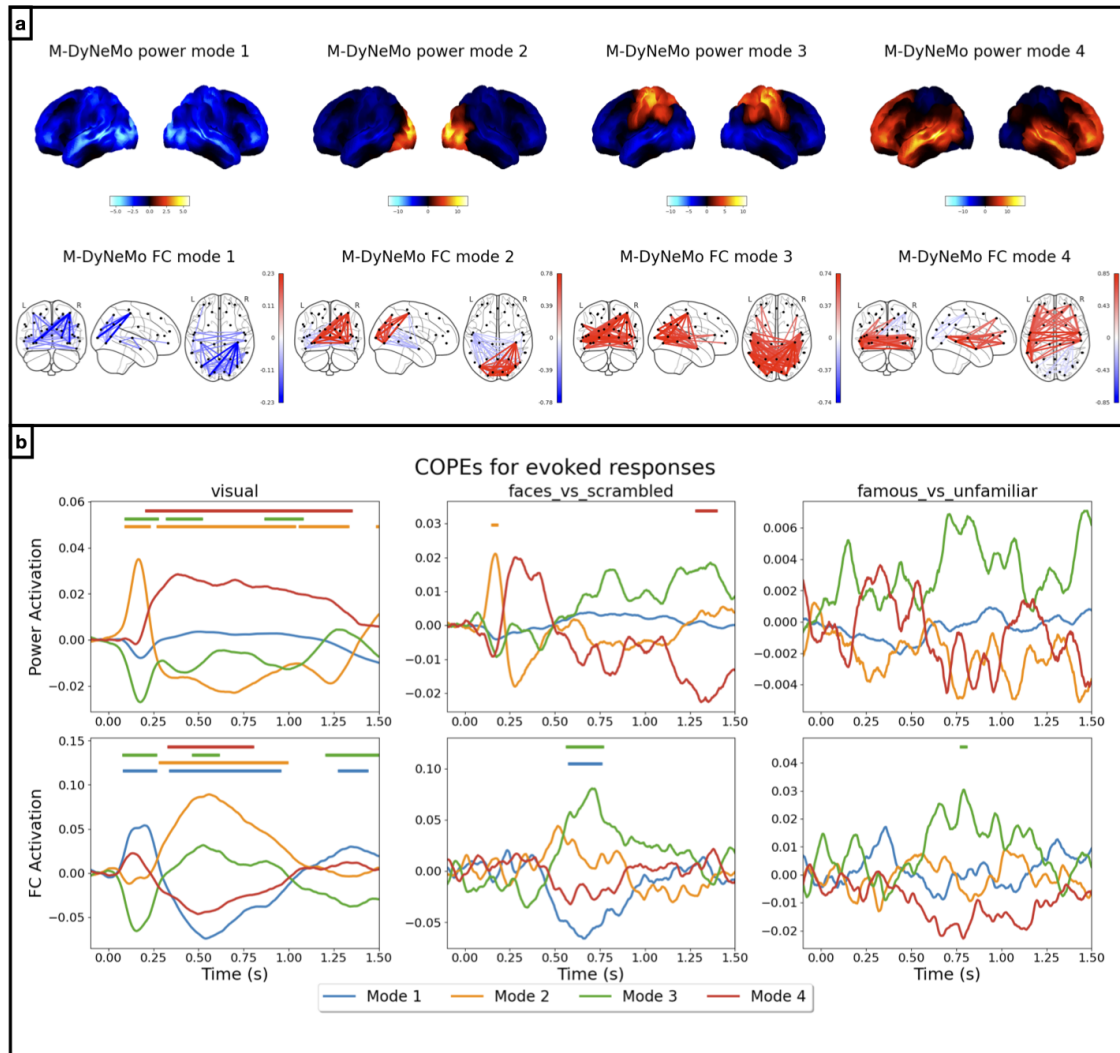


Figure 4.5: **Evoked responses have different timings in different mode time courses.** Results from amplitude time courses (1-45Hz) of 38 brain regions obtained from Wakeman-Henson data. a) Power spatial modes are shown on the top row and FC spatial modes are shown on the bottom row. b) Evoked responses for re-normalised α (top row) and β (bottom row) for different contrasts on different columns, including the average visual response (left), the difference between faces and scrambled faces (middle), and the difference between famous and unfamiliar faces (right). Here different colours represent baseline corrected evoked responses for different modes as contribution to either the overall variance (for α) or the overall correlation (for β). Periods with significant (de-)activation are indicated with a bar for each mode.

inferred on the same dataset, using a different parcellation compared to in Section 4.2 (MEGUK-38 resting-state data) and on a different dataset to Section 4.3.5 (Wakeman-Henson - which only contains visual task data and on a different set of subjects). This again demonstrates the reproducibility of M-DyNeMo on different datasets and parcellations.

The evoked network responses to visual stimuli, shown in Figure 4.6b, which is consistent with the findings in Section 4.3.5. Figure 4.6c shows the correlations between the power (α) and FC (β) dynamics during rest (left), those during epochs of 0.5 seconds before and 2 seconds after the visual stimuli (middle), as well as the difference in correlations between task and resting state. In particular, from rest to task, we see increased correlation of α mode 2 with β mode 1 and decreased correlation with β mode 2, which is consistent with the immediate activation of α mode 2 and β mode 1, and the delayed activation of β mode 2 in Figure 4.6b. It is worth noting that the sliding window approach on this dataset also provides evidence for the difference in the coupling of the two dynamics (See Appendix Figure B.7).

4.4 Discussion

With resting-state MEG data, we found preliminary evidence for the existence of distinct power and FC dynamics using a standard sliding window approach. However, this approach lacks the ability to adapt to transient dynamics at unknown time-scales in electrophysiological data. We demonstrated the deficiency of sliding window approaches to do this using simulated data in Appendix B.1.2. Therefore, we proposed a new generative model approach that automatically tunes to the required time-scales based on the data, and that can incorporate multiple dynamics: M-DyNeMo.

Using M-DyNeMo, we show that while they have similar spatial patterns, the two

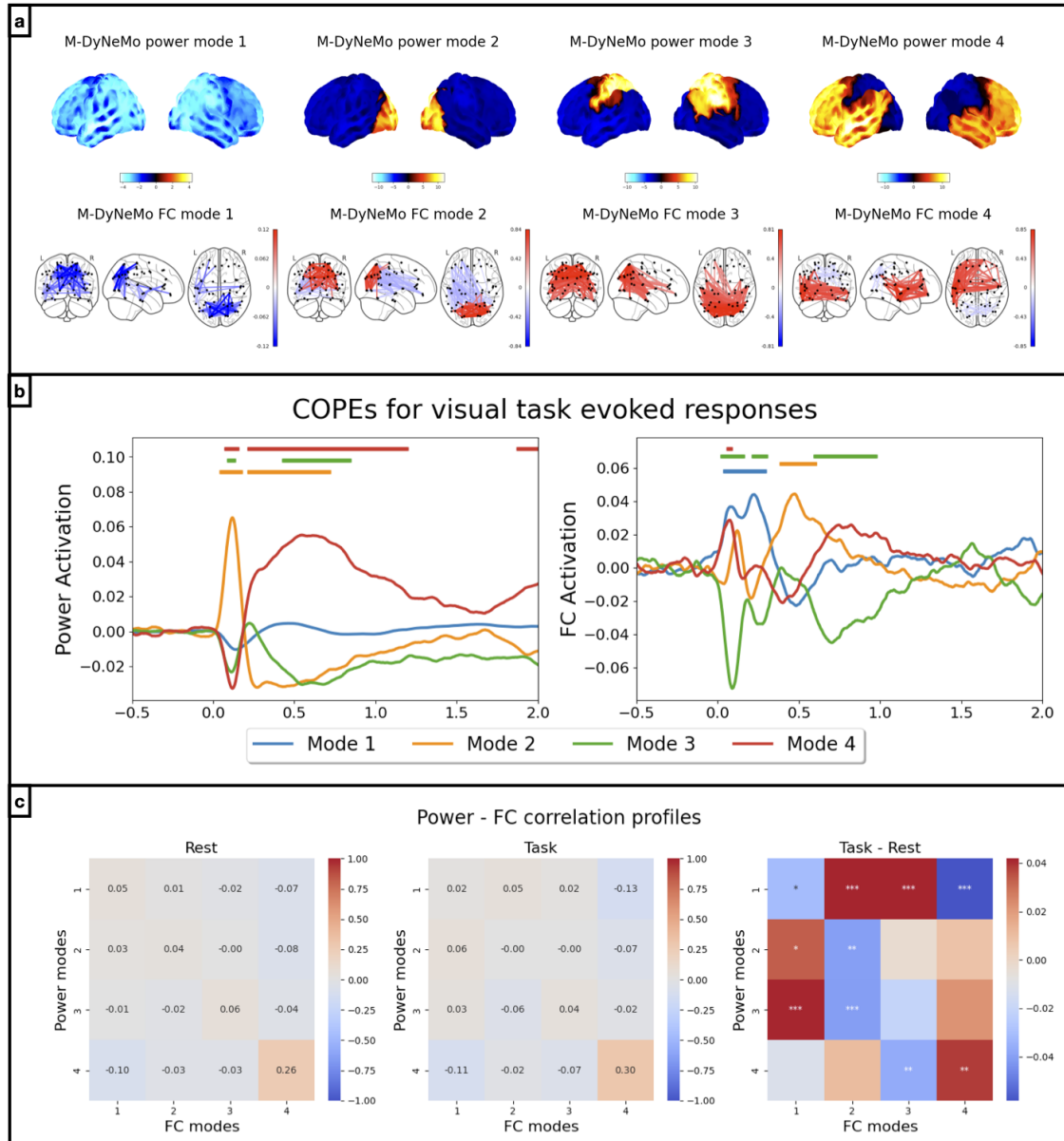


Figure 4.6: **Coupling between α and β is modulated by task stimuli.** Results from amplitude time courses (1-45Hz) of 52 brain regions obtained from MEGUK-52 data. a) Power (top row) and FC (bottom row) spatial modes are plotted. b) Evoked responses (baseline corrected) for re-normalised α (left) and β (right) to visual stimuli. Different colours show evoked responses for different modes. c) Correlations between α and β during rest (left) and during task (middle) are shown. The right plot illustrates difference between the correlation profiles where significant differences are marked with asterisks.

mode time courses, one for power and one for FC, possess very different characteristics and there is no significant correlation between them, i.e. they are decoupled. The same property can also be found in two other datasets (see Appendix Figure B.8). It should be noted that there is no constraint imposed by the generative model or inference framework in M-DyNeMo on the dynamics. It is solely the data itself that decides the dynamics are decoupled and the FC dynamics is more binarised than the power dynamics.

Using standard single-dynamic DyNeMo, consistent differences have previously been found in mode time courses between young and old participants ([Gohil et al., 2024a]). Therefore, a future direction is to see if the FC mode time courses, which to date will have been ignored due to its relative low signal in real data, can provide more understanding into aging, and distinctions between healthy and diseased brains.

Aside from resting-state data, we also explore the additional insights M-DyNeMo provides on task data. In Section 4.3.5, we show that the two mode time courses respond differently to visual stimuli. Moreover, we provide evidence that coupling between power and FC dynamics is modulated by stimuli. This could be related to the findings in [Barttfeld et al., 2015] that increased level of consciousness is associated with decreased complexity in FC dynamics, which could possibly be a result of increased coupling between the power and FC dynamics. Overall, our results demonstrate how M-DyNeMo has opened a new door to studying the relationship between cognition and TVFC.

In Section 4.3.3, we showed that standard, single-dynamic DyNeMo tends to ignore the FC dynamics in real data, and is dominated by the power dynamics. This is consistent with the idea that the signal from TVFC is weaker in real data than that from power, which is perhaps to be expected given that power is easier to estimate in general. If we look at the fluctuation of the reconstructed TV covariances from both M-DyNeMo and standard DyNeMo (Appendix Figure B.5), we can see that the shared-dynamics constraint of standard DyNeMo causes lower variability

in FC, which is consistent to the findings of applying multi-dynamic modelling to fMRI data ([Pervaiz et al., 2022]).

In Appendix B.1.3, we show that if the underlying data is generated by a single dynamic process, then M-DyNeMo does not enforce distinct dynamics for FC and power, and correctly infers a single, shared set of dynamics. It is worth noting that although data are simulated using the generative model of a Hidden Markov Model, where time courses are binarised, M-DyNeMo is still capable to learn from the data and infer binarised time courses, without a constraint being posed on its generative model. We have also explored using an edge time series technique proposed by [Faskowitz et al., 2020] on the simulated dataset. However, we find it gives noisy estimates of FC and fails to capture the transient temporal changes present in MEG data accurately.

In this chapter, we used amplitude envelope time courses (1-45Hz) from MEG data, and correspondingly used correlation between amplitude envelopes as a measure of FC. However, this ignores more detailed spectral information in the MEG signal. For example, the phase-locking between brain regions (coherence) has recently gained increasing interest as a measure of FC in electrophysiology data. [Vidaurre et al., 2016] introduced a method called *time-delayed embedding* for including spectral information into the covariance matrix of the time-delay embedded data. However, it is not trivial to decompose this covariance matrix into parts that correspond to power and parts that correspond to FC, similar to what was done in Section 4.6.4. The additional PCA step for reducing dimensionality complicates the modelling even further. An important aspect of future work is to find a way to allow inference of separate dynamics based on spectral information.

4.5 Conclusions

We proposed a new method for inferring potentially separate power and FC dynamics. We show that the proposed model is robust and the inferred networks are reproducible across subjects, datasets and parcellations. We use this to show that power and FC have similar spatial patterns, but distinct dynamics, in ongoing MEG brain activity. Furthermore, we show that the two types of dynamics respond differently to visual stimuli in MEG task data, and that the extent to which there is coupling between the two types of dynamics is modulated by task.

4.6 Methods

4.6.1 Dataset

Three real MEG datasets are used, including the Wakeman-Henson dataset (Section 1.5.2), MEGUK-38 (where only the eyes-open resting-state recordings were used) and MEGUK-52 (where both the eyes-open resting-state and visual task recordings were used) (Section 1.5.4).

Data preparation before training

Before training M-DyNeMo on the MEG datasets mentioned above, extra steps are taken to help with convergence of the model. The following steps are taken:

- Band-pass filter the data between 1 and 45 Hz. This is to concentrate the data in the frequency range where the neural activity is most prominent.
- Extract amplitude envelope signals (See Section 1.4.4 for details).
- Non-overlapping moving average of window size 25 samples (100ms at 250 Hz) is applied to smooth the amplitude envelope data.

- Standardise (z-transform) to ensure zero mean and unit variance for each channel. This is done independently for each subject.
- Orthogonalising the data with a full-rank PCA. This is done to help with the convergence of the model. See Section 4.7 for details.

4.6.2 Measure of power and FC

In this chapter, variance, instead of mean activation, of the amplitude of the recording signals is used as a measure of power. This is because we adopt the so-called *zero-mean model*, which has been widely used in previous literature ([Vidaurre et al., 2016, Vidaurre et al., 2018b, Gohil et al., 2022]). Correlation between amplitude signals is used as a measure of FC.

4.6.3 Sliding window approach for uncoupled dynamics

The aim of this method is to use a model free approach to partition the time dimension into a finite number of states according to either dynamic changes in power or FC. This heuristic approach is a combination of applying the sliding window approach and the K-means clustering. The steps include:

- Apply sliding window with window size of 500 samples (2s at 250Hz) and step size of 10 samples (40ms at 250Hz).
- For each window, calculate the correlation matrix and the standard deviations of the data.
- Apply K-means clustering algorithm to the TV correlation matrices and standard deviations separately.

Each time point is assigned to a cluster, which in our case can be thought of as a state, according to either TV power or TVFC. Hence we have formed two state time courses for power and FC separately. Notice in this chapter the number of clusters

K in the K-means clustering is chosen to be 4 to compare with M-DyNeMo where the number of modes is chosen to be 4.

4.6.4 Generative model of M-DyNeMo

In this section we formulate the generative model of M-DyNeMo. Similar to DyNeMo, it is assumed that the data is generated by a multivariate Gaussian distribution with zero mean. Let $\mathbf{x}_{1:T}$ be the observed data, where T is the number of samples/timestamps and $\mathbf{x}_t$ is a vector of length N_c - the number of channels. Then

$$\mathbf{x}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{C}_t) \quad \forall t \in [T] \quad (4.3)$$

independently, where $\mathbf{C}_t$ is the TV covariance matrix of the data at time t .

In M-DyNeMo, it is further assumed that $\mathbf{C}_t$ can be decomposed into

$$\mathbf{C}_t = \mathbf{G}_t \mathbf{F}_t \mathbf{G}_t, \quad (4.4)$$

where $\mathbf{G}_t$ is a diagonal matrix with strictly positive diagonal entries and $\mathbf{F}_t$ is a positive-definite matrix with ones on the diagonal. In particular, the diagonal elements of $\mathbf{G}_t$ are the TV standard deviations and the off diagonal elements of $\mathbf{F}_t$ are the TV correlations of the data at time t . The TV standard deviations and correlations are assumed to be generated by a linear mixture of a finite number of modes:

$$\begin{aligned} \mathbf{G}_t &= \sum_{j=1}^{J^1} \alpha_{jt} \mathbf{E}_j, \\ \mathbf{F}_t &= \sum_{j=1}^{J^2} \beta_{jt} \mathbf{R}_j. \end{aligned} \quad (4.5)$$

Here $\mathbf{E}_j$ is a diagonal matrix with positive diagonal entries, and $\mathbf{R}_j$ is a positive-definite matrix with ones on the diagonal. Notice here that the number of modes J^1, J^2 can be different. The mixing coefficients α_{jt} and β_{jt} are generated through

a softmax transformation of latent probabilistic variables $\boldsymbol{\theta}_{1:T}^1, \boldsymbol{\theta}_{1:T}^2$, respectively, whose prior distribution is parameterised by an RNN:

$$\begin{aligned}
 p(\boldsymbol{\theta}_{1:T}^{1,2}) &= \prod_{t=2}^T p(\boldsymbol{\theta}_t^{1,2} | \boldsymbol{\theta}_{1:t-1}^{1,2}) \\
 &= \prod_{t=2}^T \prod_{k=1}^2 p(\boldsymbol{\theta}_t^k | \boldsymbol{\theta}_{1:t-1}^{1,2}) \\
 &= \prod_{t=2}^T \prod_{k=1}^2 \mathcal{N}(\boldsymbol{\theta}_t^k | \boldsymbol{\mu}_{\boldsymbol{\theta}_t^k}(\boldsymbol{\theta}_{1:t-1}^{1,2}), \boldsymbol{\sigma}_{\boldsymbol{\theta}_t^k}^2(\boldsymbol{\theta}_{1:t-1}^{1,2}) \mathbf{I}_{J^k}),
 \end{aligned} \tag{4.6}$$

where

$$\begin{aligned}
 \boldsymbol{\mu}_{\boldsymbol{\theta}_t^k}(\boldsymbol{\theta}_{1:t-1}^{1,2}) &= g_{gen}^k(\text{LSTM}([\boldsymbol{\theta}_{1:t-1}^1, \boldsymbol{\theta}_{1:t-1}^2])), \\
 \boldsymbol{\sigma}_{\boldsymbol{\theta}_t^k}(\boldsymbol{\theta}_{1:t-1}^{1,2}) &= \rho(h_{gen}^k(\text{LSTM}([\boldsymbol{\theta}_{1:t-1}^1, \boldsymbol{\theta}_{1:t-1}^2]))), \quad k = 1, 2,
 \end{aligned} \tag{4.7}$$

and $[\boldsymbol{\theta}_{1:t-1}^1, \boldsymbol{\theta}_{1:t-1}^2]$ represents concatenating $\boldsymbol{\theta}_{1:t-1}^1, \boldsymbol{\theta}_{1:t-1}^2$ along the mode dimension. Here LSTM is a long short-term memory recurrently neural network, g_{gen}^k, h_{gen}^k are learnable linear transformations (dense layers), and ρ is the softplus activation.

With this generative model, each of the latent logits $\boldsymbol{\theta}_t^1, \boldsymbol{\theta}_t^2$ are generated by the history of both logits. This allows information to be communicated between the two different time courses in the generative model. An overview of the generative model is illustrated in Figure 4.7.

4.7 Orthogonalisation with PCA

Due to increased model complexity compared to DyNeMo, the stability of training the generative model above is sub-optimal, especially on relatively small datasets (around 50 subjects). We found in practice that orthogonolising the data, i.e. removing the correlation between channels, improved convergence of model training significantly. To this end, a full rank Principal Component Analysis (PCA) was applied to the data. Notice that the PCA-projected channels are linear combinations

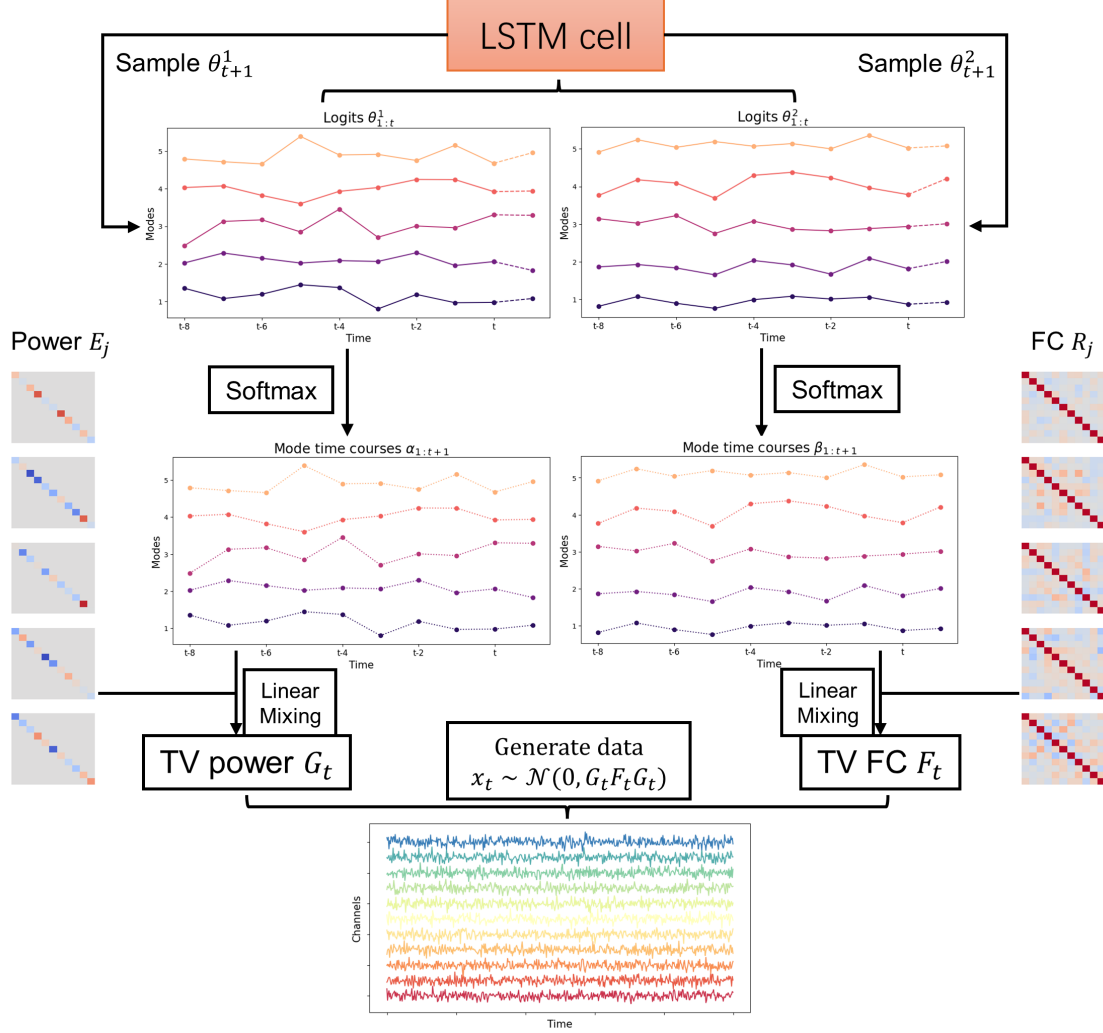


Figure 4.7: **Generative model of M-DyNeMo.** Historic logits $\theta_{1:t}^1$ and $\theta_{1:t}^2$ are concatenated and passed to the LSTM cell. The outputs of the LSTM cell are the parameters of the distribution from which we sample the future logits θ_{t+1}^1 and θ_{t+1}^2 . The sampled logits are concatenated to the original logit sequences and the process is repeated. The generated sequences of logits are normalised with a softmax transformation and we get the mode time courses $\alpha_{1:T}, \beta_{1:T}$. Diagonal matrices E_j and correlation matrices R_j are linearly mixed with $\alpha_{1:T}$ and $\beta_{1:T}$ respectively to give the time-varying power G_t and FC F_t . Finally the observed data are generated through the multivariate Gaussian distribution with time-varying covariance $G_t F_t G_t$.

of the original channels, which makes separating dynamics of power and FC in this projected space non-trivial. Hence we instead modify the generative model and incorporate the PCA projection matrix into the generative model specified above. This allows easy interpretation of the learnt observation model parameters.

Formally, let $\mathbf{W} \in \mathbb{R}^{N_c \times N_c}$ be the matrix with eigenvectors of the covariance matrix of the data on the rows, then the PCA-transformed data $\tilde{\mathbf{x}}_t = \mathbf{W}\mathbf{x}_t$ at time t has the following distribution, under the M-DyNeMo generative model:

$$\tilde{\mathbf{x}}_t \sim \mathcal{N}(\mathbf{0}, \tilde{\mathbf{C}}_t), \quad (4.8)$$

where $\tilde{\mathbf{C}}_t = \mathbf{W}\mathbf{C}_t\mathbf{W}^T$. Due to the fact that $\mathbf{W}$ is full-rank and hence $\mathbf{W}^T\mathbf{W} = \mathbf{I}_{N_c}$,

$$\begin{aligned} \tilde{\mathbf{C}}_t &= \mathbf{W}\mathbf{C}_t\mathbf{W}^T \\ &= \mathbf{W}\mathbf{G}_t\mathbf{F}_t\mathbf{G}_t\mathbf{W}^T \\ &= (\mathbf{W}\mathbf{G}_t\mathbf{W}^T)(\mathbf{W}\mathbf{F}_t\mathbf{W}^T)(\mathbf{W}\mathbf{G}_t\mathbf{W}^T) \\ &:= \tilde{\mathbf{G}}_t\tilde{\mathbf{F}}_t\tilde{\mathbf{G}}_t. \end{aligned} \quad (4.9)$$

Then the projection matrix $\mathbf{W}$ can be passed in into linear combination equations:

$$\begin{aligned} \tilde{\mathbf{G}}_t &= \mathbf{W}\mathbf{G}_t\mathbf{W}^T = \sum_{j=1}^{J^1} \alpha_{jt} \mathbf{W}\mathbf{E}_j\mathbf{W}^T, \\ \tilde{\mathbf{F}}_t &= \mathbf{W}\mathbf{F}_t\mathbf{W}^T = \sum_{j=1}^{J^2} \beta_{jt} \mathbf{W}\mathbf{R}_j\mathbf{W}^T. \end{aligned} \quad (4.10)$$

Now we have defined the generative model in the PCA-transformed space with the observation model parameters θ_{obs} in the original space.

4.8 Model training

The model parameters ϕ to estimate in M-DyNeMo include $\{\mathbf{E}_j : j \in [J^1]\}, \{\mathbf{R}_j : j \in [J^2]\}$, the weights of the LSTM, and of the affine transformations g_{gen}^k, h_{gen}^k , for $k = 1, 2$. Due to the highly intractable nature of the posterior distribution, variational inference is employed where we have used the mean field approximation to approximate the true posterior:

$$\begin{aligned}
 q(\boldsymbol{\theta}_{1:T}^{1,2}) &= \prod_{t=1}^T \prod_{k=1}^2 q(\boldsymbol{\theta}_t^k), \\
 &= \prod_{t=1}^T \prod_{k=1}^2 \mathcal{N}(\boldsymbol{\theta}_t^k | \mathbf{m}_{\boldsymbol{\theta}_t^k}, \mathbf{s}_{\boldsymbol{\theta}_t^k}^2 \mathbf{I}_{J^k}), \\
 \mathbf{m}_{\boldsymbol{\theta}_t^k} &= g_{inf}^k(\text{BiLSTM}(\tilde{\mathbf{x}}_{1:T_{seq}})), \\
 \mathbf{s}_{\boldsymbol{\theta}_t^k} &= \rho(h_{inf}^k(\text{BiLSTM}(\tilde{\mathbf{x}}_{1:T_{seq}}))),
 \end{aligned} \tag{4.11}$$

The learning of variational parameters are amortised by learning a map from the data to the variational parameters $\mathbf{m}_{\boldsymbol{\theta}_t^k}, \mathbf{s}_{\boldsymbol{\theta}_t^k}$ using bi-directional LSTM (BiLSTM) and learnable linear transformations $g_{inf}^k, h_{inf}^k, k = 1, 2$. Hence the encoder parameters ψ include the weights of BiLSTM, and of the affine transformations $g_{inf}^k, h_{inf}^k, k = 1, 2$.

Similar to training a DyNeMo, in practice, it is infeasible to feed the whole time series to the LSTM due to memory restrictions. Hence we separate the data into N_{seq} sequences of length $T_{seq} = 200$ (so that $\lfloor T/T_{seq} \rfloor = N_{seq}$). For M-DyNeMo, the loss (variational free energy) for each sequence n is

$$\mathcal{L}_n(\phi, \psi) = \underbrace{- \sum_{t=1}^{T_{seq}} \mathbb{E}_{q_{\psi}(\boldsymbol{\theta}_t^{1,2})} [\log p_{\phi}(\tilde{\mathbf{x}}_t | \boldsymbol{\theta}_t^{1,2})]}_{\text{LL}} + \underbrace{\sum_{t=2}^{T_{seq}} \sum_{k=1}^2 D_{KL}(q_{\psi}(\boldsymbol{\theta}_t^k) || p_{\phi}(\boldsymbol{\theta}_t^k | \boldsymbol{\theta}_{1:t-1}^{1,2}))}_{\text{KL}}, \tag{4.12}$$

and ADAM is used to minimise the loss with respect to ϕ and ψ .

Several tricks are used to help with the convergence of training and minimisation of the loss. See Appendix B.3.2 for more details.

Chapter 5

Modelling variability in functional brain networks using embeddings

5.1 Introduction

So far we have introduced different methods for inferring TVFC and they have provided numerous insights. However, all of these methods assume the same set of basis networks is shared by all recording sessions. This is an unrealistic assumption as functional brain activity is known to possess significant heterogeneity. Several key factors which contribute to this variability include:

- **Subject variability:** Physical structures of different individuals' brain can vary greatly (e.g. due to age, sex, disease states, etc), resulting in differences in recording signals.
- **Different scanners:** The number of sensors, sensor type, sensor positions and characteristics of the magnetically shielded room (MSR) can all contribute to differences in the resulting signal.
- **Scanning site:** Environmental factors, such as nearby power lines or variations in scanning protocols, can introduce additional differences.

While this variability poses challenges on data analysis and the derivation of normative group-level estimates, it provides a way for investigating the underlying cause of the variability. For example, we can study the effect of age or sex on the brain or even try to predict demographics from functional brain signals. This chapter delves into a framework that simultaneously yield consistent estimates at both the group level and the individual level.

It is possible to get session-specific estimates with post-hoc analyses, in particular, with a method called *dual estimation*, which has the similar rationale as dual regression in Independent Component Analysis (ICA, [Nickerson et al., 2017], [Beckmann et al., 2009]). However, this approach does not allow the inference of dynamic networks from the data to benefit from the subpopulation structure within the group. An approach to overcome this is to use hierarchical models that capture variability across sessions. For example, PROFUMO ([Harrison et al., 2015, Harrison et al., 2020]) has been successfully used to capture aspects of subject variability in functional modes in fMRI data. However, variability within a population is typically modelled using very simple parametric distributions (e.g. univariate Gaussian), which fails to capture the rich, multivariate variability expected across sessions.

In this chapter of the thesis, we employ embedding vectors, a technique widely used in the Natural Language Processing literature for characterising semantic relationships between words ([Mikolov et al., 2013]), to explicitly model variability in the basis set of networks. Embedding vectors have been previously used to deal with between-subject variability in supervised ([Csaky et al., 2023, Chehab et al., 2021]) and self-supervised learning ([Défossez et al., 2023, Jayalath et al., 2024]) contexts. Here we aim to incorporate it into an unsupervised generative model of dynamic network activity. Specifically, we incorporate embedding vectors and hierarchical Bayesian Modelling ([Kruschke and Vanpaemel, 2015]) into the generative model of the HMM (Section 3.1), which has been successfully applied to both fMRI and MEG

data ([Baker et al., 2014, Vidaurre et al., 2016, Vidaurre et al., 2018b, Vidaurre et al., 2017, Higgins et al., 2021, Quinn et al., 2018]), and we call this extension **HIVE** (HMM with Integrated Variability Estimation)

In this chapter, with simulated and real MEG datasets, we show that our proposed method can infer the underlying population structure in an unsupervised manner. This helps to give a more accurate description of the session-specific network estimates than traditional method based on the group-level model. Source code for HIVE is available in the *osl-dynamics* toolbox and scripts to reproduce results in this paper are available here:

github.com/OHBA-analysis/Huang2024_ModelVariabilityWithEmbeddings.

5.2 Dual estimation in the traditional HMM approach

In the HMM, the state means and covariances μ_j, Σ_j are group-level estimates. Similar to dual regression in ICA, dual estimation ([Vidaurre et al., 2021]) is a way to compute session-specific means and covariances in the HMM. For each session, we fix the state probabilities from the group-level HMM and re-estimate the observation model parameters with the session’s data. We use **HMM-DE** (HMM with dual estimation) to refer to the pipeline of training a group-level HMM followed by dual estimation for estimating session-specific parameters. See Appendix C.1 for derivations.

5.3 The use of embedding vectors

The use of word embeddings is a common practice in Natural language processing (NLP, [Mikolov et al., 2013]). A word embedding is a real-valued vector that encodes the meaning of the word such that words closer in the vector space have similar

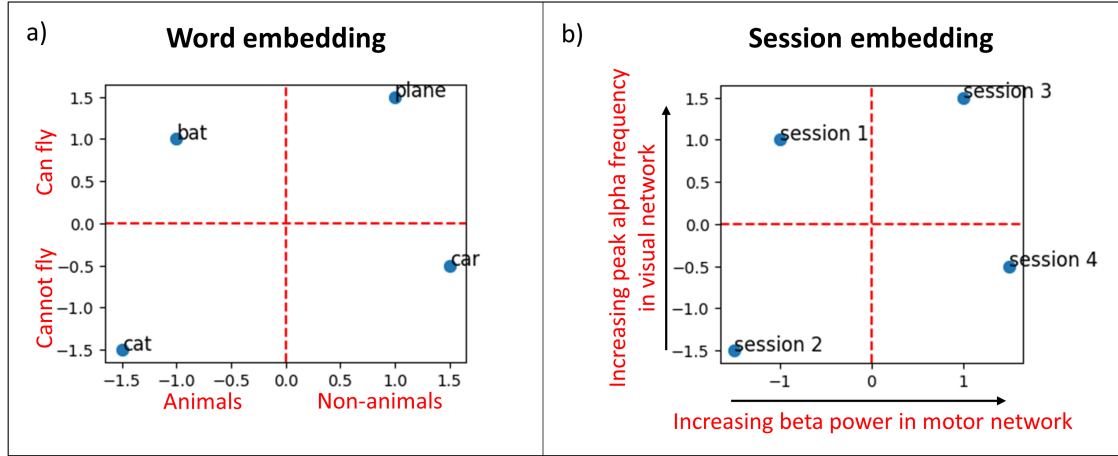


Figure 5.1: **Examples of embedding vectors.** a) Example of word embedding vectors. The x-axis encodes whether a word is for animals or non-animals and the y-axis encodes whether the object can fly or not. b) Illustration of possible variability captured by embedding vectors. The x-axis encodes the direction of increasing beta power in motor network whereas the y-axis encodes the direction of increasing peak alpha frequency in visual network.

meanings. In Figure 5.1a, we show an example of word embeddings. Here the x-axis encodes whether a word is for animals or non-animals and the y-axis encodes whether the object can fly or not. Similarly, we can assign a vector for each recording session and hope that sessions with similar properties are close to each other in the vector space. In Figure 5.1b, we show an illustration of possible variability captured by embedding vectors. Here the x-axis encodes the direction of increasing beta power in motor network whereas the y-axis encodes the direction of increasing peak alpha frequency in visual network.

5.4 Generating session-specific covariances

In this section we outline the process of modelling session-specific covariances. For simplicity, we will assume that the state-specific means are fixed to zero (i.e. $\mu_j = 0, \forall j \in [J]$), but similar modelling can be easily applied to the means. We still

assume that our data is generated from a multivariate Gaussian distribution with

$$\mathbf{x}_t^i | (s_t^i = j) \sim \mathcal{N}(0, \mathbf{\Sigma}_j^i), \quad \forall t \in [T], i \in [N], \quad (5.1)$$

where i is the session index and N is the number of sessions. We further assume that the transition probability matrices for all sessions are the same, i.e. $\mathbf{A}^i = \mathbf{A} \quad \forall i \in [N]$. Importantly, we assume each session i has its own set of covariances $\{\mathbf{\Sigma}_j^i : j \in [J]\}$ and we want the covariances for different sessions to respect (be regularised) by the relationships between sessions in the population. Furthermore, we want the session-specific covariances to vary around the group-level covariances. This is done using our novel variability encoding block described in Figure 5.2.

Given a session i , we want to generate the session-specific covariances $\mathbf{\Sigma}_j^i$. Let $\boldsymbol{\omega}^i$ be the session embedding vector of session i , which is a vector of pre-defined length n_ω that provides a “fingerprint” for this session. Next, let $\mathbf{\Sigma}_j$ be the group-level covariance matrix for state j and to ensure legitimacy of generated session-specific covariances, we work in the Cholesky space. Let $\mathbf{l}_j, \mathbf{l}_j^i$ be the flattened vectors of the lower triangular entries of the Cholesky factors $\mathbf{L}_j, \mathbf{L}_j^i$ of $\mathbf{\Sigma}_j, \mathbf{\Sigma}_j^i$ respectively, then the session-specific Cholesky vector $\mathbf{l}_j^i$ is generated by adding a deviation to the group-level Cholesky vector $\mathbf{l}_j$ with

$$\mathbf{l}_j^i = \mathbf{l}_j + d_j^i \mathbf{e}_j^i, \quad (5.2)$$

where d_j^i is a scalar representing the spatially global magnitude of the deviation and $\mathbf{e}_j^i$ is a vector representing the standardised pattern of the spatial deviation, in the sense that it has unit sample standard deviation. We would expect the deviation magnitude d_j^i to be small so that the session-specific covariances vary around the group-level covariances, while enforcing positivity of this quantity. Hence we put an exponential prior on d_j^i :

$$d_j^i \sim \text{Exp}(\lambda_j^i). \quad (5.3)$$

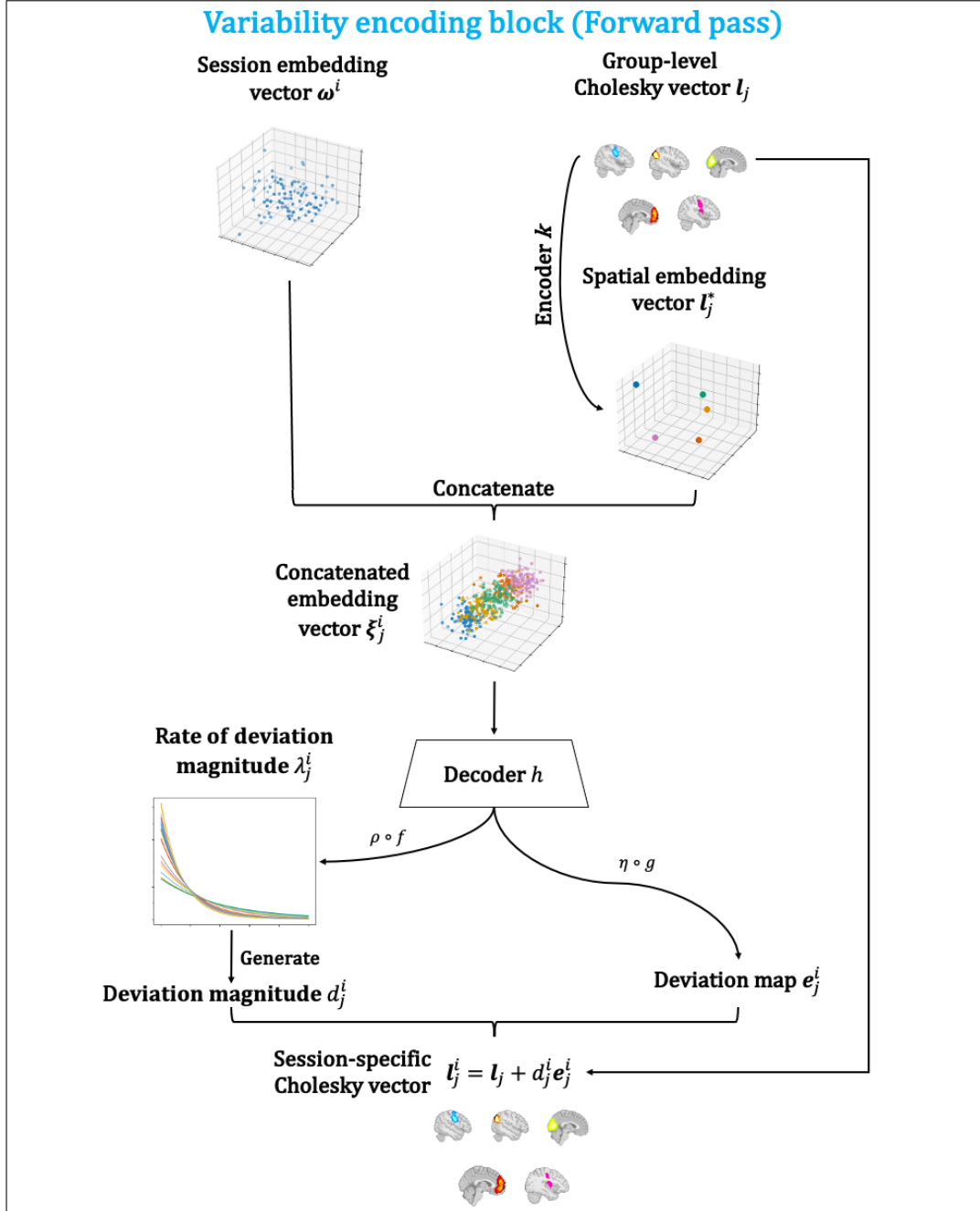


Figure 5.2: **Forward pass of the variability encoding block.** Session-specific information from the session embedding vectors ω^i and state-specific information from the group-level Cholesky vectors l_j are combined to generate session and state-specific Cholesky vectors l_j^i . Here the **encoder k** is an affine transformation that condenses state-specific information to a lower dimensional space. The **decoder h** is a multi-layer perceptron (MLP) that decodes state and session-specific deviations from the concatenated embedding vectors ξ_j^i . Deviation magnitude d_j^i is generated by an exponential distribution with rate λ_j^i , which is generated by applying an affine transformation f followed by a softmax ρ to the output of h . The deviation map e_j^i is generated by applying an affine transformation g and normalisation η to the output of h . Finally we get the session-specific Cholesky vector l_j^i with $l_j^i = l_j + d_j^i e_j^i$.

The rate parameter λ_j^i and deviation pattern $\mathbf{e}_j^i$ are generated through

$$\begin{aligned}\lambda_j^i &= \rho(f(h(\boldsymbol{\xi}_j^i))), \\ \mathbf{e}_j^i &= \eta(g(h(\boldsymbol{\xi}_j^i))),\end{aligned}\tag{5.4}$$

where ρ is the softplus function to ensure positivity of λ_j^i and η is a Layer Normalisation layer ([Ba et al., 2016]) with non-trainable scale parameter of 1 to ensure unit standard deviation of $\mathbf{e}_j^i$. Affine functions f, g are learnable transformations that extract different information from the hidden state given by the decoder h , a multi-layer perceptron (MLP, [Popescu et al., 2009]). $\boldsymbol{\xi}_j^i$ is the key object called a **concatenated embedding vector** that encodes both session and state-specific information. It is formed by concatenating the session embedding vector $\boldsymbol{\omega}^i$ and spatial embedding vector $\mathbf{l}_j^*$, which is a lower dimensional representation of the Cholesky vectors:

$$\boldsymbol{\xi}_j^i = [\boldsymbol{\omega}^i, \mathbf{l}_j^*], \quad \mathbf{l}_j^* = k(\mathbf{l}_j),\tag{5.5}$$

where k is an affine transformation that serves as an encoder to encode spatial information and $[\boldsymbol{\omega}^i, \mathbf{l}_j^*]$ concatenates the two vectors. The complete generative process is summarised in Figure 5.3.

5.5 Training HIVE

The model parameters of HIVE, ϕ include

- the transition probability matrix $\mathbf{A}$,
- the embedding vectors $\boldsymbol{\omega} = \{\boldsymbol{\omega}^i : i \in [N]\}$,
- the group-level Cholesky vectors $\mathbf{l} = \{\mathbf{l}_j : j \in [J]\}$,
- the weights of the encoder k , the decoder h , the layer normalisation layer η and the affine transformations f, g .

HIVE Generative Model

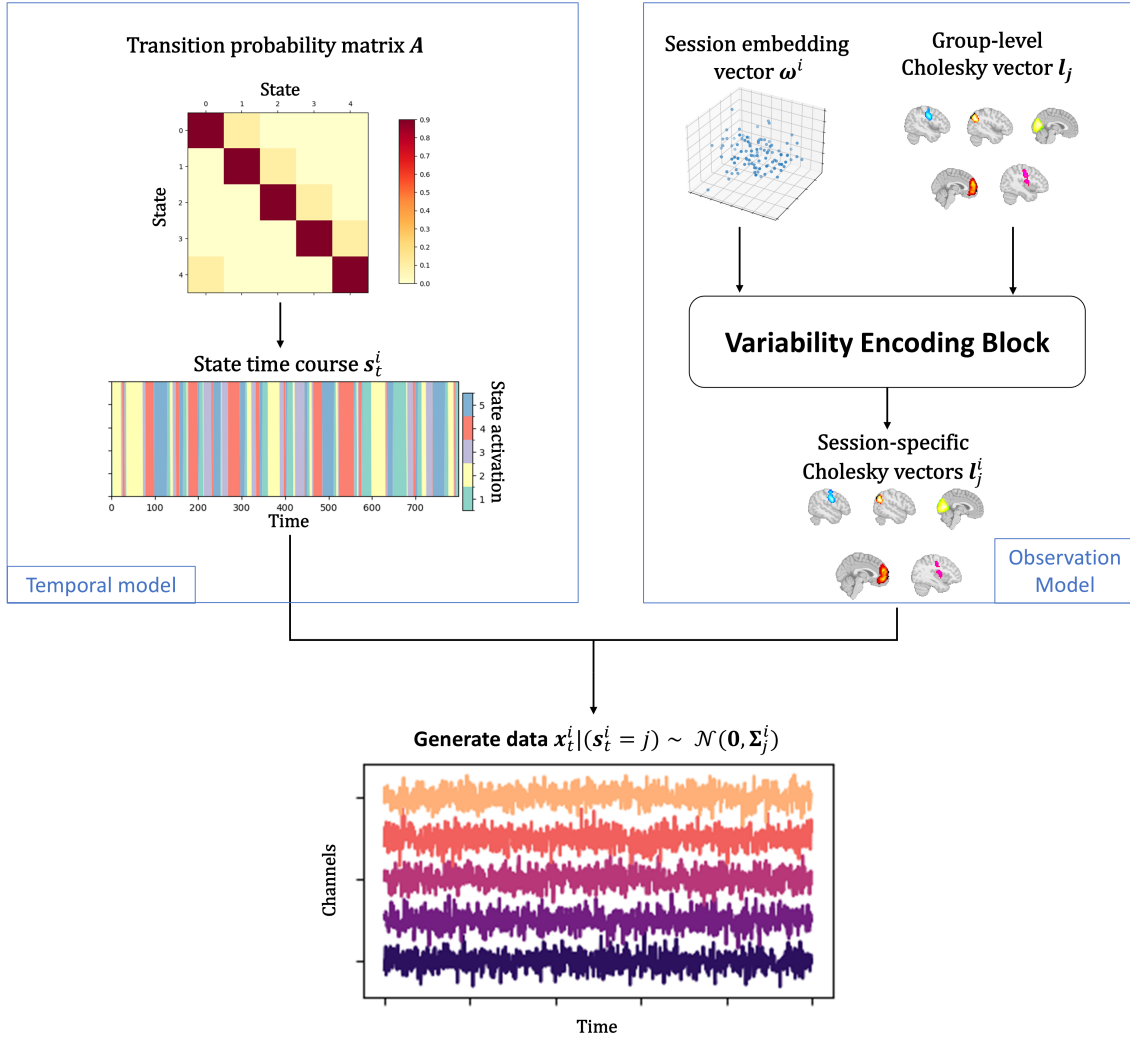


Figure 5.3: **Generative model of HIVE.** State time courses are generated from the group-level transition probability matrix A through a Markov process. At the same time, session embedding vectors ω^i and group-level Cholesky vectors l_j are passed to the variability encoding block to give session-specific Cholesky vectors l_j^i . Given the state s_t^i at each time, the observed data is generated through the multivariate Gaussian likelihood.

To perform inference on these parameters, we employ the variational EM algorithm (Section 2.1.6, [Dempster et al., 1977]). During the “E-step”, we update the state probabilities (the posterior distribution of the states) with the forward-backward algorithm of the Baum-Welch algorithm, similar to the HMM (Section 3.1.2). During the “M-step”, given the state probabilities, we again update the transition probability matrix $\mathbf{A}$ with the stochastic update technique used in [Vidaurre et al., 2018a] (see Section 3.1.2). Notice we do not have access to the exact posterior distribution of the deviation magnitudes $p(\mathbf{d}|\mathbf{x})$ and we approximate it with a mean field approximation (See section 2.1.6):

$$q(\mathbf{d}) = \prod_{i=1}^N \prod_{j=1}^J q(d_j^i) = \prod_{i=1}^N \prod_{j=1}^J \Gamma(d_j^i | a_j^i, b_j^i). \quad (5.6)$$

Here the shape a_j^i and the rate b_j^i parameters of the Gamma distributions Γ are learnable parameters. The reason that Gamma distributions are chosen is that there is an analytic solution for the KL divergence between a Gamma distribution and an Exponential distribution. As a result, the variational parameters $\boldsymbol{\psi} = \{a_j^i, b_j^i : i \in [N], j \in [J]\}$. All other parameters in $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$ are optimised by minimising the variational free energy given the state probabilities.

5.5.1 Free energy given the state probabilities

Similar to training DyNeMo and M-DyNeMo, data is separated into sequences for scalability. We ensure that data in each sequence n belong to the same session i_n

and the loss for this sequence is

$$\begin{aligned}
\mathcal{L}_n(\phi, \psi) &= -\mathbb{E}_{q(\mathbf{s}^{in})} \mathbb{E}_{q_\psi(\mathbf{d}^{in})} [\log p_\phi(\mathbf{x}^{in} | \mathbf{s}^{in}, \mathbf{d}^{in})] + \frac{1}{N_{seq}} D_{KL}(q_\psi(\mathbf{d}) || p_\phi(\mathbf{d})) \\
&= -\sum_{t=1}^{T_{seq}} \mathbb{E}_{q(s_t^{in})} \mathbb{E}_{q_\psi(\mathbf{d}^{in})} [\log p_\phi(\mathbf{x}_t^{in} | s_t^{in}, \mathbf{d}^{in})] + \frac{1}{N_{seq}} D_{KL}(q_\psi(\mathbf{d}) || p_\phi(\mathbf{d})) \\
&= -\underbrace{\sum_{t=1}^{T_{seq}} \sum_{j=1}^J \mathbb{P}(s_t^{in} = j | \mathbf{x}^{in}) \mathbb{E}_{q_\psi(\mathbf{d}^{in})} [\log \mathcal{N}(\mathbf{x}_t^{in} | 0, \Sigma_j^{in}(\phi, \mathbf{d}_j^{in}(\phi)))]}_{\text{LL}} \\
&\quad + \underbrace{\frac{1}{N_{seq}} \sum_{i=1}^N \sum_{j=1}^J D_{KL}(q_\psi(d_j^i) || p_\phi(d_j^i))}_{\text{KL}},
\end{aligned} \tag{5.7}$$

where the $\frac{1}{N_{seq}}$ factor is due to the fact that the loss is averaged over sequences in a batch.

In summary, for a randomly drawn mini-batch $B \subseteq [N_{seq}]$, the training iterates between getting the posterior probabilities of the hidden states $q(\mathbf{s})$ and one step of gradient descent with the loss function

$$\mathcal{L}(\phi, \psi) = \frac{1}{|B|} \sum_{n \in B} \mathcal{L}_n(\phi, \psi), \tag{5.8}$$

where the reparameterisation trick is used to pass gradient through to the variational parameters ψ . See Figure 5.4 for an overview of the forward pass for training HIVE.

5.5.2 Choosing the embedding dimension

The length n_ω of the embedding vectors ω^i is a hyperparameter that needs to be pre-defined. Given a fixed decoder h , a smaller n_ω will lead to a more parsimonious model but there could be a risk of over-regularisation and underfitting. On the other hand, a larger n_ω will lead to a more flexible model but could lead to overfitting. In practice, we use the loss function and follow the following procedure. Firstly, we

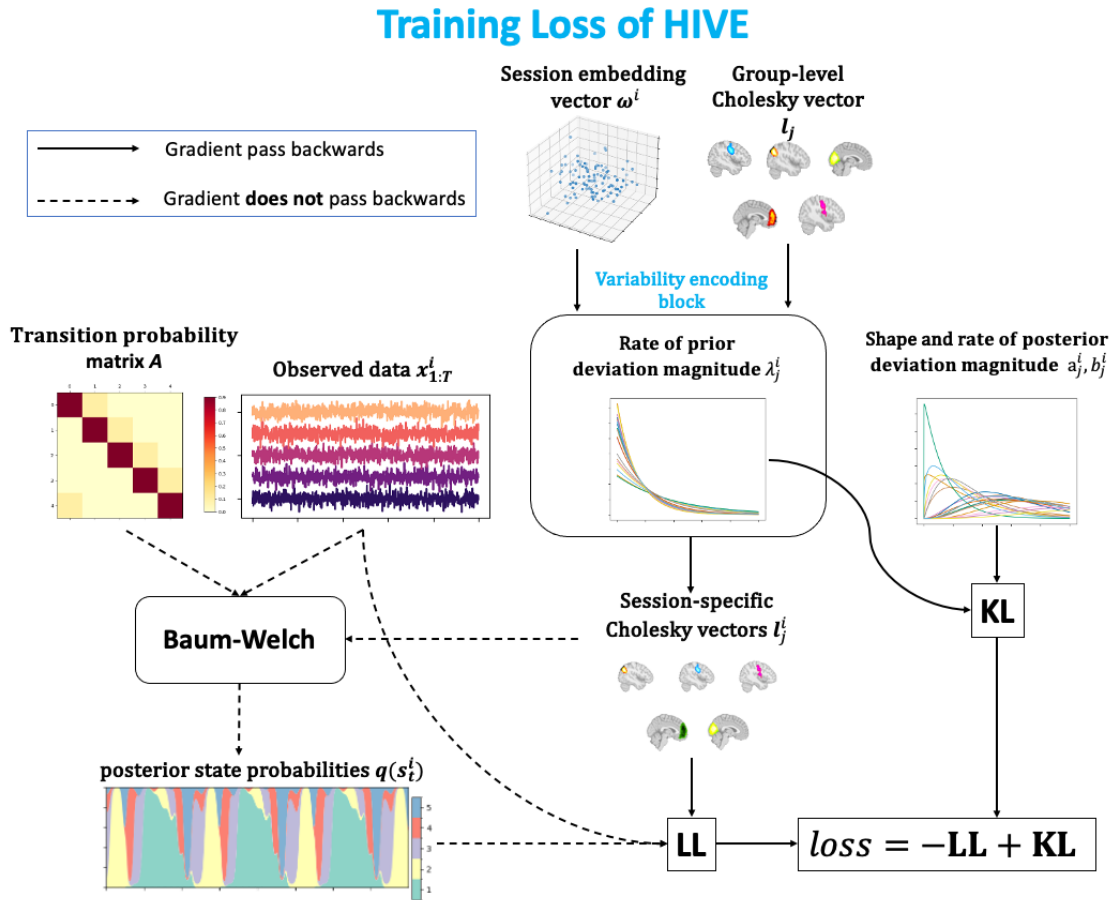


Figure 5.4: **The full HIVE training framework.** The black solid arrows in the plot show connections where gradient flows through during back-propagation and the dashed black arrows show connections where gradient does not flow through.

define a set of candidate values for n_ω , say $\{n_\omega^1, \dots, n_\omega^M\}$ where $n_\omega^k < n_\omega^{k+1}$, $\forall k \in [M - 1]$. Then we train the model with each of these candidates of embedding dimension a number of times independently and for each run, we get the training loss (variational free energy on the training data) of the model. Starting from $k = 1$, we test whether the training loss of the model with embedding dimension n_ω^{k+1} is significantly lower than that with embedding dimension n_ω^k . If it is, we continue to test for n_ω^{k+1} and n_ω^{k+2} . If it is not, we choose n_ω^k as the embedding dimension. The test for significance can be done with a one sided t-test or a non-parametric permutation test. Formally, the procedure is summarised in Algorithm 2.

Algorithm 2: Choosing the embedding dimension

Input: Embedding dimension candidates $\{n_\omega^1, \dots, n_\omega^M\}$

Output: Chosen embedding dimension n_ω^*

Procedure:

for $k \in [M - 1]$ **do**

 Test the null hypothesis that the training loss of model with n_ω^k is not significantly lower than that with n_ω^{k+1}

if *Null hypothesis is not rejected* **then**

return n_ω^k

return n_ω^M

5.5.3 Initialisation of variational parameters

In practice, we found that the a good initialisation of the variational parameters of the Gamma distributions $\{a_j^i, b_j^i : j \in [J], i \in [N]\}$ is crucial for the training of HIVE, without which the model can either get stuck in a local minimum or diverge. Here we initialise the shape and rate parameters of the Gamma distributions as

follows:

$$\begin{aligned}
d_{mean} &= \frac{1}{N} \sum_{i=1}^N |l_{static}^i - \overline{l_{static}^i}| \\
d_{var} &= \frac{1}{5} d_{mean} \\
a_j^i &= \frac{d_{mean}^2}{d_{var}} \quad \forall i \in [N], j \in [J] \\
b_j^i &= \frac{d_{mean}}{d_{var}} \quad \forall i \in [N], j \in [J].
\end{aligned} \tag{5.9}$$

The idea is to estimate the mean deviation magnitude d_{mean} with the deviation of the Cholesky vectors $\{l_{static}^i\}_{i=1}^N$ of the static covariance matrices from different subjects and get the parameters so that the resulting Gamma distribution has mean d_{mean} and variance $d_{var} = \frac{1}{5} d_{mean}$, which is what we found to be a good initialisation in practice.

5.5.4 Annealing during training

Similar to the annealing technique discussed in Section 3.2.3, we anneal the contribution from the KL divergence term in the loss function during training. Additionally, we anneal the sampling process of the variational distribution of deviation magnitudes $q(\mathbf{d})$ when applying the reparameterisation trick:

$$\hat{\mathbf{d}} = (1 - \kappa) \bar{\mathbf{d}} + \kappa \mathbf{d}^*, \tag{5.10}$$

where $\bar{\mathbf{d}}$ is the expectation of $q(\mathbf{d})$, $\mathbf{d}^*$ is a sample from $q(\mathbf{d})$ and κ is the annealing parameter that starts from 0 and increases to 1 during training. We find this significantly improves the convergence of the model and we believe this serves as an exploration-exploitation mechanism, which is extensively studied in the fields of reinforcement learning ([Wang et al., 2018]) and Bayesian Optimisation ([Jalali et al., 2012, Shahriari et al., 2015]). At the beginning of the training, only the expectation is used and the gradient will intend to push the mean deviations to the correct position. This is the exploration phase. As training progresses, we gradually take into

account the uncertainty in the latent variables, i.e. the variance and higher moments of the variational distribution, which allows the training to fine tune the variational parameters. This is the exploitation phase.

5.6 Datasets

5.6.1 Simulated datasets

In the simulation studies, we simulate data with an HMM generative model (See A.1, A.2). The simulated data for the 3 simulation studies are described below.

- **Simulation study 1.** We simulate data with $N = 10$, $T_i = 25,600, \forall i \in [N]$, $N_c = 11$, $J = 5$. Here the state covariance matrices of sessions 1 and 2 are altered and all other sessions have the same unaltered group-level covariance matrices - session 1 has increased variance in channel 1 and decreased variance in channel 2 whilst session 2 has decreased variance in channel 1 and increased variance in channel 2. For both sessions 1 and 2, the increased variances are 5 times and the decreased variances are 20% of the group-level variances.
- **Simulation study 2.** Here $N = 100$, $T_i = 3000, \forall i \in [N]$, $N_c = 40$, $J = 5$. Sessions are assigned into 3 groups and an embedding vector for each session is simulated according to the session's group assignment. Session-specific covariances are simulated based on the simulated embedding vectors. (See A.4 for details).
- **Simulation study 3.** In this study 4 datasets are simulated with $N \in \{5, 10, 50, 100\}$ and $T_i = 3000, \forall i \in [N]$, $N_c = 40$, $J = 5$. Session-specific covariances are simulated in the same way as in Simulation study 2.

5.6.2 Real MEG datasets

Three real MEG datasets are used:

- The MEGUK-38 dataset (Section 1.5.4). Only the eyes-closed resting-state recordings (64 subjects) are used. This subset of recordings is referred to as the **Nottingham** dataset in this chapter.
- The Wakeman-Henson dataset (Section 1.5.2).
- The Cam-CAN dataset (Section 1.5.3).

Data preparation before training

Before training HIVE on these real MEG data, extra steps are taken to help the model capture spectral information in the data, as well as to speed up training and convergence. The following steps are used:

- Data are time delay embedded with ± 7 lags (See Section 1.4.5).
- Data are standardised across time for each session. This is done so that the next PCA step is not confused by potential different scaling in different channels.
- Data are projected by PCA with 80 components. This is done to reduce the dimensionality of the data as it is impossible to train the model on the original time delay embedded data even on modern hardware.
- Data are standardised again to help with convergence of the model.

The combined dataset

In one of the studies, HIVE is trained on a combined dataset that has data from both the Nottingham and the Cam-CAN dataset. In this combined dataset, age and sex profiles of the subjects are matched in the sense that in each of the age groups and sex, there is the same number of subjects from each of the datasets. There are 128 subjects in total, with 64 subjects from each of the Nottingham and Cam-CAN datasets. Subjects are scanned with their eyes closed in this combined dataset.

5.7 Simulation results

Here we show the results of training HIVE on the 3 simulated datasets described in Section 5.6.1. We also compare these results with those from HMM-DE (See Section 5.2). For model specifications see Appendix C.4.

5.7.1 Simulation study 1

Variability encoding block learns multivariate session-specific covariance deviations

We aim to test if patterns of deviations across multiple channels can be learnt by the generative model, in particular, the variability encoding block, during inference. This is an important feature in the sense that we want the generative model to act as a prior that regularises how each session can deviate from the group. Firstly, we can see from Figure 5.5a that both HMM-DE and HIVE can infer the group-level covariances correctly.

What HIVE offers in addition is shown in Figure 5.5b where the inferred session embedding vectors are plotted. We can clearly see that the embedding vectors for sessions 1 and 2 are far away from the cluster formed by other sessions which have unaltered covariances, which matches well with the ground truth. The model’s capability to encode deviation pattern across multiple channels in the generative model is illustrated in Figure 5.5c, in which the simulated patterns of deviation for sessions 1 and 2 can be generated from the trained variability encoding block. Although HMM-DE can infer session-specific covariances via dual estimation, it cannot generate data with patterns that deviate from the group mean. This is because intrinsically HMM-DE is a group-level model.

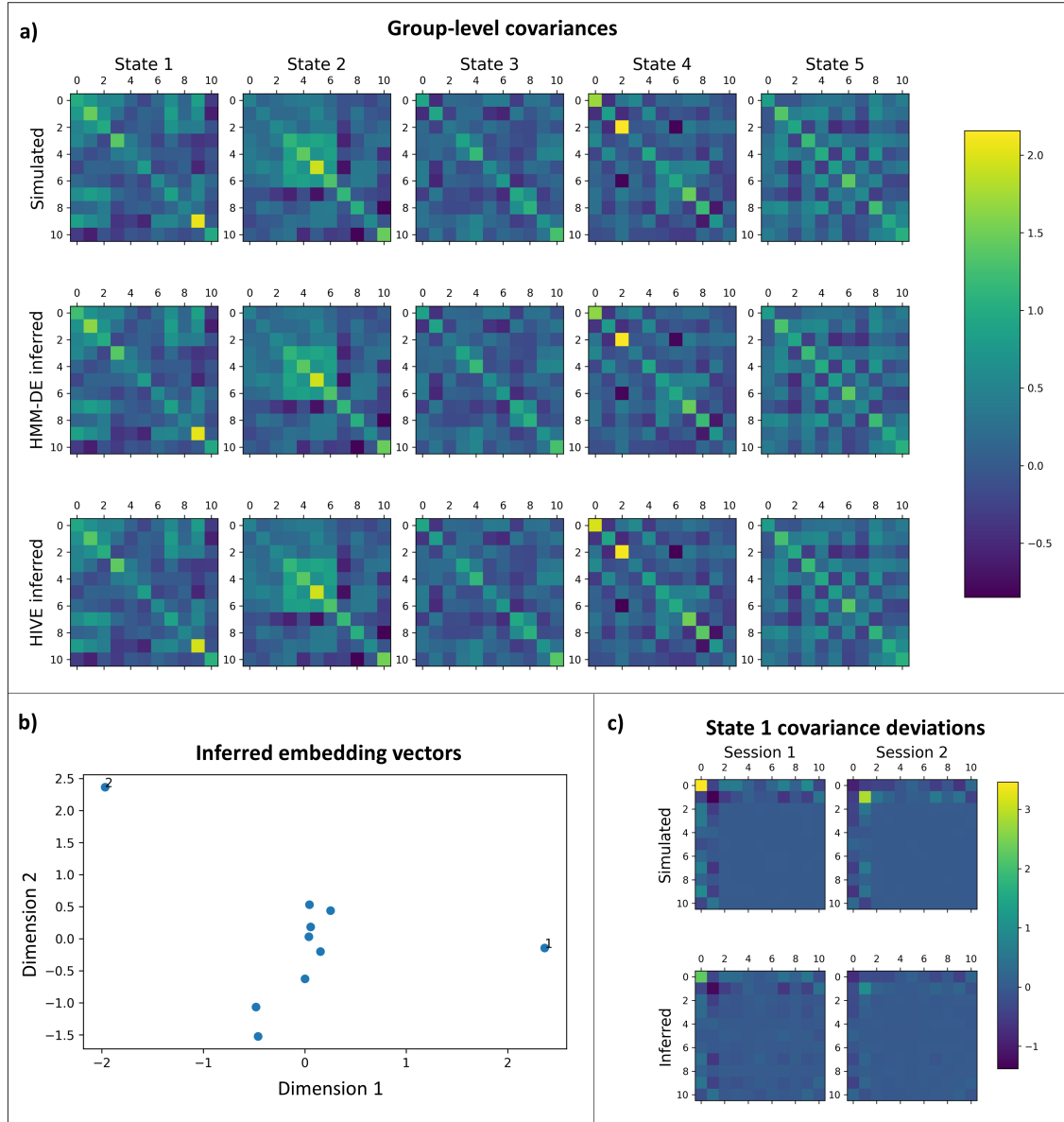


Figure 5.5: Variability encoding block learns session-specific covariance deviations. Results obtained using *Simulation study 1*. a) Simulated (top row), HMM-DE inferred (middle row) and HIVE inferred (bottom row) group-level covariances. The columns show covariances of different states. b) Inferred session embedding vectors. Sessions 1 and 2 are annotated. c) Simulated deviations (top row) and deviations from trained generative model of HIVE (bottom row) for state 1 for session 1 (left column) and for session 2 (right column).

5.7.2 Simulation study 2

Underlying subpopulation structure is inferred by HIVE

In this simulation study, we aim to show that HIVE can recover the ground truth underlying subpopulation structure (simulated embedding vectors of sessions). This is demonstrated in Figure 5.6a, where the ground truth grouping of sessions is recovered by the inferred session embedding vectors. In particular, sessions which are close together in the simulated space (e.g. sessions 7 and 45) stay close in the inferred space. Figure 5.6b shows both HMM-DE and HIVE can infer the state time courses perfectly. Furthermore, we can see from Figure 5.6c that both approaches can recover the pairwise session relationship between session-specific covariances, though HMM-DE overestimates the pairwise distance due to noise added to the data, whereas HIVE has the preferable behaviour of underestimating the pairwise distance due to the regularising effect of the prior on the deviation magnitude. This prior makes the inferred session-specific covariances more similar to the group average, when there is insufficient evidence available in the data to do otherwise.

5.7.3 Simulation study 3

HIVE performs more accurate inference than HMM-DE

Now we focus on comparing HIVE and HMM-DE in terms of accuracy of inferred session-specific covariances. Both HIVE and HMM-DE are trained 10 times to get a more accurate and stable estimation of the accuracy. From Figure 5.7 we see HIVE can infer more accurate session-specific covariances than HMM-DE. Crucially, with the number of samples per session being kept the same, the accuracy for HIVE improves with more sessions in the dataset while that for HMM-DE does not have a significant improvement. This shows the advantage of the variability encoding block in making use of data of heterogeneous sessions to help infer on every single session.

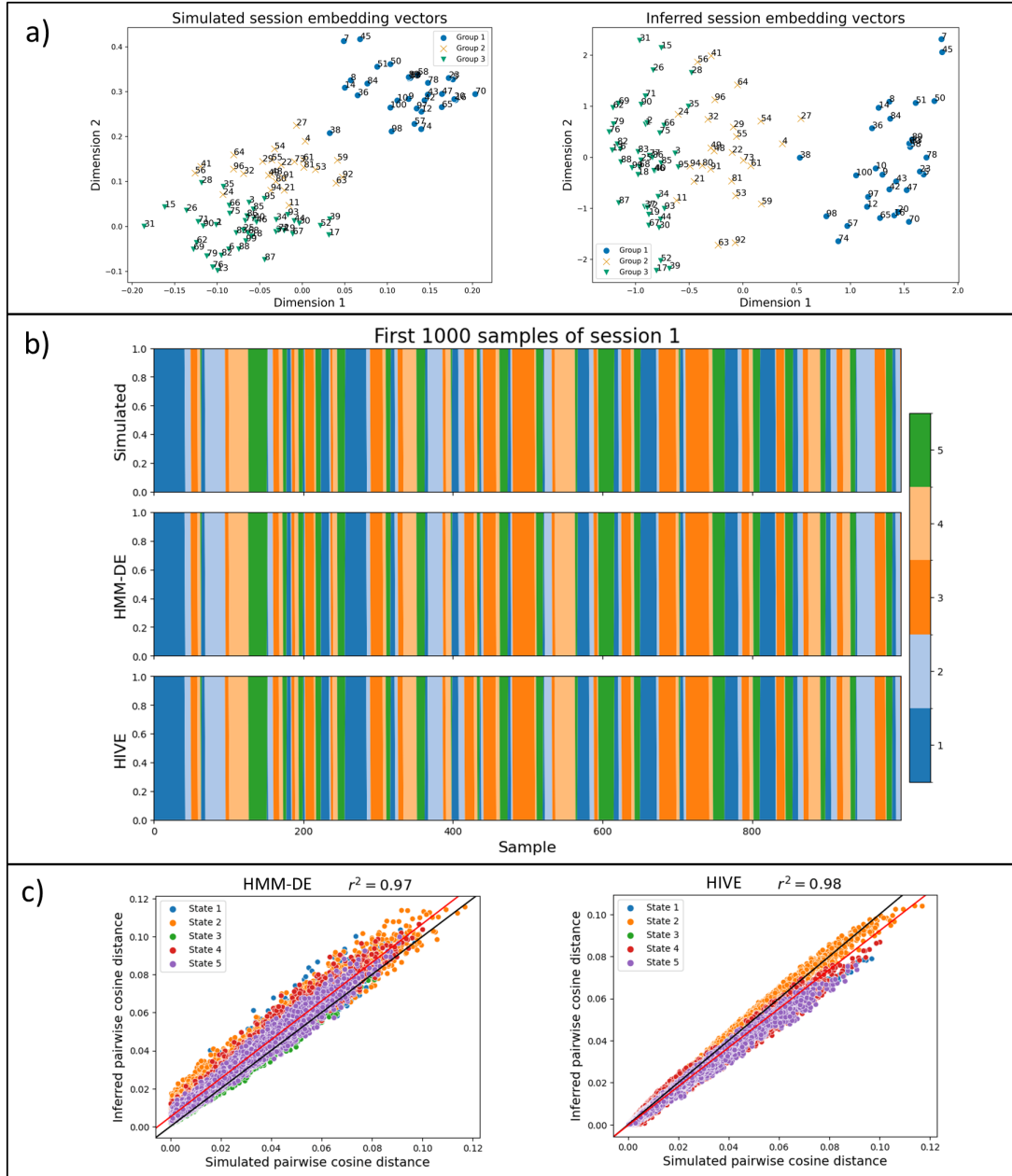


Figure 5.6: **HIVE infers the underlying subpopulation structure.** *Results obtained using Simulation study 2.* a) Simulated (left) and inferred (right) session embedding vectors. Each point is marked and coloured by the ground truth group assignment, and is annotated with the session number. b) Simulated (top), HMM-DE inferred (middle) and HIVE inferred (bottom) state time courses. c) Session-pairwise cosine distance of simulated (x-axis) against inferred (y-axis) covariances from HMM-DE (left) and HIVE (right). The black line shows the $y = x$ line, which corresponds to optimal performance, and the red line is a fitted line through the points, with the coefficient of determination r^2 displayed in the title.

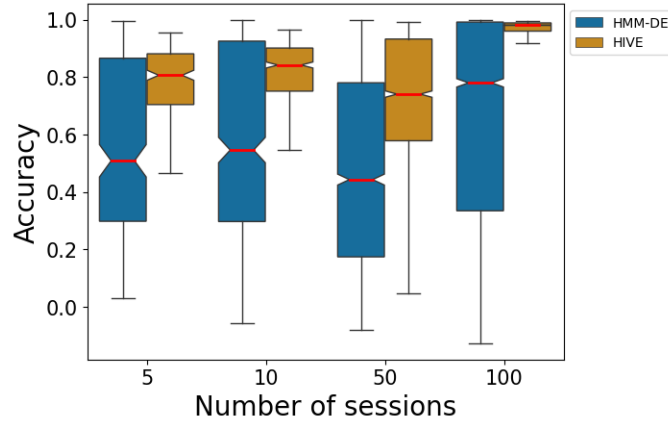


Figure 5.7: **HIVE performs more accurate inference than HMM-DE.** *Results obtained using Simulation study 3.* Accuracy of session-specific covariances (y-axis) are plotted against the number of sessions in the dataset (x-axis) for HMM-DE (blue) and HIVE (orange). Here the number of samples per session is always the same. Correlation between inferred and simulated session-specific covariances is used as a metric for accuracy.

5.8 Real data results

For model specifications see Appendix C.4.

5.8.1 Wakeman-Henson dataset

HIVE reveals similarities and differences between MEG recordings

In this section, we study the Wakeman-Henson dataset (see Section 5.6.2). This dataset contains 6 sessions for each subject. Ideally, we would expect the sessions for a subject to be more similar than sessions for different subjects. In this study, we assign each session an embedding vector and train HIVE on this data with $n_\omega = 10$ (see Section 5.5.2). The session-pairwise cosine distances of embedding vectors are plotted in Figure 5.8a, which shows a clear block diagonal structure - session embedding vectors from the same subject are closer together than those from different subjects. This shows the model is able to identify certain recordings which have similar deviations from the group, despite HIVE being trained in an unsupervised manner with no knowledge of which sessions belong to which subjects.

To compare with the traditional approaches, we trained both HIVE and HMM-DE on this dataset. In Figure 5.8b, the session-pairwise L2 distances of inferred covariances from both approaches are plotted. Although the same block diagonal structure can be seen in both approaches, it is clearer in the case of HIVE. One can observe there is a particular session (session 3 of subject 8) that has much higher distance with all other sessions, especially with HMM-DE. This is related to the fact that this particular session has very different oscillatory activity compared to other sessions (see Section C.2). In order to quantify the advantages with HIVE, we employ 3 different metrics - Silhouette score ([Rousseeuw, 1987]), Davies-Bouldin score ([Davies and Bouldin, 1979]) and Calinski-Harabasz score ([Caliński and Harabasz, 1974]) for assessing how well the inferred covariances form distinct and well-separated subject clusters. Figure 5.8c shows that with all three metrics, HIVE inferred covariances form tighter and more distinct clusters than the dual estimated covariances from HMM-DE.

5.8.2 Combined dataset

Embedding vectors reflect meaningful variability regarding scanner

In this study, we want to test if our model is able to differentiate data acquired by different scanners. We trained on two different resting-state MEG datasets (Nottingham and Cam-CAN) described in Section 5.6.2. There are 128 subjects in total and 64 subjects from each datasets. To avoid the possibility that the model is biased towards either dataset, we match the age and sex profiles. Here we choose $n_w = 20$ (See Section 5.5.2). We can clearly see two clusters of embedding vectors inferred in Figure 5.9a, and at the same time an age gradient in the embedding space. This means scanner type and age information are simultaneously encoded (in different directions in the embedding space) by the embedding vectors, despite the fact that HIVE was trained unsupervised with no knowledge that the sessions were from different scanner types. We can also see a block diagonal structure in

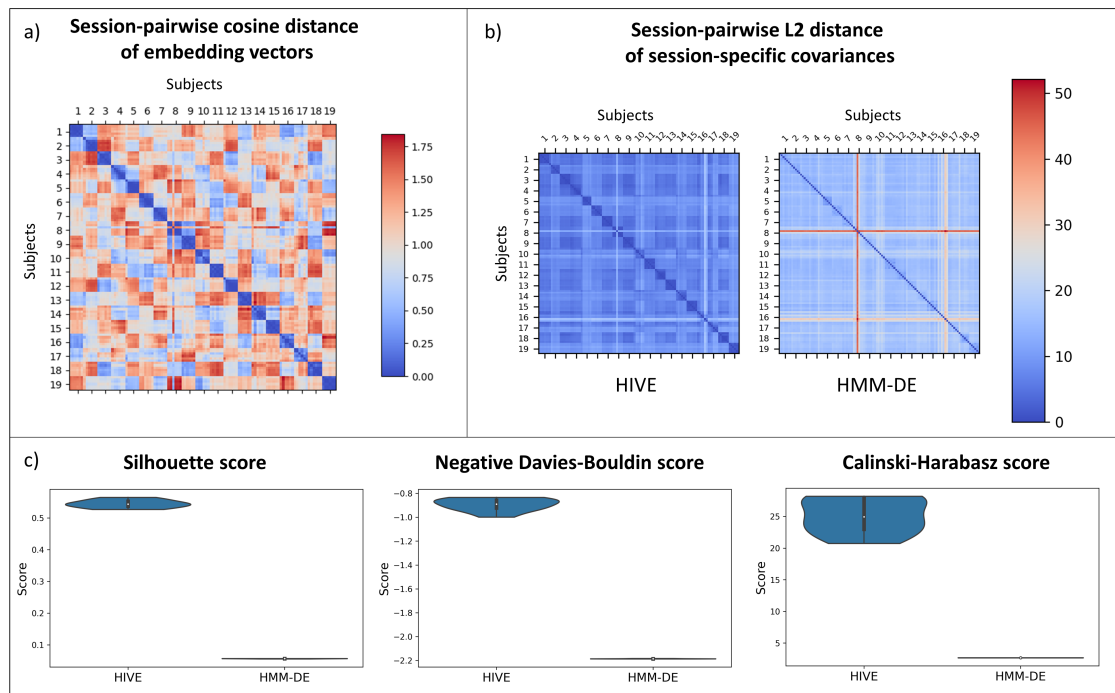


Figure 5.8: **HIVE reveals similarities and differences between MEG recordings.** Results obtained on the *Wakeman-Henson dataset*. a) Session-pairwise cosine distance of inferred embedding vectors. Embedding vectors for the same subject are grouped together and have smaller distance between them. b) Session-pairwise L2 distance of inferred covariances from HIVE (left) and HMM-DE (right). c) Clustering metrics - Silhouette score (left), negative Davies-Bouldin score (middle), Calinski-Harabasz score (right), based on subject labels for 10 independent runs of both approaches. Higher values for these metrics indicate better clustering.

Figure 5.9b where subjects from the same dataset have smaller pairwise cosine distances of their inferred embedding vectors. With the clustering metrics described in Section 5.8.1, we see from Figure 5.9c that subject-specific covariances from HIVE form better-defined clusters based on scanners/sites.

HIVE also provides a way to summarise the differences in state-specific spectral content between scanner types. Ten nearest neighbours in the embedding space are selected for each of the dataset centroids and are used to give a representation of spectral contents of both datasets. In Figure 5.9d, we can see these differences between the “centroids” of the datasets. In particular, subjects from the Nottingham dataset generally have lower power than those from Cam-CAN, especially in posterior regions and alpha band.

5.8.3 Cam-CAN dataset

HIVE infers age structure in the population more accurately than dual estimation

Here we use the Cam-CAN dataset described in Section 5.6.2, that consists of 612 healthy subjects aged between 18-88 and choose $n_\omega = 50$ (See Section 5.5.2). From Figure 5.10a, We can clearly see an age gradient from darker to lighter dots in both plots. This means after training, age information is encoded by the embedding vectors of the subjects, despite the fact that HIVE was trained unsupervised with no knowledge of the subjects’ ages. In order to see if the learnt hidden representation given by the embedding vectors helps improve the model’s ability to distinguish between subject demographics, we try to predict age with the inferred subject-specific covariances. Here HMM-DE and HIVE with increasing embedding dimensions are trained on this dataset. Shown in Figure 5.10b are the distributions of accuracy (given by different folds of cross validation) of predicting age with inferred subject-specific covariances (See below for details). The coefficient of determinant (r^2) is used as a measure of prediction accuracy and it gradually increases with the em-

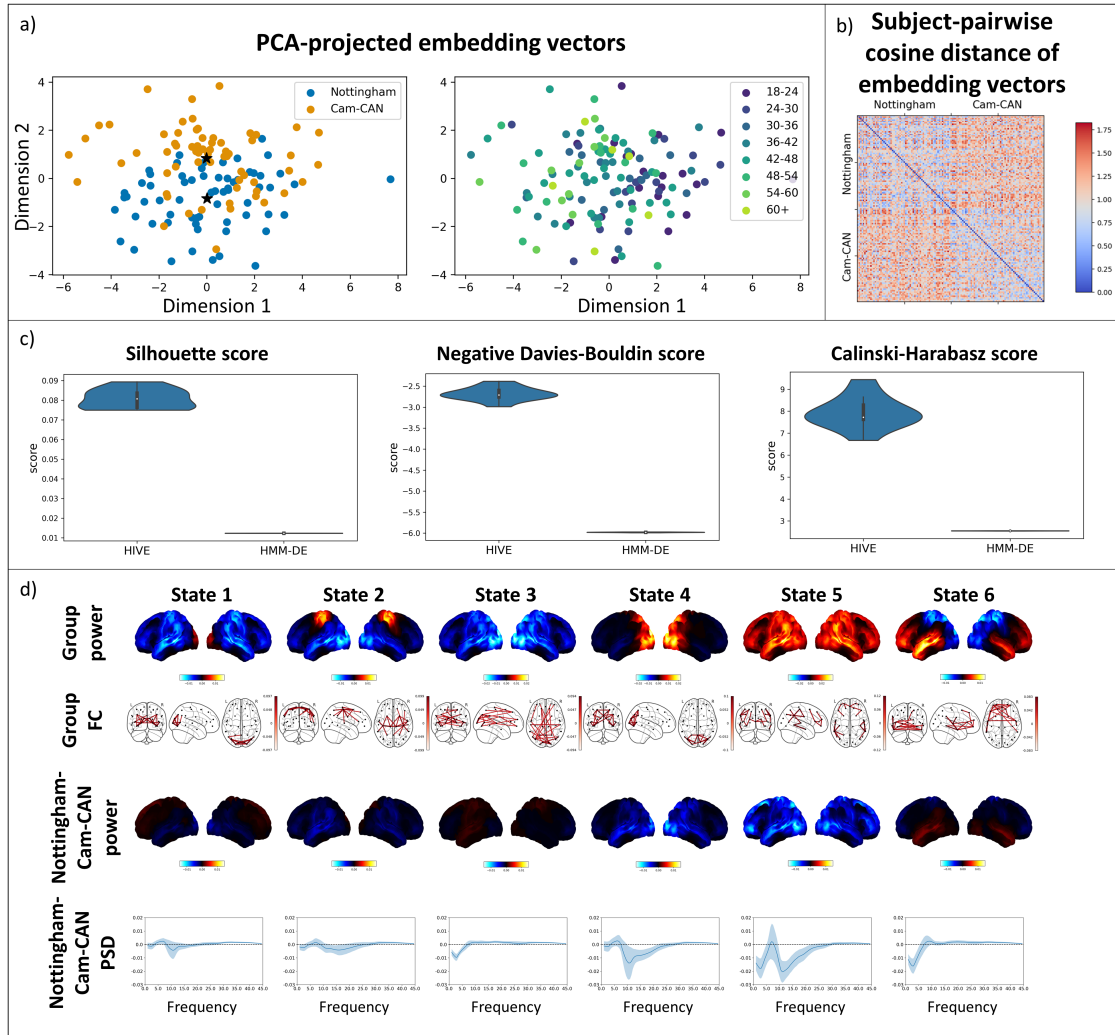


Figure 5.9: **Embedding vectors reflect meaningful variability regarding scanner/site.** Results obtained with a combination of the Nottingham and Cam-CAN datasets. a) PCA-projected embedding vectors coloured with different datasets (left, black stars indicate centroids of embedding vectors from the two datasets) and different age groups (right, older subjects are coloured with lighter colours). b) Subject-pairwise cosine distance of embedding vectors. c) Clustering metrics - Silhouette score (left), negative Davies-Bouldin score (middle), Calinski-Harabasz score (right), based on subject labels for 10 independent runs for both approaches. d) 6 states are inferred with HIVE. The first and second row shows the group level power (red areas show above average and blue areas show below average power across states) and FC maps (top 3% edges are plotted). The third row shows power difference (red areas show higher and blue areas show lower power in Nottingham subjects than Cam-CAN subjects) for each state between centroids of both datasets. The bottom row shows the difference in PSDs (solid lines show the means and shaded areas show one standard deviation across the channels) across frequencies between datasets.

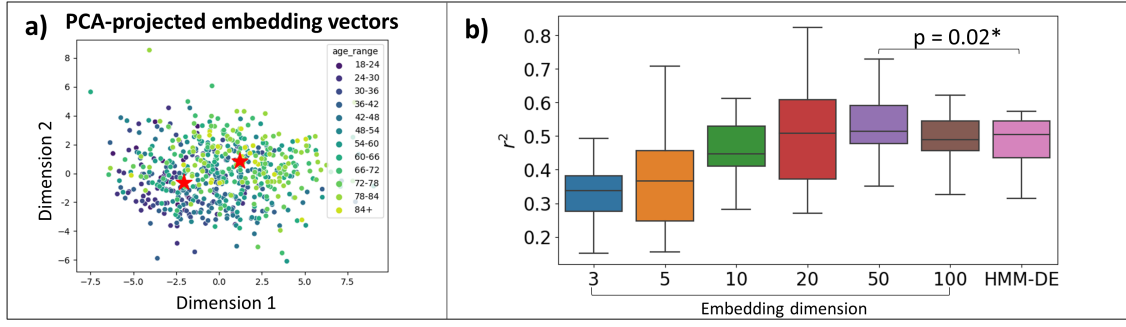


Figure 5.10: **HIVE infers age structure in the population more accurately than dual estimation.** Results obtained with the Cam-CAN dataset. a) Inferred embedding vectors projected to 2 dimensions with PCA. Darker-coloured points are younger participants and lighter-coloured ones are older participants. The two red stars show centroids of subjects with age < 36 and > 72. b) 20-fold cross validated age prediction accuracy for HIVE with different embedding dimensions and HMM-DE.

bedding dimension until there is a significant improvement over HMM-DE when using an embedding dimension of 50. There is a slight drop in accuracy when using an embedding dimension of 100, which could be due to overfitting. Notice that when selecting the embedding dimension, we do not see a significant decrease in free energy if we increase n_ω from 50 to 100 (Figure 5.12).

Classifier for predicting age with subject-specific covariances

HMM-DE and HIVE with embedding dimensions 3, 5, 10, 20, 50, 100 were trained on the Cam-CAN dataset. Each of the 7 models was trained 10 times independently and the best model was chosen based on free energy. Next the inferred subject-specific covariances were projected by PCA, whose first few PCs were used as regressors in a ridge regression to predict subject age. The subject age data were separated into 20 folds. For each fold, a classifier was trained on 19 folds and tested on the remaining 1 fold. During training, a 5-fold cross validated grid search was used to select the best dimension of PCA projection and the regularisation strength of the ridge regression. PCA projection dimension was chosen from {5, 20, 50, 100} and regularisation strength was chosen from $\{10^i : i \in \{-5, -4, -3, -2, -1, 0, 1, 2, 3\}\}$.

HIVE infers interpretable age patterns in networks

Centroids of subjects of age groups < 36 and > 72 were used as representatives of young and old subjects. Similar to the analysis in Section 5.8.2, we can find the nearest neighbours (here we choose 20 subjects) for both representatives and get the power maps, PSDs for young and old subjects. We show the results in Figure 5.11. We can conclude that HIVE can discover meaningful age patterns in the data.

5.8.4 Choosing embedding dimension on the real datasets

In this section we fill in the details behind choosing n_ω for different datasets. We use the procedure described in Section 5.5.2 to choose n_ω for the 3 real dataset results. For the model trained on the Wakeman-Henson dataset in Section 5.8.1, the decrease in free energy is insignificant when comparing $n_\omega = 10$ and $n_\omega = 20$. Hence for this dataset, we choose $n_\omega = 10$. For the model trained on the combined dataset in Section 5.8.2, the tests are all significant and we choose $n_\omega = 20$. For the model trained on the Cam-CAN dataset in Section 5.8.3, the tests are all significant until $n_\omega = 50$. Hence we choose $n_\omega = 50$ for this dataset. See Figure 5.12 for details.

In fact, one may notice that the procedure described in Algorithm 2 is not perfect. See Appendix C.3 for possible improvements to this algorithm.

5.9 Variability and model complexity

The number of state/modes is an important hyper-parameter in HMM and DyNeMo that could potentially affect the interpretation of the data. Previous studies show the variational free energy of these models keep decreasing with increasing number of states/modes ([Baker et al., 2014, Gohil et al., 2022]). We hypothesise that this could be due to variability in the data.

To test this hypothesis, we simulate two datasets, one without variability, the other with variability. The ground truth number of states for both datasets is 4. The

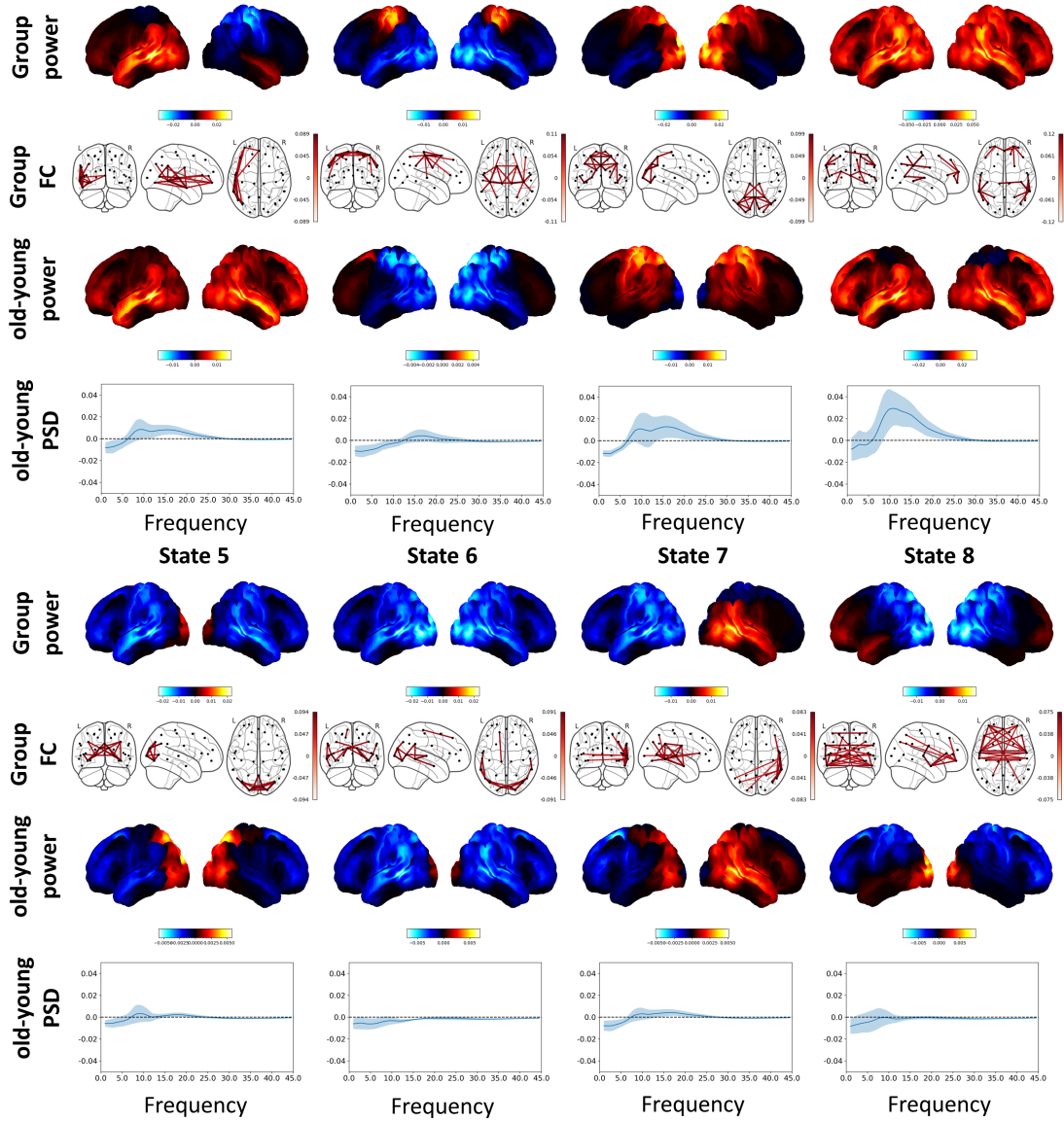


Figure 5.11: **HIVE infers interpretable age patterns in networks.** *Results obtained with the Cam-CAN dataset.* 8 states are inferred by HIVE. Group level power (red areas show above average and blue areas show below average power across states) and FC maps (top 3% edges are plotted) are shown as well as the differences in power (red areas show higher and blue areas show lower power in older subjects than younger subjects) and PSDs (solid lines show the means and shaded areas show one standard deviation across the channels) between old and young subjects.

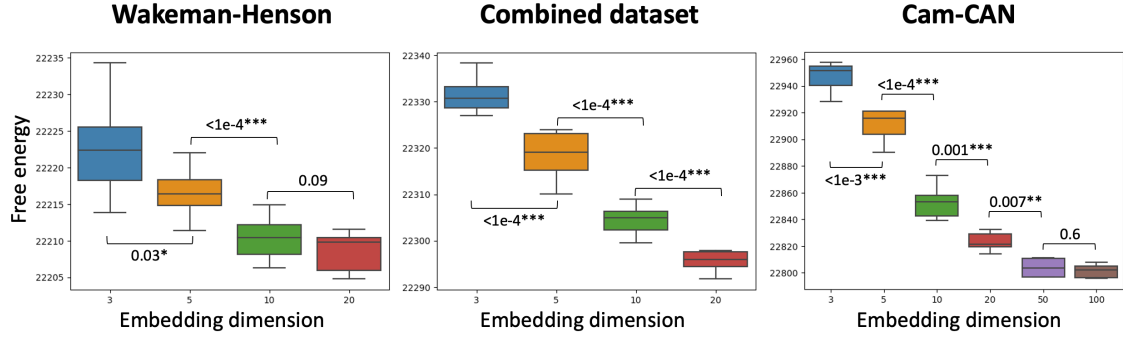


Figure 5.12: **Free energy is used to choose n_ω .** For each of the real datasets results, different models with different candidate embedding dimensions are trained 10 times independently. The free energy on training data is plotted against embedding dimension. The p-values of the t-tests are plotted between the boxplots.

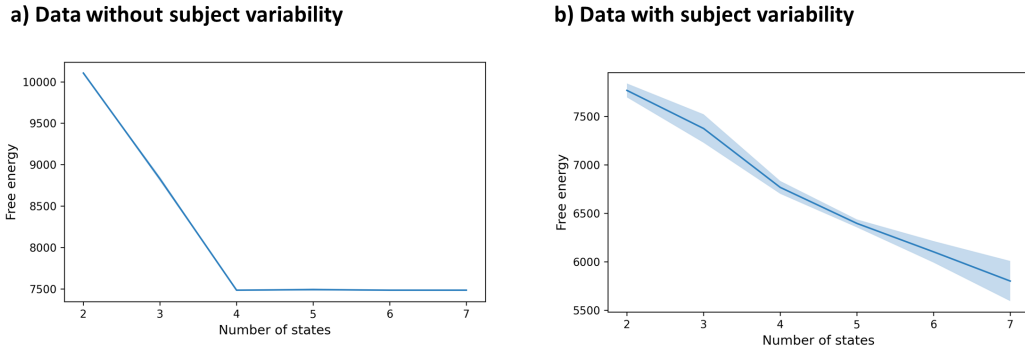


Figure 5.13: **Variability in the data can cause ever-decreasing free energy in HMM.** a) There is no subject variability in the data and the variational free energy is plotted against number of states. b) There is subject variability in the data and the variational free energy is plotted against the number of states

HMM was trained 5 times independently on both datasets and the free energy on training data is plotted in Figure 5.13. It shows that when there is no variability, the free energy decreases with increasing number of states until it plateaus at the ground truth number of states. However, when there is variability, the free energy keeps decreasing with increasing number of states. This is because the model can keep finding more states to explain the variability in the data. By modelling variability in the data, HIVE could be a potential solution to the problem of choosing the number of states.

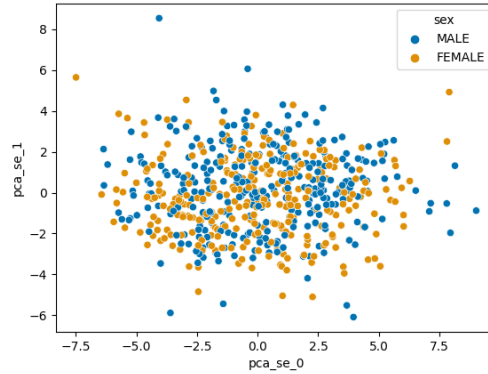


Figure 5.14: **Sex differences are not visible from PCA-projected embedding vectors.** The same set of embedding vectors shown in Figure 5.10a are coloured with the sex of individual. Embedding vectors associated with male and female participants are coloured with blue and orange respectively.

5.10 Constraining embedding vectors

In Sections 5.8.2 and 5.8.3, we saw a clear age gradient in the PCA-projected embedding vectors. However, if we colour the points instead with the sex of the subject, we cannot see clear clusters according to sex (see Figure 5.14). There are two possible reasons for this, one of which is that PCA looks for directions of largest variance but sex is not the biggest source of variation in the dataset. The other relates to the first one in the sense that because variation due to sex is not as prominent as variations due to age and scanning sites, the model is not rewarded to look for sex variations during minimisation of loss.

One possible solution is by putting constraints and forcing a structure on the embedding space. We do this by assuming the embedding space is formed by the (orthogonal) sum of individual embedding spaces defined by different sources of variation. By using this approach, variability in the data due to different sources of variation are isolated to the corresponding individual embedding spaces, which regularises the embedding space and allows us to post-hoc investigate the effects of each of the sources of variation by only looking at the corresponding embedding space.

Instead of assigning an embedding vector to each session, we assign an embedding vector to each condition of different sources of variation. For example, consider the case where there are 3 subjects and each subject undergoes scanning with 3 different tasks, totalling 9 sessions. We would like to assign an embedding vector for each of the 3 subjects and each of the 3 tasks. When computing the embedding vector for each session, we can simply add up (or concatenate) the embedding vectors, which is a common practice in literature to add extra information to the embedding vectors (e.g. the use of positional encoding to provide positional information for transformers ([Vaswani et al., 2017])). Formally, let $\mathcal{S}$ be the set of sources of variations (e.g. task, sex, age) and for each $s \in \mathcal{S}$, $\mathcal{C}_s$ the set of different conditions for source s (e.g. male and female or eyes-open and eyes-closed resting state), then for each session i ,

$$\omega^i = \sum_{s \in \mathcal{S}} \sum_{c \in \mathcal{C}_s} \mathbb{1}\{C_s(i) = c\} \Omega_s(c) \quad (5.11)$$

where $C_s : [N] \rightarrow \mathcal{C}_s$ returns the condition of source s of session i and $\Omega_s(c)$ is the embedding vector corresponding to condition c in the matrix of embedding vectors Ω_s for source s .

Notice this formulation allows us to train on data where there are missing combinations of sources of variation as long as $\forall s \in \mathcal{S}, \forall c \in \mathcal{C}_s, \exists i \in [N] \text{ s.t. } C_s(i) = c$. Consequently, we are able to predict the embedding vector for a missing combination. For instance, consider a dataset where some subjects have both eyes-open (EO) and eyes-closed (EC) resting state recordings, but some subjects only have EO recording sessions. With this formulation, we are able to train on this dataset and recover the session embedding vector for EC sessions for those undergo only EO scans. This is extremely useful as we can leverage what we learn about the relationship between EO and EC to reconstruct the networks of EC recordings for subjects without EC recordings through the variability encoding block. We show its potential through a simulation example.

5.10.1 Simulation example

Here consider the case where 3 subjects, each of them performing 3 tasks, totalling 9 recording sessions. We do this by simulating 3 embedding vectors for each subject and 3 embedding vectors for each task. Figure 5.15 shows the ground truth simulated embeddings on the left. In particular, embedding vectors for different tasks form different clusters and the same subject have the same deviation from the group for different tasks. Time series data are then generated through the HMM generative model similar to Appendix A.4 and A.2.

During training, the session with task 0 and subject 0 is not included in the training set and the right plot of Figure 5.15 shows the inferred embedding vectors of HIVE in which the circled vector corresponds to the masked session and was generated by adding up the corresponding learnt embedding vectors of task ID and subject ID. We see that HIVE learns the relationship between subjects and tasks independently and was able to recover the missing combination of task 0 and subject 0. Moreover, the generated embedding vector of the masked session can be passed through the learnt variability encoding block to generate covariance matrices for the missing session. From Figure 5.16 we see that the covariance matrices are recovered with good accuracy.

5.11 Discussion

In this chapter we propose the use of embedding vectors as a means of characterising individual sessions in a dataset similar to how word embedding vectors characterise semantic differences between words in a word dictionary. We use this approach to extend the HMM, referring to the new method as HIVE. In particular, we propose a novel variability encoding block to translate the information encoded in the embedding space to session-specific networks (covariances), which are then fed to the HMM generative model.

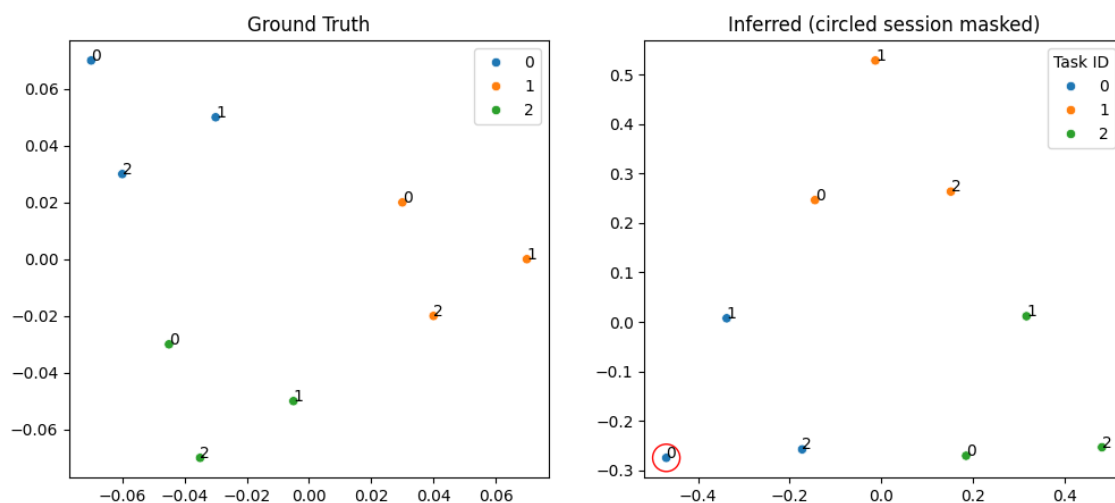


Figure 5.15: **The embedding vector of the masked session can be recovered.** Ground truth simulated embedding vectors (left) and inferred embedding vectors are plotted (right). Each point is coloured according to the task ID and annotated by the subject ID. The circled session on the right indicate the masked session during training.

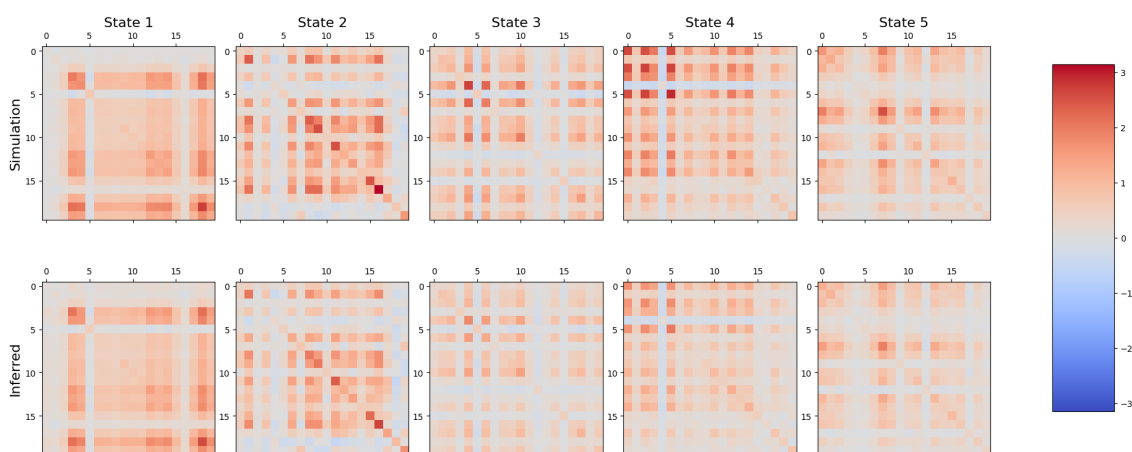


Figure 5.16: **The covariance matrices of the masked session can be recovered.** Ground truth simulated (top row) and generated (bottom row) covariance matrices of the masked session are plotted. Each column shows the covariance matrices for each state.

Using simulations, we show that during inference the variability encoding block learns multivariate session-specific deviations (Section 5.7). This is an important feature because multivariate session-specific deviations are expected in real data and this model is capable of learning these. Additionally, we also see that the model can accurately infer the pairwise relationships between sessions, i.e. discover subpopulation structure through the embedding vectors. This allows the model to perform more accurate inference than dual estimation on individual sessions with the HMM (Section 5.7.3).

Similar to the approach taken in PROFUMO ([Harrison et al., 2015, Harrison et al., 2020]), session deviations from the group-level estimates are generated through a Bayesian prior. The additional feature of embedding vectors allows the model to find subpopulations in the group. By training HIVE on the Wakeman-Henson dataset, we observe that inter-subject variability is much greater than inter-session variability (Section 5.8.1). This was also found in fMRI literature ([Gratton et al., 2018]). Although the HIVE inferred covariances have slightly underestimated deviations from the group due to the regularising effect of the Bayesian prior, it is robust to noise in the dataset (Figure 5.8). Furthermore, the covariances from this approach form better separated clusters due to this de-noising effect compared to the traditional HMM-DE.

In Section 5.8.2, we can observe systematic differences between scanners by training HIVE on data from different sites. The model is trained in an unsupervised fashion and is unaware of the scanners/sites. By finding the centroids of embedding vectors of both datasets and pooling over nearest neighbours, we are able to find the power maps, FC maps and PSD profiles of representatives from both datasets. This allows us to study the systematic differences between scanners. Moreover, HIVE inferred covariances form better-separated clusters than HMM-DE inferred ones based on dataset.

Finally, in Section 5.8.3, by training on a large-scale dataset, we show that

the inferred embedding vectors contains age information and there is a clear age gradient. By training classifiers to predict subject age with inferred subject-specific covariances, we see that accuracy increases with embedding dimension and at some point, HIVE inferred features are able to outperform HMM-DE inferred ones. By picking representatives of young and old subjects, and again pooling over information of nearest neighbours, we can acquire brain networks and PSD profiles of young and old brains.

5.11.1 Future work

In this chapter we incorporated embedding vectors into the HMM to model individual sessions. The variability encoding block used in this work is also compatible with another popular model for inferring dynamic functional networks - DyNeMo (see Appendix D for details and simulation results). We have shown that HIVE provides a way of learning the underlying population structure, which can be used to regularise the estimates of session-specific measures (e.g. PSD). This could be potentially useful in boutique studies which has relatively small number of subjects and contains subjects with specific demographics. By training our proposed model on a large-scale dataset and using the learned population structure, subject-specific estimates could be better inferred on the small dataset. Additionally, although the model has been shown to be robust with around 3000 data points per subject (12s of MEG data on 250Hz), the lower bound for number of data points for robust inference is still unknown. Further work on this matter is needed and essential for this model to be trained on fMRI data where the number of subjects is much larger but number of samples per subject is much smaller. Furthermore, we have shown that HIVE gives better clustering results than dual estimation. A next step will be to investigate the possibility of utilising this advantage in demographics and disease prediction. Lastly, the model provides a natural way to provide subject/session information (e.g. task, age, sex, etc) via the embedding vectors (Section 5.10).

This could be useful when there is an intrinsic subpopulation structure in the group (e.g. multiple sessions for different subjects and both task and resting-state data are available for the subjects).

5.11.2 Hyper-parameters

Due to the additional complexity of the model, it has more hyper-parameters than HMM and DyNeMo, including the number of layers, number of neurons per layer in the decoders of the variability encoding block. However in practice, we find that results are very robust to the choice of these hyper-parameters.

The number of states/modes is an important hyper-parameter in HMM and DyNeMo that could affect the interpretation of the data. Previous studies show the variational free energy keeps decreasing with increasing number of state/modes ([Baker et al., 2014], [Gohil et al., 2022]). We hypothesise that this could be due to the inter-session variability in the data (Section 5.9). Modelling the variability in the data could be a potential solution to this problem.

5.12 Conclusions

We proposed the use of embedding vectors to model individual functional neuroimaging sessions and applied this approach to extend the HMM, giving us HIVE. The variability encoding block explicitly models variability within a population in a principled way. We provide a way to perform efficient inference on the model parameters and the algorithm is readily scalable to large amount of data. With a Bayesian prior, the model pools information across individuals for how they may deviate from the group mean. The embedding vectors allow the model to group together similar data and help the interpretation of sources of variation in a population. This is an important step towards making use of the large-scale datasets collected with different protocols. We believe the above results demonstrate that the proposed model

provides a novel perspective in population modelling and the inference of dynamic functional networks.

Chapter 6

Conclusion

This thesis primarily concerns the use of computational methods, in particular, techniques in statistical learning and deep learning (Chapter 2) in extracting information from functional neuroimaging data. We describe two of the popular methods for estimating dynamic brain networks in Chapter 3, and aim to extend existing methods in order to overcome their limitations.

The work done in Chapter 4 is motivated by the observation that the model free heuristic approach (sliding window approach followed by clustering) identifies uncoupled state time courses for power and FC. However, the sliding window approach can fail to accurately infer the transient, sub-second dynamics present in electrophysiological data and we resort to generative models. We then extended the recent DyNeMo model to M-DyNeMo to allow inference of potentially uncoupled dynamics. With simulated data, we show the advantages of M-DyNeMo over DyNeMo when there is indeed uncoupled power and FC dynamics and crucially, M-DyNeMo does not “invent” uncoupled dynamics if the ground truth is that the two dynamics are coupled, meaning it is solely the data itself that determines whether there are separate dynamics or not. We then move onto validating M-DyNeMo with real MEG data and we show that the model reveals power and FC networks have similar spatial patterns but have uncoupled dynamics at the same time, which is

robust and reproducible across different subsets of subjects, different datasets, and different parcellations. We further show that the two uncoupled dynamics respond to visual stimulus differently and there is evidence of the coupling between dynamics being modulated by task stimuli.

In contrast to developing a model that helps gain more insights about the mechanisms of the brain directly as in Chapter 4, the framework proposed in Chapter 5 provides a way to extend many current generative models to incorporate the modelling of variability in a principled way, providing more accurate group-level and individual-level estimates at the same time. Motivated by how word embeddings are used in NLP to characterise semantic relationships between words, we use embedding vectors to characterise relationships between recording sessions. We show how this approach learns the subpopulation structure and is able to extract meaningful sources of variation within and between datasets. The work done in this chapter opens a path into accurate inference and analysis on multiple datasets simultaneously, and normative modelling/transfer learning by leveraging information learnt in large datasets to help inference on small datasets.

This body of work is by no means complete and flawless, and there are countless directions in which we can explore. In Section 5.10, we discuss a way to constrain the embedding space and provide known information to the model and have shown its potential, with simulation, in generating networks of unseen recording sessions. Additionally, we tried using Gaussian Process to capture (theoretically) unlimited long range temporal dependencies in a non-parametric way (Appendix E). Another challenging, yet important direction is to develop a way to compare different generative models quantitatively. For instance, we can show qualitatively the advantages of DyNeMo over the HMM ([Gohil et al., 2022, Gohil et al., 2024b]), of M-DyNeMo over DyNeMo (Chapter 4, and of HIVE over the HMM (Chapter 5), but these interpretations all depends somewhat on prior knowledge and heuristics. Section 3.3 lists some of our attempts to develop quantitative metrics to compare different generative

models, but they need further validation.

Finally, the problem of how to choose the number of states in the HMM and the number of modes in DyNeMo-based models remains unsolved. This hyperparameter in particular has profound influence on the interpretability and reproducibility of the results, which might hinder the generalisation of the developed methods. To date the choice of this hyperparameter has been based on a combination of heuristics and “hacky” metrics (e.g. split-half reproducibility). Therefore, developing a general framework for choosing this hyperparameter with less heuristics is paramount for these generative models to be applied more widely.

Summary

In summary, in this thesis we attempted to address two questions in computational neuroscience concerning dynamic FC by utilising probabilistic generative models. We developed a dedicated computational method that provides evidence of uncoupled power and FC dynamics and how cognition is related to the coupling between the dynamics. We also provided a framework that can be used to extend popular generative models for inferring dynamic FC so that variability is modelled in a principled way. We believe that with computational methods, the work in this thesis made an incremental step in the marathon of our ultimate goal: understanding the brain.

Bibliography

- [Acevedo et al., 2018] Acevedo, B., Aron, E., Pospos, S., and Jessen, D. (2018). The functional highly sensitive brain: a review of the brain circuits underlying sensory processing sensitivity and seemingly related disorders. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 373(1744):20170161.
- [Allen et al., 2014] Allen, E. A., Damaraju, E., Plis, S. M., Erhardt, E. B., Eichele, T., and Calhoun, V. D. (2014). Tracking whole-brain connectivity dynamics in the resting state. *Cerebral cortex*, 24(3):663–676.
- [Armony and Vuilleumier, 2013] Armony, J. and Vuilleumier, P. (2013). *The Cambridge handbook of human affective neuroscience*. Cambridge university press.
- [Ba et al., 2016] Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.
- [Baddeley, 2007] Baddeley, A. (2007). *Working memory, thought, and action*, volume 45. OUP Oxford.
- [Baker et al., 2014] Baker, A. P., Brookes, M. J., Rezek, I. A., Smith, S. M., Behrens, T., Probert Smith, P. J., and Woolrich, M. (2014). Fast transient networks in spontaneous human brain activity. *elife*, 3:e01867.
- [Barttfeld et al., 2015] Barttfeld, P., Uhrig, L., Sitt, J. D., Sigman, M., Jarraya, B., and Dehaene, S. (2015). Signature of consciousness in the dynamics of resting-

state brain activity. *Proceedings of the National Academy of Sciences*, 112(3):887–892.

[Baum and Eagon, 1967] Baum, L. E. and Eagon, J. A. (1967). An inequality with applications to statistical estimation for probabilistic functions of markov processes and to a model for ecology. *Bulletin of the American Mathematical Society*, 73(3):360–363.

[Baum and Petrie, 1966] Baum, L. E. and Petrie, T. (1966). Statistical inference for probabilistic functions of finite state markov chains. *The annals of mathematical statistics*, 37(6):1554–1563.

[Beckmann et al., 2005] Beckmann, C. F., DeLuca, M., Devlin, J. T., and Smith, S. M. (2005). Investigations into resting-state connectivity using independent component analysis. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 360(1457):1001–1013.

[Beckmann et al., 2009] Beckmann, C. F., Mackay, C. E., Filippini, N., Smith, S. M., et al. (2009). Group comparison of resting-state fMRI data using multi-subject ICA and dual regression. *Neuroimage*, 47(Suppl 1):S148.

[Bengio et al., 1994] Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

[Blei et al., 2017] Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877.

[Bottou, 1998] Bottou, L. (1998). Online algorithms and stochastic approximations. In Saad, D., editor, *Online Learning and Neural Networks*. Cambridge University Press, Cambridge, UK. revised, oct 2012.

- [Bowman et al., 2015] Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., and Bengio, S. (2015). Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*.
- [Bowyer, 2016] Bowyer, S. M. (2016). Coherence a measure of the brain networks: past and present. *Neuropsychiatric Electrophysiology*, 2:1–12.
- [Brookes et al., 2011] Brookes, M. J., Woolrich, M., Luckhoo, H., Price, D., Hale, J. R., Stephenson, M. C., Barnes, G. R., Smith, S. M., and Morris, P. G. (2011). Investigating the electrophysiological basis of resting state networks using magnetoencephalography. *Proceedings of the National Academy of Sciences*, 108(40):16783–16788.
- [Caliński and Harabasz, 1974] Caliński, T. and Harabasz, J. (1974). A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods*, 3(1):1–27.
- [Chang and Glover, 2010] Chang, C. and Glover, G. H. (2010). Time–frequency dynamics of resting-state brain connectivity measured with fMRI. *Neuroimage*, 50(1):81–98.
- [Chehab et al., 2021] Chehab, O., Defossez, A., Loiseau, J.-C., Gramfort, A., and King, J.-R. (2021). Deep recurrent encoder: A scalable end-to-end network to model brain signals. *arXiv preprint arXiv:2103.02339*.
- [Cheyne, 2013] Cheyne, D. O. (2013). MEG studies of sensorimotor rhythms: a review. *Experimental neurology*, 245:27–39.
- [Colclough et al., 2015] Colclough, G. L., Brookes, M. J., Smith, S. M., and Woolrich, M. W. (2015). A symmetric multivariate leakage correction for meg connectomes. *Neuroimage*, 117:439–448.

- [Cole et al., 2014] Cole, M. W., Bassett, D. S., Power, J. D., Braver, T. S., and Petersen, S. E. (2014). Intrinsic and task-evoked network architectures of the human brain. *Neuron*, 83(1):238–251.
- [Comon, 1994] Comon, P. (1994). Independent component analysis, a new concept? *Signal processing*, 36(3):287–314.
- [Csaky et al., 2023] Csaky, R., Van Es, M. W., Parker Jones, O., and Woolrich, M. (2023). Group-level brain decoding with deep learning. *Human Brain Mapping*, 44(17):6105–6119.
- [Damianou and Lawrence, 2013] Damianou, A. and Lawrence, N. D. (2013). Deep gaussian processes. In *Artificial intelligence and statistics*, pages 207–215. PMLR.
- [David et al., 2006] David, O., Kilner, J. M., and Friston, K. J. (2006). Mechanisms of evoked and induced responses in MEG/EEG. *Neuroimage*, 31(4):1580–1591.
- [Davies and Bouldin, 1979] Davies, D. L. and Bouldin, D. W. (1979). A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, (2):224–227.
- [Défossez et al., 2023] Défossez, A., Caucheteux, C., Rapin, J., Kabeli, O., and King, J.-R. (2023). Decoding speech perception from non-invasive brain recordings. *Nature Machine Intelligence*, 5(10):1097–1107.
- [DeGroot and Schervish, 2012] DeGroot, M. and Schervish, M. (2012). *Probability and Statistics*, pages 213–214. Addison-Wesley.
- [Del Moral et al., 2006] Del Moral, P., Doucet, A., and Jasra, A. (2006). Sequential monte carlo samplers. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 68(3):411–436.

- [Dempster et al., 1977] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society: series B (methodological)*, 39(1):1–22.
- [Diamond, 2013] Diamond, A. (2013). Executive functions. *Annual review of psychology*, 64:135–168.
- [Esposito et al., 2006] Esposito, F., Bertolino, A., Scarabino, T., Latorre, V., Blasi, G., Popolizio, T., Tedeschi, G., Cirillo, S., Goebel, R., and Di Salle, F. (2006). Independent component model of the default-mode brain function: Assessing the impact of active thinking. *Brain research bulletin*, 70(4-6):263–269.
- [Eysenck, 2012] Eysenck, M. (2012). *Attention and arousal: Cognition and performance*. Springer Science & Business Media.
- [Fair et al., 2009] Fair, D. A., Cohen, A. L., Power, J. D., Dosenbach, N. U., Church, J. A., Miezin, F. M., Schlaggar, B. L., and Petersen, S. E. (2009). Functional brain networks develop from a “local to distributed” organization. *PLoS computational biology*, 5(5):e1000381.
- [Faskowitz et al., 2020] Faskowitz, J., Esfahlani, F. Z., Jo, Y., Sporns, O., and Betzel, R. F. (2020). Edge-centric functional network representations of human cerebral cortex reveal overlapping system-level architecture. *Nature neuroscience*, 23(12):1644–1654.
- [Feldman, 2011] Feldman, M. (2011). Hilbert transform in vibration analysis. *Mechanical systems and signal processing*, 25(3):735–802.
- [Fernández et al., 2006] Fernández, A., Hornero, R., Mayo, A., Poza, J., Gil-Gregorio, P., and Ortiz, T. (2006). MEG spectral profile in Alzheimer’s disease and mild cognitive impairment. *Clinical Neurophysiology*, 117(2):306–314.

- [Fornito et al., 2012] Fornito, A., Harrison, B. J., Zalesky, A., and Simons, J. S. (2012). Competitive and cooperative dynamics of large-scale brain functional networks supporting recollection. *Proceedings of the National Academy of Sciences*, 109(31):12788–12793.
- [Fries, 2015] Fries, P. (2015). Rhythms for cognition: communication through coherence. *Neuron*, 88(1):220–235.
- [Friston, 1994] Friston, K. J. (1994). Functional and effective connectivity in neuroimaging: a synthesis. *Human brain mapping*, 2(1-2):56–78.
- [Friston, 2011] Friston, K. J. (2011). Functional and effective connectivity: a review. *Brain connectivity*, 1(1):13–36.
- [Fu and Perry, 2020] Fu, W. and Perry, P. O. (2020). Estimating the number of clusters using cross-validation. *Journal of Computational and Graphical Statistics*, 29(1):162–173.
- [Gardner and Dorling, 1998] Gardner, M. W. and Dorling, S. (1998). Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment*, 32(14-15):2627–2636.
- [Gohil et al., 2024a] Gohil, C., Huang, R., Roberts, E., van Es, M. W., Quinn, A. J., Vidaurre, D., and Woolrich, M. W. (2024a). osl-dynamics, a toolbox for modeling fast dynamic brain activity. *Elife*, 12:RP91949.
- [Gohil et al., 2024b] Gohil, C., Kohl, O., Huang, R., van Es, M. W., Parker Jones, O., Hunt, L. T., Quinn, A. J., and Woolrich, M. W. (2024b). Dynamic network analysis of electrophysiological task data. *bioRxiv*, pages 2024–01.
- [Gohil et al., 2022] Gohil, C., Roberts, E., Timms, R., Skates, A., Higgins, C., Quinn, A., Pervaiz, U., van Amersfoort, J., Notin, P., Gal, Y., et al. (2022).

- Mixtures of large-scale dynamic functional brain network modes. *NeuroImage*, 263:119595.
- [Gratton et al., 2018] Gratton, C., Laumann, T. O., Nielsen, A. N., Greene, D. J., Gordon, E. M., Gilmore, A. W., Nelson, S. M., Coalson, R. S., Snyder, A. Z., Schlaggar, B. L., et al. (2018). Functional brain networks are dominated by stable group and individual factors, not cognitive or daily variation. *Neuron*, 98(2):439–452.
- [Greicius, 2008] Greicius, M. (2008). Resting-state functional connectivity in neuropsychiatric disorders. *Current opinion in neurology*, 21(4):424–430.
- [Gu et al., 2021] Gu, A., Goel, K., and Ré, C. (2021). Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*.
- [Guyton and Hall, 2011] Guyton, A. C. and Hall, J. E. (2011). *Guyton and Hall textbook of medical physiology*. Elsevier.
- [Harrison et al., 2020] Harrison, S. J., Bijsterbosch, J. D., Segerdahl, A. R., Fitzgibbon, S. P., Farahibozorg, S.-R., Duff, E. P., Smith, S. M., and Woolrich, M. W. (2020). Modelling subject variability in the spatial and temporal characteristics of functional modes. *NeuroImage*, 222:117226.
- [Harrison et al., 2015] Harrison, S. J., Woolrich, M. W., Robinson, E. C., Glasser, M. F., Beckmann, C. F., Jenkinson, M., and Smith, S. M. (2015). Large-scale probabilistic functional modes from resting state fMRI. *NeuroImage*, 109:217–231.
- [Hastie et al., 2009] Hastie, T., Tibshirani, R., Friedman, J. H., and Friedman, J. H. (2009). *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer.
- [Hastings, 1970] Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109.

- [Heine et al., 2012] Heine, L., Soddu, A., Gómez, F., Vanhaudenhuyse, A., Tshibanda, L., Thonnard, M., Charland-Verville, V., Kirsch, M., Laureys, S., and Demertzi, A. (2012). Resting state networks and consciousness. *Frontiers in psychology*, 3:26774.
- [Hellier, 2014] Hellier, J. L. (2014). *The Brain, the Nervous System, and Their Diseases*. Bloomsbury Publishing USA.
- [Higgins et al., 2021] Higgins, C., Liu, Y., Vidaurre, D., Kurth-Nelson, Z., Dolan, R., Behrens, T., and Woolrich, M. (2021). Replay bursts in humans coincide with activation of the default mode and parietal alpha networks. *Neuron*, 109(5):882–893.
- [Hochberg, 1987] Hochberg, Y. (1987). *Multiple comparison procedures*. Wiley.
- [Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.
- [Honey et al., 2009] Honey, C. J., Sporns, O., Cammoun, L., Gigandet, X., Thiran, J.-P., Meuli, R., and Hagmann, P. (2009). Predicting human resting-state functional connectivity from structural connectivity. *Proceedings of the National Academy of Sciences*, 106(6):2035–2040.
- [Hornik et al., 1989] Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366.
- [Hutchison et al., 2013] Hutchison, R. M., Womelsdorf, T., Allen, E. A., Bandettini, P. A., Calhoun, V. D., Corbetta, M., Della Penna, S., Duyn, J. H., Glover, G. H., Gonzalez-Castillo, J., et al. (2013). Dynamic functional connectivity: promise, issues, and interpretations. *Neuroimage*, 80:360–378.
- [Jain, 2010] Jain, A. K. (2010). Data clustering: 50 years beyond K-means. *Pattern recognition letters*, 31(8):651–666.

- [Jalali et al., 2012] Jalali, A., Azimi, J., and Fern, X. (2012). Exploration vs exploitation in bayesian optimization. *CoRR*.
- [Jayalath et al., 2024] Jayalath, D., Landau, G., Shillingford, B., Woolrich, M., and Parker Jones, O. (2024). The brain’s bitter lesson: Scaling speech decoding with self-supervised learning. *arXiv preprint arXiv:2406.04328*.
- [Jenkinson et al., 2005] Jenkinson, M., Pechaud, M., Smith, S., et al. (2005). BET2: MR-based estimation of brain, skull and scalp surfaces. 17(3):167.
- [Jolliffe and Cadima, 2016] Jolliffe, I. T. and Cadima, J. (2016). Principal component analysis: a review and recent developments. *Philosophical transactions of the royal society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202.
- [Jozefowicz et al., 2015] Jozefowicz, R., Zaremba, W., and Sutskever, I. (2015). An empirical exploration of recurrent network architectures. In *International conference on machine learning*, pages 2342–2350. PMLR.
- [Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- [Kingma and Welling, 2013] Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.
- [Kohl et al., 2024] Kohl, O., Gohil, C., Zokaei, N., Hu, M. T., Nobre, A. C., Woolrich, M., and Quinn, A. (2024). Changes in sensorimotor network dynamics in resting-state recordings in Parkinson’s disease. *medRxiv*, pages 2024–01.
- [Kohl et al., 2023] Kohl, O., Woolrich, M., Nobre, A. C., and Quinn, A. (2023). Glasser52: A parcellation for MEG-Analysis. *Zenodo*.

- [Kramer et al., 2010] Kramer, M. A., Eden, U. T., Kolaczyk, E. D., Zepeda, R., Eskandar, E. N., and Cash, S. S. (2010). Coalescence and fragmentation of cortical networks during focal seizures. *Journal of Neuroscience*, 30(30):10076–10085.
- [Kruschke and Vanpaemel, 2015] Kruschke, J. K. and Vanpaemel, W. (2015). Bayesian estimation in hierarchical models. *The Oxford handbook of computational and mathematical psychology*, pages 279–299.
- [Kuhn, 1955] Kuhn, H. W. (1955). The Hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97.
- [Kullback and Leibler, 1951] Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86.
- [Lehmann et al., 1986a] Lehmann, E. L., Romano, J. P., and Casella, G. (1986a). *Testing statistical hypotheses*, volume 3. Springer.
- [Lehmann et al., 1986b] Lehmann, E. L., Romano, J. P., and Casella, G. (1986b). *Testing statistical hypotheses*, volume 3, page 200. Springer.
- [Liu and Pierce, 1994] Liu, Q. and Pierce, D. A. (1994). A note on Gauss—Hermite quadrature. *Biometrika*, 81(3):624–629.
- [Loehlin, 2004] Loehlin, J. C. (2004). *Latent variable models: An introduction to factor, path, and structural equation analysis*. Psychology Press.
- [Luckhoo et al., 2012] Luckhoo, H., Hale, J. R., Stokes, M. G., Nobre, A. C., Morris, P. G., Brookes, M. J., and Woolrich, M. W. (2012). Inferring task-related networks using independent component analysis in magnetoencephalography. *Neuroimage*, 62(1):530–541.
- [Mandal et al., 2018] Mandal, P. K., Banerjee, A., Tripathi, M., and Sharma, A. (2018). A comprehensive review of magnetoencephalography (MEG) studies for

- brain functionality in healthy aging and Alzheimer’s disease (AD). *Frontiers in computational neuroscience*, 12:60.
- [McFarland and Wolpaw, 2017] McFarland, D. J. and Wolpaw, J. R. (2017). EEG-based brain–computer interfaces. *current opinion in Biomedical Engineering*, 4:194–200.
- [McLachlan and Krishnan, 2007] McLachlan, G. J. and Krishnan, T. (2007). *The EM algorithm and extensions*. John Wiley & Sons.
- [Meindl et al., 2010] Meindl, T., Teipel, S., Elmouden, R., Mueller, S., Koch, W., Dietrich, O., Coates, U., Reiser, M., and Glaser, C. (2010). Test–retest reproducibility of the default-mode network in healthy individuals. *Human brain mapping*, 31(2):237–246.
- [Menon, 2011] Menon, V. (2011). Large-scale brain networks and psychopathology: a unifying triple network model. *Trends in cognitive sciences*, 15(10):483–506.
- [Mikolov et al., 2013] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
- [Miller et al., 2009] Miller, K. J., Sorensen, L. B., Ojemann, J. G., and Den Nijs, M. (2009). Power-law scaling in the brain surface electric potential. *PLoS computational biology*, 5(12):e1000609.
- [Montgomery et al., 2021] Montgomery, D. C., Peck, E. A., and Vining, G. G. (2021). *Introduction to linear regression analysis*. John Wiley & Sons.
- [Nair and Hinton, 2010] Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

- [Nestler et al., 2020] Nestler, E. J., Kenny, P. J., Russo, S. J., and Schaefer, A. (2020). *Molecular neuropharmacology: a foundation for clinical neuroscience, Fourth Edition*. McGraw-Hill Education.
- [Nichols and Hayasaka, 2003] Nichols, T. and Hayasaka, S. (2003). Controlling the familywise error rate in functional neuroimaging: a comparative review. *Statistical methods in medical research*, 12(5):419–446.
- [Nickerson et al., 2017] Nickerson, L. D., Smith, S. M., Öngür, D., and Beckmann, C. F. (2017). Using dual regression to investigate network shape and amplitude in functional connectivity analyses. *Frontiers in neuroscience*, 11:115.
- [Nir et al., 2008] Nir, Y., Mukamel, R., Dinstein, I., Privman, E., Harel, M., Fisch, L., Gelbard-Sagiv, H., Kipervasser, S., Andelman, F., Neufeld, M. Y., et al. (2008). Interhemispheric correlations of slow spontaneous neuronal fluctuations revealed in human sensory cortex. *Nature neuroscience*, 11(9):1100–1108.
- [Ou et al., 2009] Ou, W., Hämäläinen, M. S., and Golland, P. (2009). A distributed spatio-temporal EEG/MEG inverse solver. *NeuroImage*, 44(3):932–946.
- [O’Neill et al., 2015] O’Neill, G. C., Barratt, E. L., Hunt, B. A., Tewarie, P. K., and Brookes, M. J. (2015). Measuring electrophysiological connectivity by power envelope correlation: a technical review on MEG methods. *Physics in Medicine & Biology*, 60(21):R271.
- [Park and Friston, 2013] Park, H.-J. and Friston, K. (2013). Structural and functional brain networks: from connections to cognition. *Science*, 342(6158):1238411.
- [Paszke et al., 2017] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch.

- [Penny et al., 2004] Penny, W. D., Stephan, K. E., Mechelli, A., and Friston, K. J. (2004). Modelling functional integration: a comparison of structural equation and dynamic causal models. *Neuroimage*, 23:S264–S274.
- [Pervaiz et al., 2022] Pervaiz, U., Vidaurre, D., Gohil, C., Smith, S. M., and Woolrich, M. W. (2022). Multi-dynamic modelling reveals strongly time-varying resting fmri correlations. *Medical image analysis*, 77:102366.
- [Pesarin and Salmaso, 2010] Pesarin, F. and Salmaso, L. (2010). The permutation testing approach: a review. *Statistica*, 70(4):481–509.
- [Petersen and Sporns, 2015] Petersen, S. E. and Sporns, O. (2015). Brain networks and cognitive architectures. *Neuron*, 88(1):207–219.
- [Pinel, 2009] Pinel, J. P. (2009). *Biopsychology*. Pearson education.
- [Popescu et al., 2009] Popescu, M.-C., Balas, V. E., Perescu-Popescu, L., and Mastorakis, N. (2009). Multilayer perceptron and neural networks. *WSEAS Transactions on Circuits and Systems*, 8(7):579–588.
- [Quinn et al., 2022] Quinn, A. J., van Es, M., Gohil, C., and Woolrich, M. W. (2022). OHBA Software Library in Python (OSL) (0.1.1). *Zenodo*.
- [Quinn et al., 2018] Quinn, A. J., Vidaurre, D., Abeysuriya, R., Becker, R., Nobre, A. C., and Woolrich, M. W. (2018). Task-evoked dynamic network analysis through hidden markov modeling. *Frontiers in neuroscience*, 12:603.
- [Rabinovich et al., 2012] Rabinovich, M. I., Friston, K. J., and Varona, P. (2012). *Principles of brain dynamics*. MIT Press Cambridge, Mass.
- [Reynolds et al., 2009] Reynolds, D. A. et al. (2009). Gaussian mixture models. *Encyclopedia of biometrics*, 741(659-663).

- [Rezende and Viola, 2018a] Rezende, D. J. and Viola, F. (2018a). Generalized ELBO with Constrained Optimization, GECO. In *Workshop on Bayesian Deep Learning, NeurIPS*.
- [Rezende and Viola, 2018b] Rezende, D. J. and Viola, F. (2018b). Taming VAEs. *arXiv preprint arXiv:1810.00597*.
- [Ribeiro et al., 2020] Ribeiro, A. H., Tiels, K., Aguirre, L. A., and Schön, T. (2020). Beyond exploding and vanishing gradients: analysing RNN training using attractors and smoothness. In *International conference on artificial intelligence and statistics*, pages 2370–2380. PMLR.
- [Rousseeuw, 1987] Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65.
- [Rumelhart et al., 1986] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation, parallel distributed processing, explorations in the microstructure of cognition, ed. de rumelhart and j. mcclelland. vol. 1. 1986. *Biometrika*, 71(599-607):6.
- [Sen and Singer, 1994] Sen, P. K. and Singer, J. M. (1994). *Large sample methods in statistics: an introduction with applications*, volume 25. CRC press.
- [Shahriari et al., 2015] Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., and De Freitas, N. (2015). Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175.
- [Smith, 2002] Smith, S. M. (2002). Fast robust automated brain extraction. *Human brain mapping*, 17(3):143–155.
- [Squire et al., 2012] Squire, L., Berg, D., Bloom, F. E., Du Lac, S., Ghosh, A., and Spitzer, N. C. (2012). *Fundamental neuroscience*. Academic press.

- [Taylor et al., 2017] Taylor, J. R., Williams, N., Cusack, R., Auer, T., Shafto, M. A., Dixon, M., Tyler, L. K., Henson, R. N., et al. (2017). The Cambridge Centre for Ageing and Neuroscience (Cam-CAN) data repository: Structural and functional MRI, MEG, and cognitive data from a cross-sectional adult lifespan sample. *neuroimage*, 144:262–269.
- [Titsias, 2009] Titsias, M. (2009). Variational learning of inducing variables in sparse Gaussian processes. In *Artificial intelligence and statistics*, pages 567–574. PMLR.
- [Tokdar and Kass, 2010] Tokdar, S. T. and Kass, R. E. (2010). Importance sampling: a review. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(1):54–60.
- [Tzourio-Mazoyer et al., 2002] Tzourio-Mazoyer, N., Landeau, B., Papathanassiou, D., Crivello, F., Etard, O., Delcroix, N., Mazoyer, B., and Joliot, M. (2002). Automated anatomical labeling of activations in SPM using a macroscopic anatomical parcellation of the MNI MRI single-subject brain. *Neuroimage*, 15(1):273–289.
- [Van Dijk et al., 2010] Van Dijk, K. R., Hedden, T., Venkataraman, A., Evans, K. C., Lazar, S. W., and Buckner, R. L. (2010). Intrinsic functional connectivity as a tool for human connectomics: theory, properties, and optimization. *Journal of neurophysiology*, 103(1):297–321.
- [Van Veen and Buckley, 1988] Van Veen, B. D. and Buckley, K. M. (1988). Beamforming: A versatile approach to spatial filtering. *IEEE assp magazine*, 5(2):4–24.
- [Van Veen et al., 1997] Van Veen, B. D., Van Drongelen, W., Yuchtman, M., and Suzuki, A. (1997). Localization of brain electrical activity via linearly constrained minimum variance spatial filtering. *IEEE Transactions on biomedical engineering*, 44(9):867–880.

- [Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.
- [Vidaurre et al., 2018a] Vidaurre, D., Abeysuriya, R., Becker, R., Quinn, A. J., Alfaro-Almagro, F., Smith, S. M., and Woolrich, M. W. (2018a). Discovering dynamic brain networks from big data in rest and task. *NeuroImage*, 180:646–656.
- [Vidaurre et al., 2018b] Vidaurre, D., Hunt, L. T., Quinn, A. J., Hunt, B. A., Brookes, M. J., Nobre, A. C., and Woolrich, M. W. (2018b). Spontaneous cortical activity transiently organises into frequency specific phase-coupling networks. *Nature communications*, 9(1):2987.
- [Vidaurre et al., 2021] Vidaurre, D., Llera, A., Smith, S. M., and Woolrich, M. W. (2021). Behavioural relevance of spontaneous, transient brain network interactions in fMRI. *Neuroimage*, 229:117713.
- [Vidaurre et al., 2016] Vidaurre, D., Quinn, A. J., Baker, A. P., Dupret, D., Tejero-Cantero, A., and Woolrich, M. W. (2016). Spectrally resolved fast transient brain states in electrophysiological data. *Neuroimage*, 126:81–95.
- [Vidaurre et al., 2017] Vidaurre, D., Smith, S. M., and Woolrich, M. W. (2017). Brain network dynamics are hierarchically organized in time. *Proceedings of the National Academy of Sciences*, 114(48):12827–12832.
- [Wakeman and Henson, 2015] Wakeman, D. G. and Henson, R. N. (2015). A multi-subject, multi-modal human neuroimaging dataset. *Scientific data*, 2(1):1–10.
- [Walter and Barkema, 2015] Walter, J.-C. and Barkema, G. (2015). An introduction to Monte Carlo methods. *Physica A: Statistical Mechanics and its Applications*, 418:78–87.

- [Wang et al., 2018] Wang, H., Zariphopoulou, T., and Zhou, X. (2018). Exploration versus exploitation in reinforcement learning: A stochastic control approach. *arXiv preprint arXiv:1812.01552*.
- [Winkler et al., 2014] Winkler, A. M., Ridgway, G. R., Webster, M. A., Smith, S. M., and Nichols, T. E. (2014). Permutation inference for the general linear model. *Neuroimage*, 92:381–397.
- [Yeo et al., 2011] Yeo, B. T., Krienen, F. M., Sepulcre, J., Sabuncu, M. R., Lashkari, D., Hollinshead, M., Roffman, J. L., Smoller, J. W., Zöllei, L., Polimeni, J. R., et al. (2011). The organization of the human cerebral cortex estimated by intrinsic functional connectivity. *Journal of neurophysiology*.
- [Zhang et al., 2021] Zhang, S., Cao, C., Quinn, A., Vivekananda, U., Zhan, S., Liu, W., Sun, B., Woolrich, M., Lu, Q., and Litvak, V. (2021). Dynamic analysis on simultaneous iEEG-MEG data via hidden Markov model. *NeuroImage*, 233:117923.

Appendix A

Data simulation

In this thesis, data simulation involves two components:

- Simulating the underlying hidden time courses of the brain dynamics.
- Simulating the observed data given the hidden time courses with the observation model.

A.1 Simulating the hidden time courses

Binary time courses can be generated with a Markov chain, i.e. given the number of states J , the transition probability matrix $\mathbf{A}$, we generate the hidden time course $\mathbf{z}_{1:T}$ with

$$z_t | z_{t-1} \sim \text{Cat}(\mathbf{A}_{z_{t-1},:}) \quad \text{for } t = 2, \dots, T, \quad (\text{A.1})$$

where z_1 is fixed to be the first state. Soft mixing time courses can be generated by combining logits, which are sine waves with different frequencies, amplitudes, and phases, i.e.

$$\theta_{jt} = \text{relative_activation}_j + \text{amplitude}_j * \sin(2\pi f_j t + \phi_j), \quad (\text{A.2})$$

where $f_j, amplitude_j, relative_activation_j$ are pre-specified frequency, amplitude and relative activation of mode j , and ϕ_j is the phase of mode j , which is uniformly sampled from $U[0, 2\pi]$. The hidden time course $\alpha_{1:T}$ (i.e. mixing probabilities) is generated by applying a softmax function to the logits at each time t .

A.2 Simulating the observed data

The observed data are assumed to be conditionally independent given the hidden time courses. The observation model is a multivariate Gaussian distribution. For binary time courses, the data is generated with

$$x_t \sim \mathcal{N}(\boldsymbol{\mu}_{z_t}, \boldsymbol{\Sigma}_{z_t}), \quad (\text{A.3})$$

and for soft mixing time courses the data is generated with

$$x_t \sim \mathcal{N}\left(\sum_{j=1}^J \alpha_{jt} \boldsymbol{\mu}_j, \sum_{j=1}^J \alpha_{jt} \boldsymbol{\Sigma}_j\right), \quad (\text{A.4})$$

where $\boldsymbol{\mu}_j$ is either a vector of zeros or each of its entries is sampled independently from $\mathcal{N}(0, 0.2^2)$. The simulation of covariance matrices $\boldsymbol{\Sigma}_j$ is more involved. We first sample a $N_c \times N_c$ matrix W_j whose entries are sampled independently from $\mathcal{N}(0, 0.1)$. Then to replicate the case where we have activations in different channels, a large value is added to randomly selected rows of W_j . Then we set $\boldsymbol{\Sigma}_j = W_j W_j^T + \epsilon \mathbf{I}$, where $\epsilon = 10^{-6}$ is added to the diagonal to ensure the matrix is positive definite. Lastly, independent Gaussian noise maybe added to the final time series data depending on the specific simulation settings.

A.3 Simulating data with multiple dynamics

We need to simulate standard deviations and correlation matrices. This can be easily done by generating the covariance matrices Σ_j as described in Section A.2 and extracting the standard deviations $\mathbf{E}_j$ and correlation matrices $\mathbf{R}_j$ by

$$\mathbf{E}_j = \sqrt{\text{diag}(\Sigma_j)}\mathbf{I}, \quad \mathbf{R}_j = \mathbf{E}_j^{-1}\Sigma_j\mathbf{E}_j^{-1}. \quad (\text{A.5})$$

Then we simulate independent time courses as described in Section A.1 and generate the observed data with standard deviations and correlation matrices depending on different time courses.

A.4 Simulate session-specific covariances

In the case where we want to simulate session-specific covariances, we first generate group-level covariances as described in Section A.2. Each session is associated with an embedding vector, which is assigned a group uniformly randomly from a set of groups. Each embedding vector is generated with a multivariate Gaussian distribution with the same variance but different means depending on the group assignment. Next linear transformations are generated randomly to map the group-level Cholesky vectors to spatial embeddings for each state/mode, which are concatenated with the embedding vectors to give the concatenated embeddings. The session-specific deviations are generated by applying another randomly generated linear transformation to the concatenated embeddings. Finally, the session-specific covariance matrices are generated by adding the session-specific deviations to the group-level covariances.

Appendix B

Supplementary material on multi-dynamic modelling

This contains supplementary information for materials discussed in Chapter 4.

B.1 Simulation results

In this section, we describe how we use simulation to validate M-DyNeMo and investigate potential benefits of multi-dynamic modelling.

B.1.1 Simulated data

We simulate data with the HMM generative model. We pre-specify the number of states $J^1 = J^2 = J$, the transition probability $\mathbf{A}$ and randomly simulate the observation model parameters θ_{obs} , which are then used to simulate the observed data. Two simulation datasets are used:

- **Simulation 1.** Here we simulate data with multiple dynamics using $J =$

5, $N_c = 40$, $T = 25600$. The transition probability matrix is

$$\mathbf{A} = \begin{bmatrix} 0.9 & 0.1 & 0 & 0 & 0 \\ 0 & 0.9 & 0.1 & 0 & 0 \\ 0 & 0 & 0.9 & 0.1 & 0 \\ 0 & 0 & 0 & 0.9 & 0.1 \\ 0.1 & 0 & 0 & 0 & 0.9 \end{bmatrix}. \quad (\text{B.1})$$

- **Simulation 2.** Here data is simulated with the same setting as Simulation 1 except that $\mathbf{E}_j, \mathbf{R}_j$ follow the same underlying temporal dynamics, i.e. they depend on the same underlying Markov chain of states.

B.1.2 Simulation 1: M-DyNeMo performs more accurate inference on data with multiple dynamics

Here we aim to see if M-DyNeMo can perform inference more accurately than the traditional sliding window approach and DyNeMo, given the data has multiple dynamics. We focus on the first simulated data in Section B.1.1 and follow the procedure described in Section 4.6.3 with step size of 1 and window lengths ranging from 4 to 16. We show the DICE coefficients of the inferred time courses in Figure B.1a. We see the optimal window length of 6 samples does not give perfect DICE coefficient. If we look closely and compare the simulated and inferred time courses in Figure B.1b, we observe that sliding window approach does a decent job at inferring relatively long state changes, but fails to infer the transient state changes accurately. This is the reason why we turn to generative models that implicitly learn the time scale from data.

We now train M-DyNeMo and DyNeMo on the same simulated dataset. From Figure B.2a, we see that M-DyNeMo can recover both state time courses accurately while DyNeMo cannot, due to its intrinsic model assumption of shared dynamics.

Specifically, DyNeMo only infers the correlation dynamics. A quantitative measure of the accuracy of inferred time courses measure by DICE coefficient is shown in Figure B.2b, where we see M-DyNeMo achieves perfect inference of both time courses. In this simulation study, the signal from time-varying power is relatively low compared to that from time-varying FC, which is why DyNeMo ignores the power dynamics. Additionally, we can investigate the reconstruction error measured by Riemannian distance between the reconstructed covariances and the ground truth at each time point, shown in Figure B.2c. It is clear that M-DyNeMo has a lower reconstruction error than DyNeMo, indicating M-DyNeMo is a more accurate model of time-varying covariances on this data.

B.1.3 Simulation 2: M-DyNeMo can infer accurately on single dynamics data

It is important to show that the separate dynamics is not an artefact invented by M-DyNeMo. This can be shown by training M-DyNeMo on the second simulated dataset where data were simulated with a single underlying dynamics shared by both standard deviation and correlation. Shown in Figure B.3a, both of the M-DyNeMo inferred mode time courses match the ground truth and the reconstruction error of time-varying covariances is comparable to that of DyNeMo, shown in Figure B.3b.

B.2 More results on real data

Here we show some additional results on real MEG data in Figures B.4, B.5, B.6, B.7, and B.8.

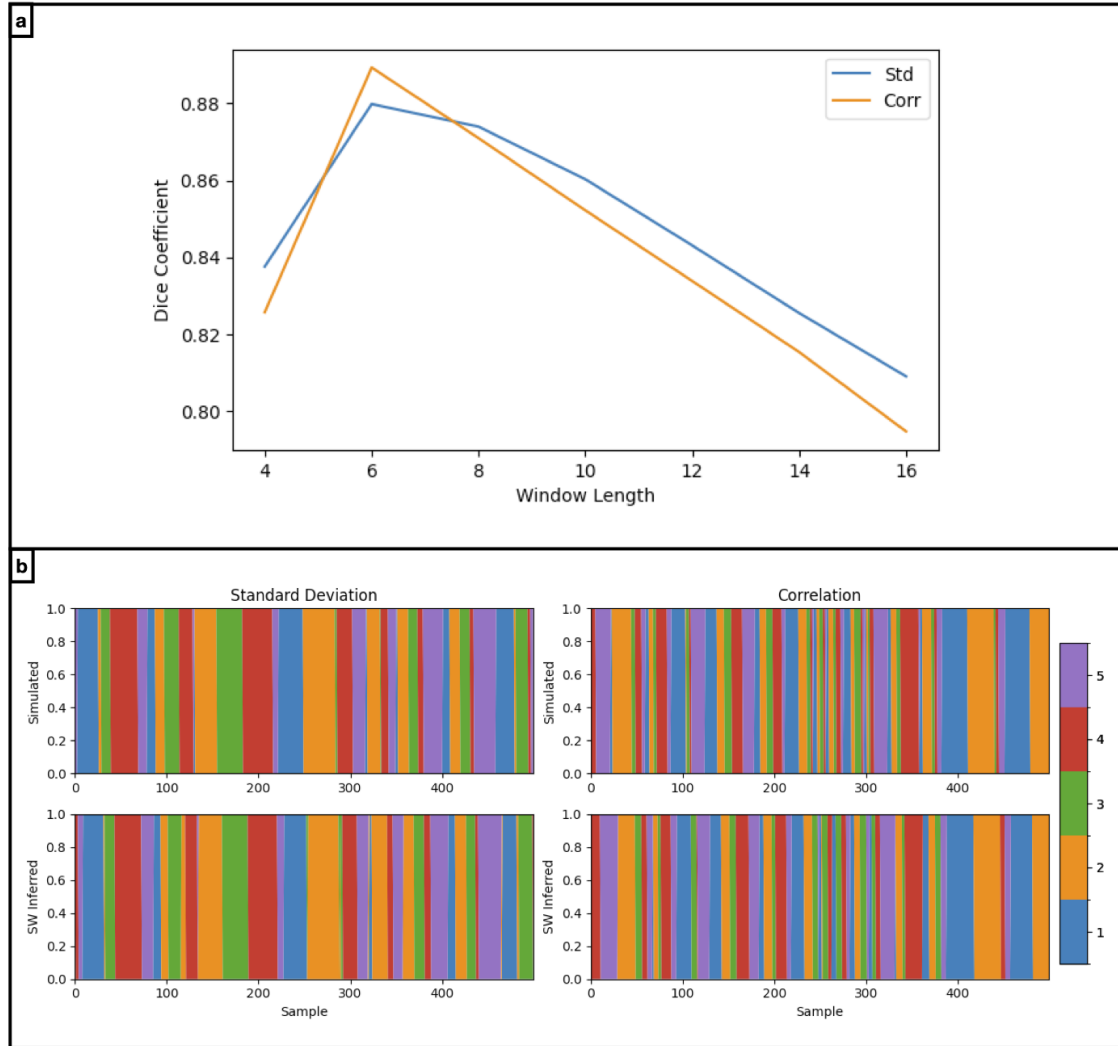


Figure B.1: **Sliding window approach fails to infer ground truth state time courses.** Results on *Simulation 1* dataset. a) Dice coefficient of state time courses inferred by sliding window approach using different window lengths. b) The simulated (top row) and sliding window approach inferred (bottom row, with window size of 6 samples) time courses. A segment of 500 samples containing short state transitions is shown.

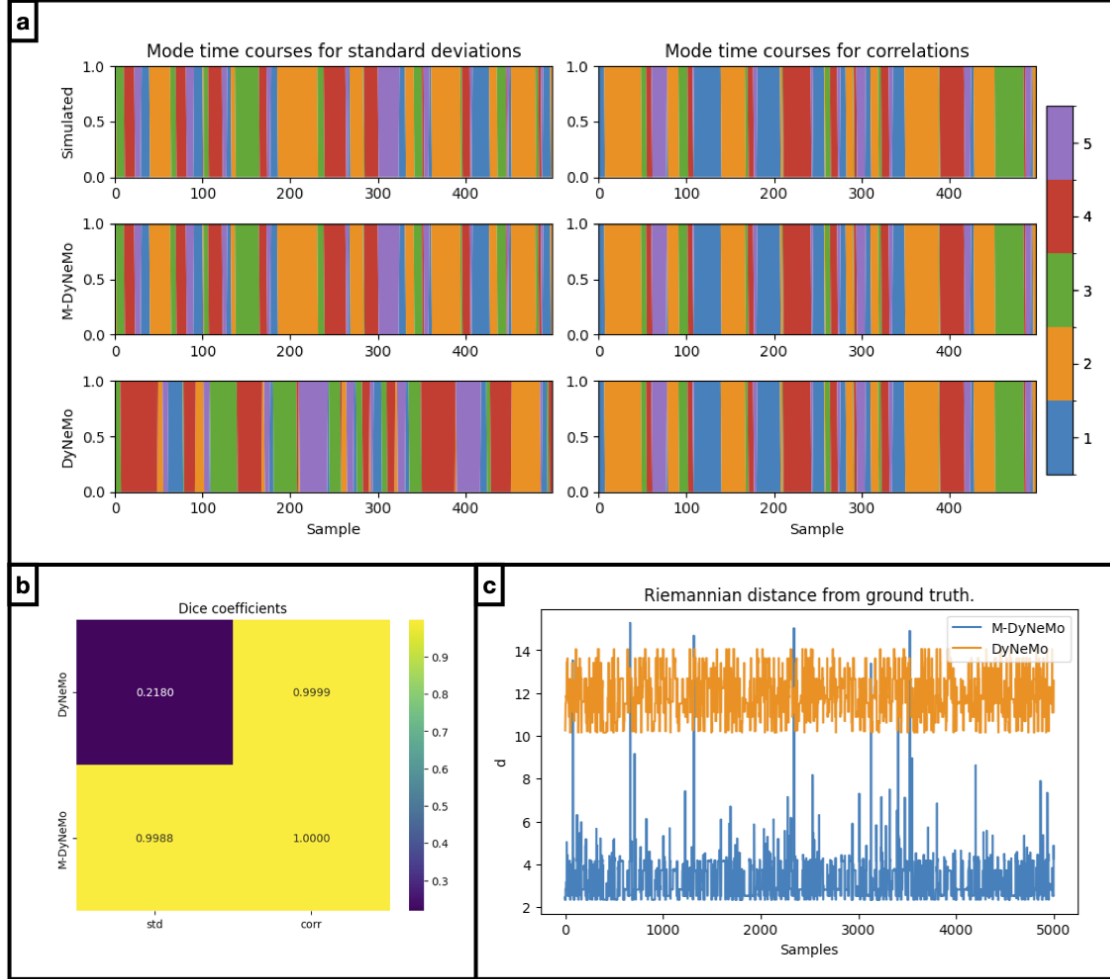


Figure B.2: **M-DyNeMo infers time courses more accurately than DyNeMo when data has multiple dynamics.** *Results on Simulation 1 dataset.* a) The simulated (top row), M-DyNeMo inferred (middle row) and DyNeMo inferred (bottom row) time courses for time-varying standard deviations (left column) and correlations (right column). Here different colours indicate activation of different states and only the first 500 samples are plotted. b) DICE coefficients of DyNeMo and M-DyNeMo inferred time courses compared to the simulated ground truth. c) The reconstruction error (Riemannian distance) of the first 500 samples of time varying covariances is plotted against sample on the x-axis. The reconstruction error of DyNeMo is plotted in orange and that of M-DyNeMo is plotted in blue.

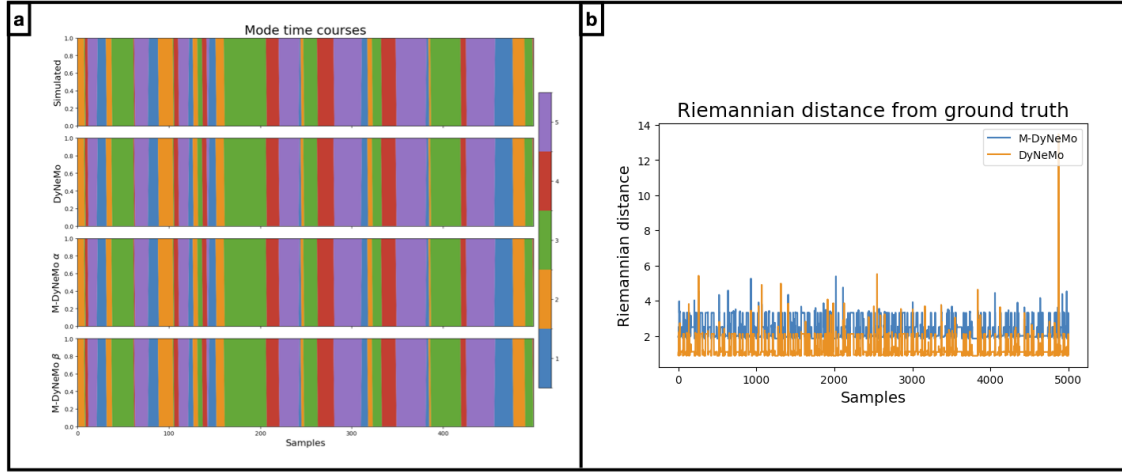


Figure B.3: **M-DyNeMo infers accurately on data with single dynamic.** *Results on Simulation 2 dataset.* a) From top to bottom row are the ground truth simulated, the DyNeMo inferred, and the two “separate” M-DyNeMo inferred mode time courses. Here different colours indicate activation of different states and only the first 500 samples are plotted. b) The reconstruction error (Riemannian distance) of the first 5000 samples of time-varying covariances is plotted against sample on the x-axis. The reconstruction error of DyNeMo is plotted in orange and that of M-DyNeMo is plotted in blue.

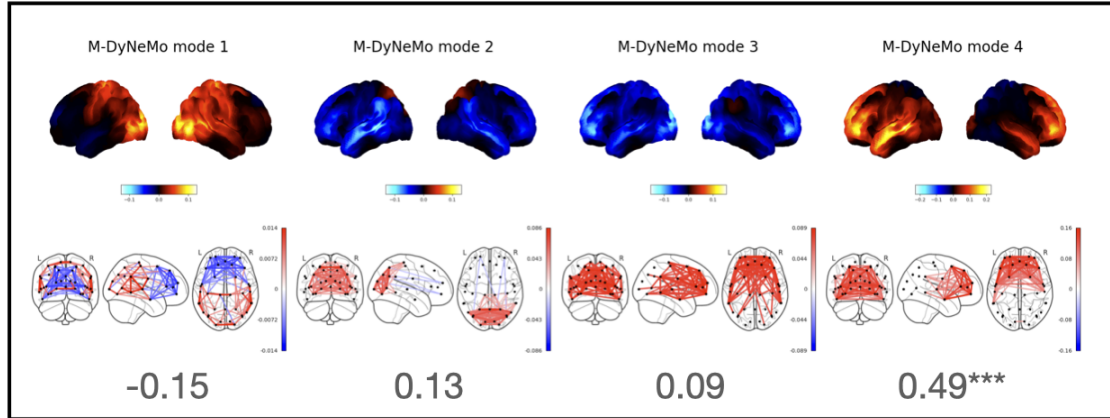


Figure B.4: **Single dynamic DyNeMo is dominated by time-varying power.** *Results from amplitude time courses (1-45Hz) of 38 brain regions obtained from MEGUK-38 resting-state data.* Power (top row) and FC (covariance, bottom row) network maps given by re-calculated networks using M-DyNeMo inferred FC mode time course β . The cosine similarity is also shown for each pair of modes between the re-calculated covariances on M-DyNeMo inferred FC mode time course β and standard DyNeMo inferred covariances.

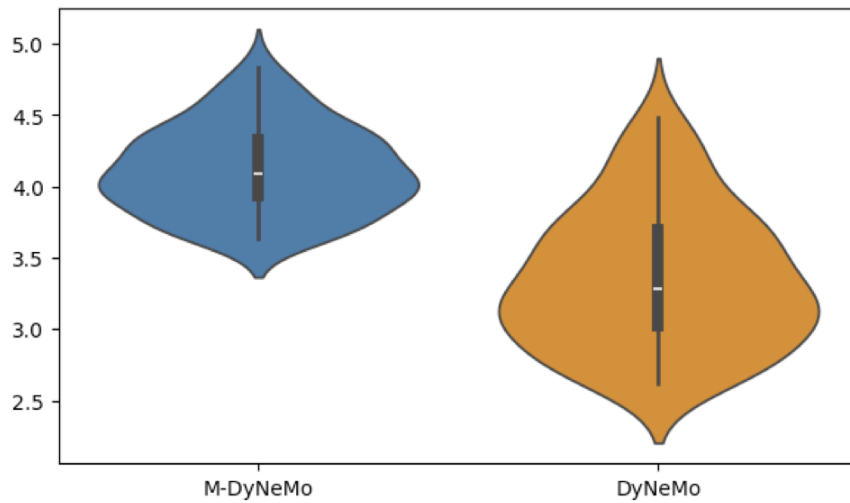


Figure B.5: **M-DyNeMo reconstructed time varying covariances show greater fluctuation over time.** Results from amplitude time courses (1-45Hz) of 38 brain regions obtained from MEGUK-38 resting-state data. For M-DyNeMo and DyNeMo, the y-axis shows the variability of model reconstructed time varying covariances in terms of Riemannian distance of time varying covariances from the mean covariance for each subject.

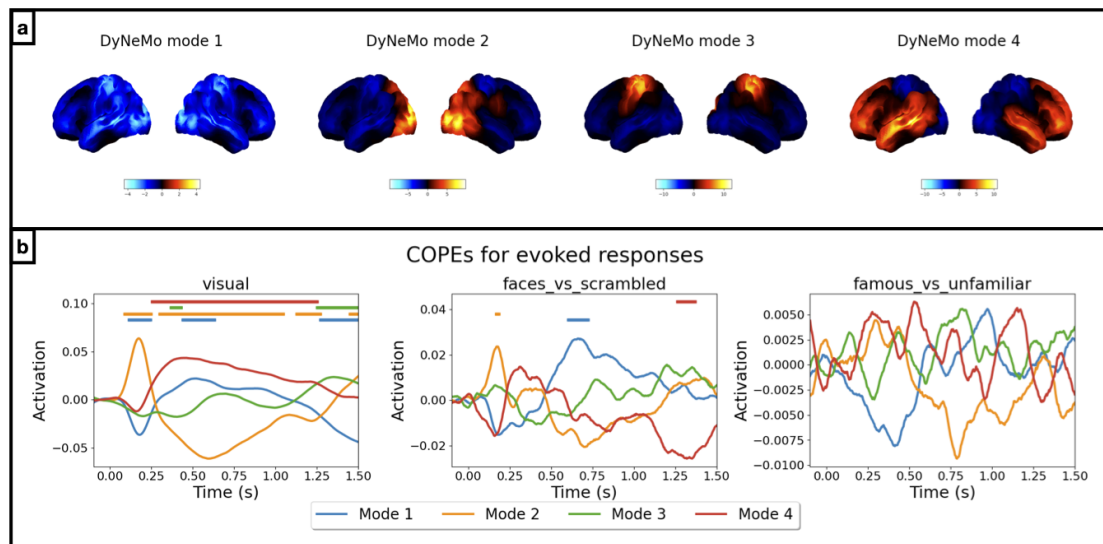


Figure B.6: **Evoked responses given by DyNeMo.** Results from amplitude time courses (1-45Hz) of 38 brain regions obtained from Wakeman-Henson data. a) Power maps for different modes given by DyNeMo. b) Evoked response analysis on DyNeMo-inferred mode time course. Different colours show baseline corrected evoked response (contribution to overall variance) of different modes.

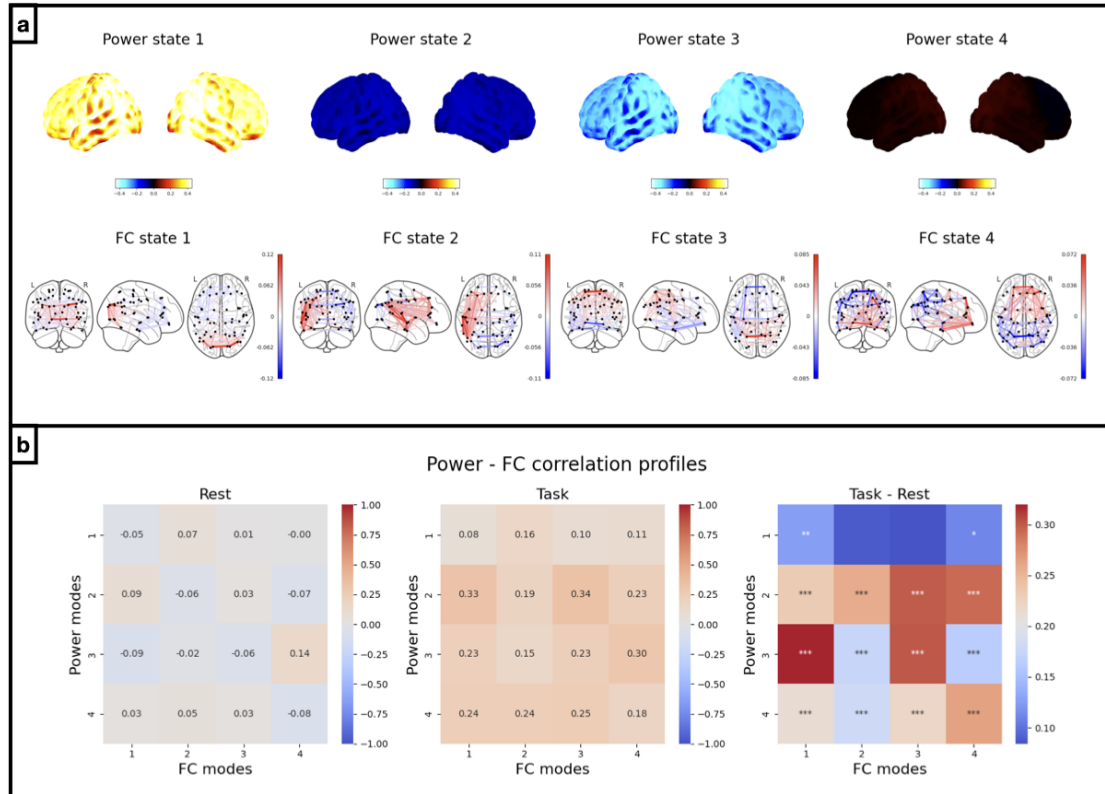


Figure B.7: **Sliding window approach reveals evidence of different correlation profiles during task versus during rest.** *Results from amplitude time courses (1-45Hz) of 52 brain regions obtained from MEGUK-52 data.* a) Power (top row) and FC (bottom row) maps for each mode of the two state time courses. b) Correlations within α (left) and within β (middle) are shown. The right plot shows the differences in correlations during task versus during rest. Correlations that significantly differ from zero are marked with asterisks depending on the p-values.

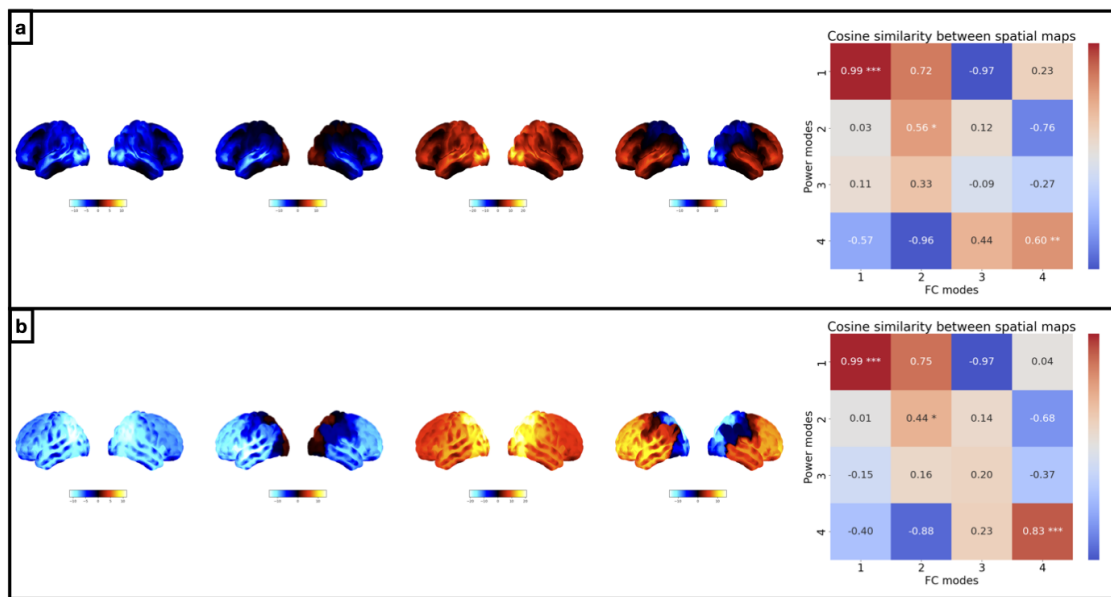


Figure B.8: **Additional FC degree spatial maps.** Results from amplitude time courses (1-45Hz) of 38 brain regions obtained from Wakeman-Henson data and of 52 brain regions obtained from MEGUK-52 data. a) FC degree spatial maps on the Wakeman-Henson dataset (left) and the cosine similarity between power maps and FC degree maps for each pair of power and FC modes (right). b) FC degree spatial maps on the MEGUK-52 dataset (left) and the cosine similarity between power maps and FC degree maps for each pair of power and FC modes (right).

B.3 Model training details

B.3.1 Learning correlation matrices

A square matrix $\mathbf{R} \in \mathbb{R}^{n \times n}$ is a correlation matrix if and only if it is a covariance matrix with diagonal of 1. Similar to learning a covariance matrix, we cannot learn the entries of a correlation matrix directly and need to parameterise the matrix with a vector using a bijection. Similar to learning a covariance matrix, we resort to learning the Cholesky decomposition of a correlation $\mathbf{R} = \mathbf{L}\mathbf{L}^T$, where the k, l element L_{kl} satisfies:

$$L_{kl} = \begin{cases} 0 & \text{if } k < l, \\ \mathbf{M}_{kl}/\mathbf{w}_k & \text{if } k > l, \\ 1/\mathbf{w}_k & \text{if } k = l, \end{cases} \quad (\text{B.2})$$

where $\mathbf{M}_{kl}$ is the k, l entry of a lower triangular matrix $\mathbf{M}$ and $\mathbf{w}$ is a vector such that its k -th element is $\mathbf{w}_k = \sqrt{1 + \sum_{l=0}^{k-1} \mathbf{M}_{kl}^2}$. It is easy to check that $\mathbf{R}$ constructed in this way is a legitimate correlation matrix and during training, we learn the vector of lower triangular entries of $\mathbf{M}$.

B.3.2 Extra tricks for model training

Some extra tricks were used to help fast and stable convergence of M-DyNeMo:

- **KL annealing:** See Section 3.2.3.
- **Multi-start initialisation:** See Section 3.2.3.
- **Learning rate scheduling:** A learning rate scheduler is used to exponentially reduce the learning rate as the training progresses. This helps the model to take large steps in the beginning for faster convergence and smaller steps towards the end for fine-tuning. In practice, we only start decreasing the learning rate after the KL annealing phase.

- **Initialisation for $\mathbf{R}_j$:** We find in practice that without a “good” initialisation of the correlation matrices, the convergence of the model is extremely slow. To help with this, we initialise the correlation matrices with “realistic” ones obtained from the data. Specifically, we use a sliding window approach to calculate the time-varying correlation matrices and used K-means clustering to identify J^2 clusters of time-varying correlation matrices. We then initialise the correlation matrices with the cluster centroids. We find that this helps the model to converge to a lower loss consistently.

B.4 Post-hoc analysis

B.4.1 Re-normalising mode time courses

As mentioned in [Gohil et al., 2022], the inferred mode time courses do not account for the difference in the relative magnitude of each of the spatial modes. Intuitively, if the covariance matrix for a mode is large in the diagonal, fluctuation in its mixing coefficient will have a large impact on the overall variance. For α , the raw mode time course is multiplied by the mean of the square of the diagonal elements of $\mathbf{E}_j$ before normalised by dividing the sum over modes at each time point. Similarly for β , the raw mode time course is multiplied by the mean of absolute off-diagonal elements of $\mathbf{R}_j$ and normalised.

B.4.2 Maximum statistic permutation test for spatial map similarity

The underlying assumption of this test is that sequences of mode time courses are exchangeable under the null hypothesis and we used the maximum statistic approach to correct for multiple comparison. The procedure for generating samples from the null distribution is as follows:

Hyper-parameters	Simulation 1, 2	MEG-38	Wakeman-Henson	MEG-52
Number of modes	5	4	4	4
Sequence length	200	200	200	200
Inference RNN hidden units	64	64	64	64
Model RNN hidden units	64	64	64	64
KL annealing sharpness	10	10	10	10
KL annealing epochs	40	40	20	20
Training epochs	80	80	40	40
Batch size	32	256	256	256
Learning rate	1e-2	1e-3	1e-3	1e-3
Learning rate decay factor	0.1	0.1	0.1	0.1
Number of multi-starts	5	10	10	10
Multi-start epochs	10	5	5	5

Table B.1: Model specifications of M-DyNeMo on different datasets.

- Separate mode time courses into sequences of 1 second.
- Shuffle the sequences.
- Regress TV variances/correlations/covariances on the shuffled mode time courses and get regressed spatial maps.
- Match the spatial maps with cosine similarity using the Hungarian algorithm ([Kuhn, 1955]).
- Get the maximum cosine similarity across pairs of modes (see Section 2.3.4).

The p-value of the cosine similarities of each pair of modes is calculated as the percentage of samples from the null distribution that exceeds the observed cosine similarity. In this paper, tests with p-value lower than 0.05, 0.01, and 0.001 are annotated with one, two, and three asterisks respectively.

B.5 Model specifications

Here we present the hyper-parameters used in training M-DyNeMo (Table B.1) and DyNeMo (Table B.2) on different datasets.

Hyper-parameters	Simulation 1, 2	MEG-38	Wakeman-Henson
Number of modes	5	4	4
Sequence length	200	200	200
Inference RNN hidden units	64	64	64
Model RNN hidden units	64	64	64
KL annealing sharpness	10	10	10
KL annealing epochs	40	20	20
Training epochs	80	40	40
Batch size	32	256	256
Learning rate	1e-2	5e-3	1e-3
Learning rate decay factor	0.1	0.1	0.1
Number of multi-starts	5	5	5
Multi-start epochs	10	2	2

Table B.2: Model specifications of DyNeMo on different datasets.

Appendix C

Supplementary material on modelling variability

This contains supplementary information for materials discussed in Chapter 4.

C.1 Dual estimation in HMM

Let γ_{jt} be the state probability of state j at time t , then the session-specific means $\{\hat{\boldsymbol{\mu}}_j\}_{j=1}^J$ and covariances $\{\hat{\boldsymbol{\Sigma}}_j\}_{j=1}^J$ are estimated by

$$\begin{aligned}\hat{\boldsymbol{\mu}}_j &= \frac{\sum_{t=1}^T \gamma_{jt} \mathbf{x}_t}{\sum_{t=1}^T \gamma_{jt}}, \\ \hat{\boldsymbol{\Sigma}}_j &= \frac{\sum_{t=1}^T \gamma_{jt} (\mathbf{x}_t - \hat{\boldsymbol{\mu}}_j)(\mathbf{x}_t - \hat{\boldsymbol{\mu}}_j)^T}{\sum_{t=1}^T \gamma_{jt}}.\end{aligned}\tag{C.1}$$

These expressions are derived by differentiating (3.4) with respect to the observation model parameters.

C.2 Outlier in the dataset

We see from Figure C.1 that run 3 of subject 8 has a very different static PSD profile from the other runs. In particular the spike at around 26 Hz. Furthermore it has

a very different session-pairwise correlation of the static covariances from the other runs.

C.3 Possible improvements on algorithm for choosing embedding dimensions

In practice, we can also use other ways to choose n_ω . One may notice that the procedure described in Algorithm 2 is flawed in the sense that it requires a “sensible” choice of candidate embedding dimensions set. Imagine if we choose a set where the elements only differ by 1, then the hypothesis tests are highly likely not rejected and we will end up always choosing the smaller embedding dimension.

C.3.1 Improving the procedure

A more robust way is the described in Algorithm 3. The idea is to keep testing with larger embedding dimensions until we reach the largest one and update the chosen embedding dimension whenever the null hypothesis is rejected. This way we can ensure that we are not missing out on larger embedding dimensions that could potentially give better performance.

Algorithm 3: Choosing the embedding dimension

Input: Embedding dimension candidates $\{n_\omega^1, \dots, n_\omega^M\}$

Output: Chosen embedding dimension n_ω^*

Procedure: $k \leftarrow 1$, $n_\omega^* \leftarrow n_\omega^1$

while $k < M$ **do**

Test the null hypothesis that the training loss of model with n_ω^{k+1} is not significantly lower than that with n_ω^*

if *Reject the null hypothesis* **then**

$n_\omega^* \leftarrow n_\omega^{k+1}$

$k \leftarrow k + 1$

return n_ω^*

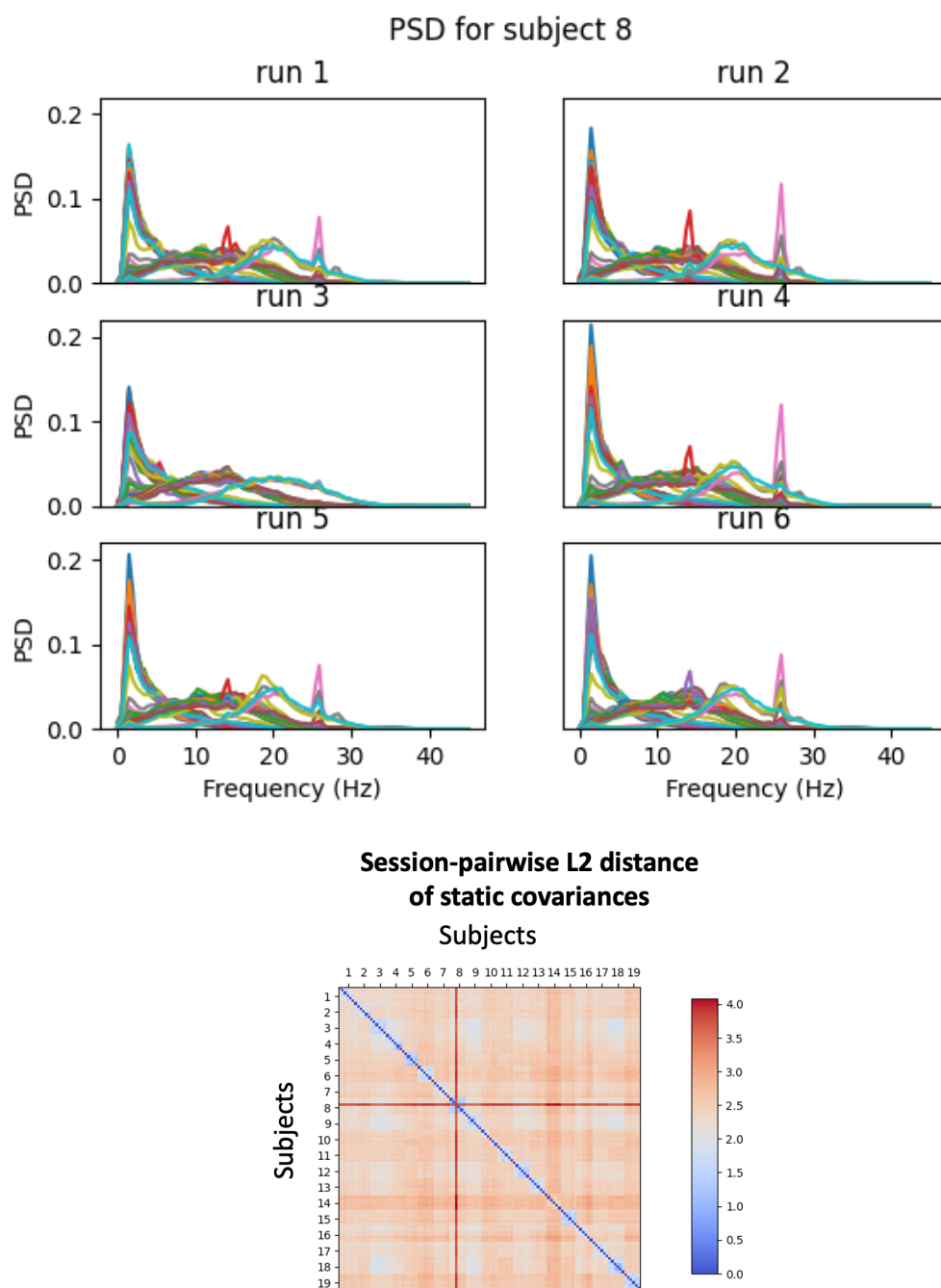


Figure C.1: **Outlier in the Wakeman-Henson dataset.** Top: PSD for the prepared data used for training is plotted for different runs of subject 8. Bottom: The session-pairwise L2 distance of the static covariances is plotted for the prepared data used for training. The outlier session is highlighted by exceptionally high distance with all other sessions.

Hyper-parameters	Simulation 1	Simulation 2	Simulation 3
Number of modes	5	5	5
Sequence length	200	200	200
Training epochs	40	40	40
Batch size	32	64	64
Learning rate	1e-2	5e-3	5e-3
Learning rate decay factor	0.05	0.05	0.05
Number of multi-starts	10	5	5
Multi-start epochs	5	3	5

Table C.1: Model specifications for training HMM on simulated datasets.

Hyper-parameters	Wakeman-Henson	Combined dataset	Cam-CAN
Number of modes	6	6	8
Sequence length	2000	2000	2000
Training epochs	20	20	10
Batch size	32	32	32
Learning rate	1e-2	1e-2	1e-3
Learning rate decay factor	0.1	0.1	0.1
Number of multi-starts	3	3	3
Multi-start epochs	1	1	1

Table C.2: Model specifications for training HMM on real datasets.

C.3.2 The one standard deviation rule

The one standard deviation rule is a simple and popular heuristic for choosing model complexity (See for example [Hastie et al., 2009] Page 244). The idea is to choose the most parsimonious model that is within one standard deviation of the model giving lowest loss.

C.4 Model specifications

Here we present the model specifications of HMM in Tables C.1, C.2 and of HIVE in Tables C.3, C.4 when training on different datasets.

Hyper-parameters	Simulation 1	Simulation 2	Simulation 3
Number of modes	5	5	5
Sequence length	200	200	200
Embedding dimension	2	2	10
Spatial embedding dimension	2	2	2
Decoder number of layers	5	5	5
Decoder number of units	32	32	32
Decoder activation	tanh	tanh	tanh
Decoder regularisation	L1	L1	L1
Decoder regularisation strength	10	10	0.1
Annealing sharpness	10	10	10
Annealing epochs	20	20	20
Training epochs	40	40	40
Batch size	32	128	64
Learning rate	5e-3	5e-3	5e-3
Learning rate decay factor	0.05	0.05	0.05
Number of multi-starts	10	10	5
Multi-start epochs	5	5	5

Table C.3: Model specifications for training HIVE on simulated datasets.

Hyper-parameters	Wakeman-Henson	Combined dataset	Cam-CAN
Number of modes	6	6	8
Sequence length	200	200	200
Embedding dimension	10	20	50
Spatial embedding dimension	2	2	2
Decoder number of layers	5	5	5
Decoder number of units	32	32	32
Decoder activation	tanh	tanh	tanh
Decoder regularisation	L1	L1	L1
Decoder regularisation strength	10	10	10
Annealing sharpness	10	10	10
Annealing epochs	15	20	15
Training epochs	30	40	30
Batch size	128	128	128
Learning rate	5e-3	5e-3	2.5e-3
Learning rate decay factor	0.1	0.1	0.1
Number of multi-starts	10	10	10
Multi-start epochs	2	2	1

Table C.4: Model specifications for training HIVE on real datasets.

Appendix D

DyNeMo with Integrated Variability Estimation

In Chapter 5, we proposed embedding vectors as a means of modelling variability between sessions. We applied this idea to the HMM as an example, giving us HIVE. The core building block in HIVE - the variability encoding block can also be applied to DyNeMo (Section 3.2, [Gohil et al., 2022]). We call this model **DIVE** (DyNeMo with Integrated Variability Estimation). Conceptually, similar to HIVE, the variability encoding block generates session-specific means and covariances, which form the basis set of networks and are linearly mixed with the mode time courses.

The generative model of DIVE differs from that of DyNeMo in the same way as HIVE differs from HMM. Formally in DIVE

$$\begin{aligned} \mathbf{x}_t^i &\sim \mathcal{N}(\mathbf{m}_t^i, \mathbf{C}_t^i), \\ \mathbf{m}_t^i &= \sum_{j=1}^J \alpha_{jt}^i \boldsymbol{\mu}_j^i, \quad \mathbf{C}_t^i = \sum_{j=1}^J \alpha_{jt}^i \boldsymbol{\Sigma}_j^i, \quad \forall t \in [T], i \in [N], \end{aligned} \tag{D.1}$$

where $\boldsymbol{\mu}_j^i, \boldsymbol{\Sigma}_j^i$ are the observation model parameters for session i generated by the variability encoding block and α_{jt}^i is the mixing coefficient generated by the same RNN across sessions. Figure D.1 summarises the generative model of DIVE and

compares it with that of HIVE. They only differ in the temporal model.

D.1 Training DIVE

The training of DIVE does not involve the iterative process used in training HIVE. Instead, we can derive from first principles the variational free energy of the model and use gradient-based optimisation to train the model, which is similar to the training of DyNeMo and VAE in general. Similar to the models discussed in this thesis so far, time series data are separated into sequences and ensure that data in each sequence n belong to the same session i_n . Let $\boldsymbol{\phi}$ be the model parameters that include $\{\boldsymbol{\mu}_j, \mathbf{D}_j : j \in [J]\}$, the weights of the RNN, and of the variability encoding block, and let $\boldsymbol{\psi}$ be the variational parameters that include the weights of bidirectional RNN, and the parameters of the variational distribution of the deviation magnitudes $\{a_j^i, b_j^i : i \in [N], j \in [J]\}$, then the loss for this sequence is given by:

$$\begin{aligned} \mathcal{L}_n(\boldsymbol{\phi}, \boldsymbol{\psi}) = & - \underbrace{\mathbb{E}_{q_{\boldsymbol{\psi}}(\boldsymbol{\theta}^{i_n}, \mathbf{d}^{i_n})}[\log p_{\boldsymbol{\phi}}(\mathbf{x}^{i_n} | \boldsymbol{\theta}^{i_n}, \mathbf{d}^{i_n})]}_{\text{LL}} \\ & + \underbrace{D_{KL}(q_{\boldsymbol{\psi}}(\boldsymbol{\theta}^{i_n}) || p_{\boldsymbol{\phi}}(\boldsymbol{\theta}^{i_n})) + \frac{1}{N_{seq}} D_{KL}(q_{\boldsymbol{\psi}}(\mathbf{d}^{i_n}) || p_{\boldsymbol{\phi}}(\mathbf{d}^{i_n}))}_{\text{KL}}, \end{aligned} \quad (\text{D.2})$$

which is effectively the loss function for DyNeMo with an extra term for the KL divergence of the variational distribution of the deviation magnitude. During training, the reparameterisation trick is used to sample from the variational distribution and pass gradient through the sampling process. The KL annealing technique described in Section 5.5.4 is also used in training DIVE.

D.2 Simulation results

In this section we show a simple simulation result to validate DIVE. We simulate mode time courses and session-specific covariances for 100 sessions (See Sections A.1,

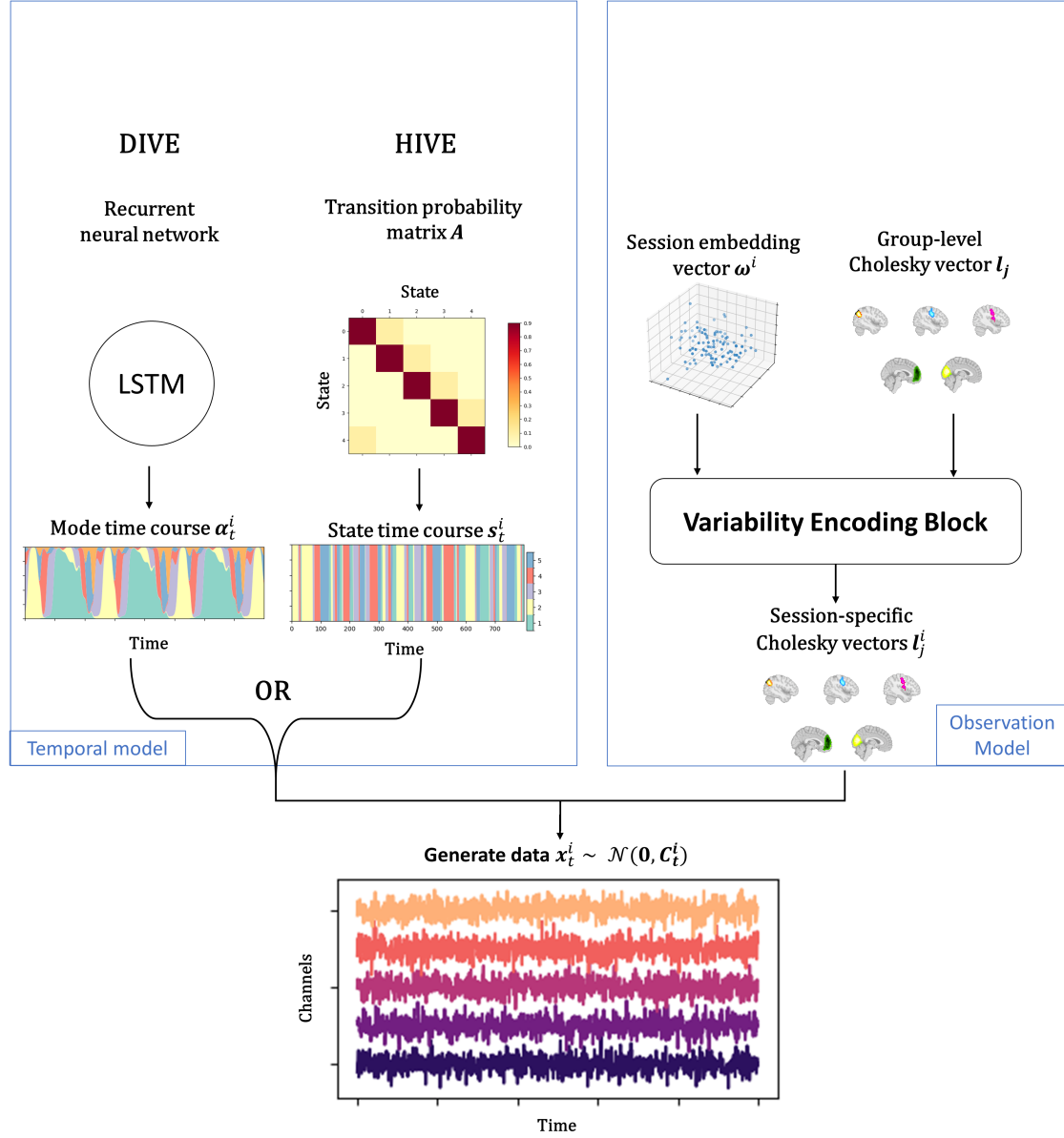


Figure D.1: **Generative model of DIVE vs HIVE.** The generative model of DIVE only differs from that of HIVE in the temporal model. Time courses are generated from either an RNN or a Markov process, which is then used to generate the time-varying observation model parameters for each session.

A.4). Each session contains data with 3000 samples and Figure D.2 shows DIVE is able to accurately infer both the mode time courses and session embedding vectors.

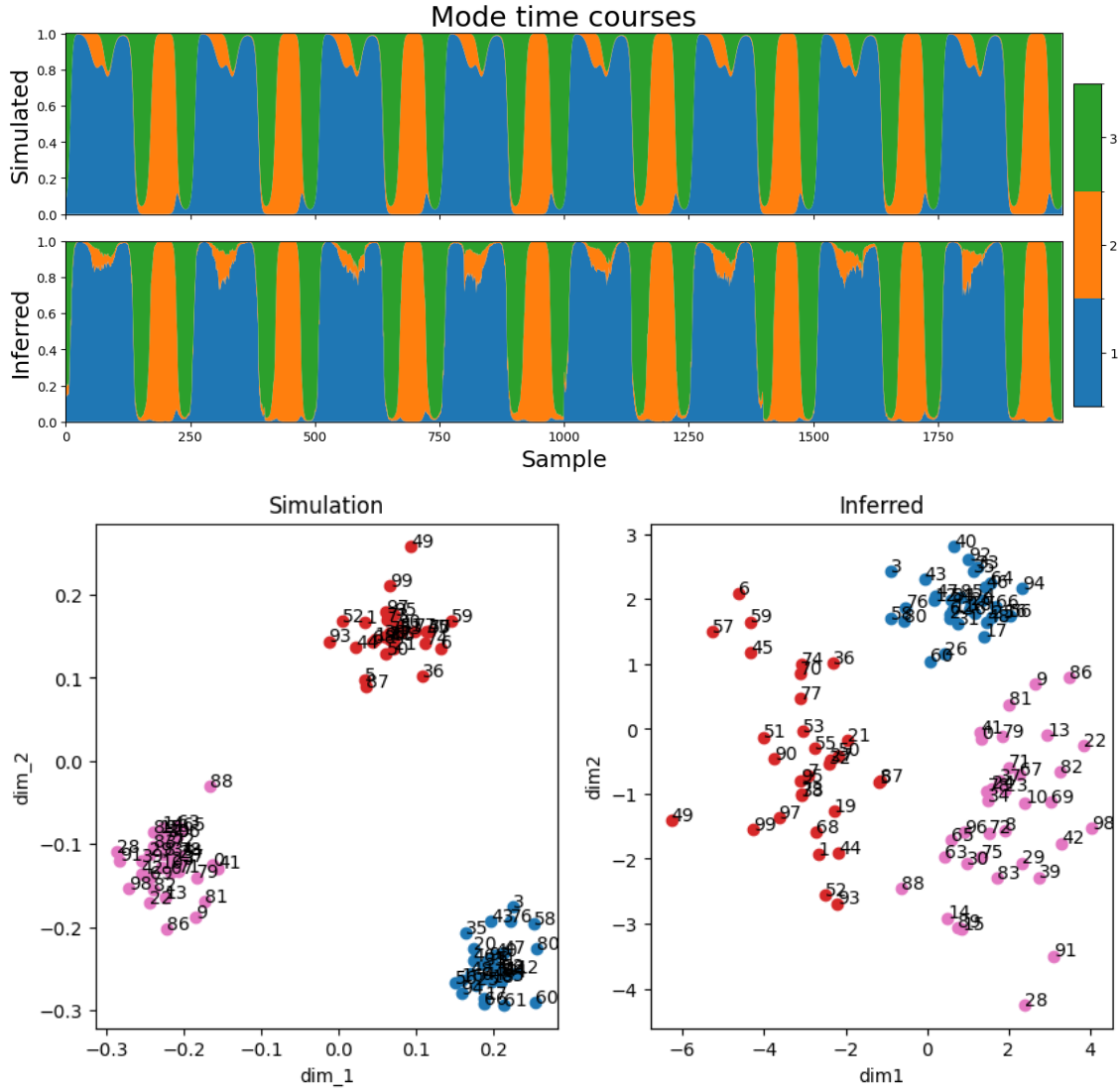


Figure D.2: **Simulation results of DIVE.** Top: Simulated and inferred mode time courses. Different colours show the mode time courses for different modes. Bottom: Simulated and inferred embedding vectors. Here the points are coloured with the ground truth group assignment and are annotated with the session ID.

Appendix E

Modelling temporal dynamics with Gaussian Process

DyNeMo attempts capture long range temporal dependency with an RNN. However in DyNeMo,

- Data is separated into smaller sequences which are fed to the RNNs. Hence longer temporal dependency (say more than 200 timestamps) cannot be captured.
- There are many hyperparameters related to the RNNs, such as the number of units, number of layers, dropout rate, etc.
- RNNs are black-boxes and it is hard to directly interpret the RNN weights.

The above motivates using Gaussian processes to model the temporal dynamics of the hidden logits $\theta_{1:T}$ instead of RNNs. A Gaussian process can be thought of as a infinite dimensional multivariate distribution.

Formally, it is characterised by a mean function $m : \mathbb{R} \rightarrow \mathbb{R}$ and a covariance function (*kernel*) $k : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$. Let $\mathcal{T}$ be an index set (which can be uncountably infinite), a real-valued stochastic process $f \sim GP(m, k)$ is a Gaussian process if for

every finite subset $\mathbf{t} = \{t_1, \dots, t_k\} \subseteq \mathcal{T}$,

$$\mathbf{f} := f(\mathbf{t}) = (f(t_1), \dots, f(t_k)) \sim \mathcal{N}(\mathbf{m}, \mathbf{K}),$$

where $\mathbf{m}_i = m(t_i)$, $\mathbf{K}_{ij} = k(t_i, t_j)$.

E.1 Generative model

Instead of RNNs, we assume the hidden logits follow Gaussian processes. For simplification, we assume that for each mode $j \in [J]$, $\theta^j \sim GP(m^j, k^j)$ independently. Then evaluated on the index set $[T]$,

$$\theta_{1:T}^j = (\theta^j(t), \dots, \theta^j(T))^T \sim \mathcal{N}(\mathbf{m}_{1:T}^j, \mathbf{K}_{1:T, 1:T}^j), \quad (\text{E.1})$$

and it is easy to show with Gaussian conditioning that

$$\theta_t^j | \theta_{1:t-1}^j \sim \mathcal{N}(\mu_t^j, \sigma_t^{j^2}), \quad (\text{E.2})$$

where

$$\begin{aligned} \mu_t^j &= \mathbf{m}_t^j + \mathbf{K}_{t, 1:t-1}^j (\mathbf{K}_{1:t-1, 1:t-1}^j)^{-1} (\theta_{1:t-1}^j - \mathbf{m}_{1:t-1}^j) \\ \sigma_t^{j^2} &= \mathbf{K}_{t,t}^j - \mathbf{K}_{t, 1:t-1}^j (\mathbf{K}_{1:t-1, 1:t-1}^j)^{-1} \mathbf{K}_{1:t-1, t}^j. \end{aligned}$$

Normally the mean functions are fixed to be zero functions and there are several choices for the kernel, which determines the relationships between values at different time points. A common family of kernels is the RBF kernel:

$$k(t, t') = \exp \left(-\frac{(t - t')^2}{2\gamma^2} \right), \quad (\text{E.3})$$

where γ is called the length scale. Notice with this kernel:

- If two time points are closer together, their covariance is smaller.

- γ controls the rate of decay of covariance. A larger γ leads to slower decay and longer temporal dependency.
- γ controls the “global” behaviour of the covariance function, and hence is less flexible than RNNs, which provide local estimates.

Finally, similar to HMM and DyNeMo, data are assumed to be generated through a multivariate Gaussian distribution conditioning on the hidden logits.

E.2 Inference

We would want to infer the posterior distribution of the hidden logits $p(\theta_{1:T}^{1:J} | \mathbf{x}_{1:T}) \propto p(\mathbf{x}_{1:T} | \theta_{1:T}^{1:J}) p(\theta_{1:T}^{1:J})$. Due to the fact that the likelihood of our data conditioned on the logits is not Gaussian, we need to use variational inference to approximate this posterior with $q(\theta_{1:T}^{1:J})$, which is assumed to be Gaussian itself and parameters to be learned. See Variational Gaussian Process (VGP, [Titsias, 2009]) for more details. However, this approach involves the inversion of the matrix $\mathbf{K}_{1:T,1:T}^j$, which requires $\mathcal{O}(T^3)$ computational cost and $\mathcal{O}(T^2)$ memory and is infeasible for large T . One of the solutions is the inducing point approximation. Intuitively, it summarises the data points with a set of pseudo data points, which is much smaller in size. The resulting model is the Sparse Variational Gaussian Process (SVGP, [Titsias, 2009]).

Formally, suppose $\mathbf{z}_{1:m}^j$ is a vector of length $m \ll T$, that lives in the domain of our Gaussian process. Let $\mathbf{u}_{1:m}^j = \theta^j(\mathbf{z}_{1:m}^j)$, called the *inducing points*, then the idea is to consider instead the variational distribution for the joint posterior $p(\theta_{1:T}^{1:J}, \mathbf{u}_{1:m}^{1:J} | \mathbf{x}_{1:T})$ in the form $q(\theta_{1:T}^{1:J}, \mathbf{u}_{1:m}^{1:J}) = p(\theta_{1:T}^{1:J} | \mathbf{u}_{1:m}^{1:J}) q(\mathbf{u}_{1:m}^{1:J})$ where

$$q(\mathbf{u}_{1:m}^{1:J}) = \prod_{j=1}^J q(\mathbf{u}_{1:m}^j) = \prod_{j=1}^J \mathcal{N}(\mathbf{u}_{1:m}^j | \boldsymbol{\mu}_u^j, \boldsymbol{\Sigma}_u^j). \quad (\text{E.4})$$

Here $\boldsymbol{\mu}_u^j, \boldsymbol{\Sigma}_u^j$ are the learnable variational parameters. The objective function is the usual ELBO/free energy, i.e.

$$\begin{aligned}
\text{ELBO} &= \mathbb{E}_{q(\theta_{1:T}^{1:J}, \mathbf{u}_{1:m}^{1:J})} [\log p(\mathbf{x}_{1:T} | \theta_{1:T}^{1:J}, \mathbf{u}_{1:m}^{1:J})] - D_{KL}(q(\theta_{1:T}^{1:J}, \mathbf{u}_{1:m}^{1:J}) || p(\theta_{1:T}^{1:J}, \mathbf{u}_{1:m}^{1:J})) \\
&= \mathbb{E}_{q(\theta_{1:T}^{1:J})} [\log p(\mathbf{x}_{1:T} | \theta_{1:T}^{1:J})] - D_{KL}(q(\theta_{1:T}^{1:J}, \mathbf{u}_{1:m}^{1:J}) || p(\theta_{1:T}^{1:J}, \mathbf{u}_{1:m}^{1:J})) \\
&= \mathbb{E}_{q(\theta_{1:T}^{1:J})} [\log p(\mathbf{x}_{1:T} | \theta_{1:T}^{1:J})] - D_{KL}(q(\mathbf{u}_{1:m}^{1:J}) || p(\mathbf{u}_{1:m}^{1:J})),
\end{aligned} \tag{E.5}$$

where $q(\theta_{1:T}^{1:J}) = \int q(\theta_{1:T}^{1:J}, \mathbf{u}_{1:m}^{1:J}) d\mathbf{u}_{1:m}^{1:J} = \int p(\theta_{1:T}^{1:J} | \mathbf{u}_{1:m}^{1:J}) q(\mathbf{u}_{1:m}^{1:J}) d\mathbf{u}_{1:m}^{1:J}$. Since we have a Gaussian process, and the variational distribution on the inducing points is also Gaussian, we get (with Gaussian conditioning)

$$q(\theta_{1:T}^j) = \mathcal{N}(\theta_{1:T}^j | \mathbf{A}^j \boldsymbol{\mu}_u^j, \mathbf{A}^j \boldsymbol{\Sigma}_u^j (\mathbf{A}^j)^T + \mathbf{B}^j), \tag{E.6}$$

where

$$\begin{aligned}
\mathbf{A}^j &= \mathbf{K}_{1:T, 1:m}^j (\mathbf{K}_{1:m, 1:m}^j)^{-1} \\
\mathbf{B}^j &= \mathbf{K}_{1:T, 1:T}^j - \mathbf{K}_{1:T, 1:m}^j (\mathbf{K}_{1:m, 1:m}^j)^{-1} \mathbf{K}_{1:m, 1:T}^j.
\end{aligned} \tag{E.7}$$

Notice now the expression only depends on the inverse of a much smaller matrix $\mathbf{K}_{1:m, 1:m}^j$. To estimate the expectation of the log-likelihood with respect to the variational distribution, we can use the reparameterisation trick mentioned before. Another popular approach involves the use of Gauss-Hermite quadrature ([Liu and Pierce, 1994]), which is the approach implemented in *tensorflow-probability*.

Finally, to get our initial goal, the posterior distribution $p(\theta_{1:T}^{1:J} | \mathbf{x}_{1:T})$, notice

$$\begin{aligned}
p(\theta_{1:T}^{1:J} | \mathbf{x}_{1:T}) &= \int p(\theta_{1:T}^{1:J}, \mathbf{u}_{1:m}^{1:J} | \mathbf{x}_{1:T}) d\mathbf{u}_{1:m}^{1:J} \\
&\approx \int q(\theta_{1:T}^{1:J}, \mathbf{u}_{1:m}^{1:J}) d\mathbf{u}_{1:m}^{1:J} \\
&= q(\theta_{1:T}^{1:J}).
\end{aligned} \tag{E.8}$$

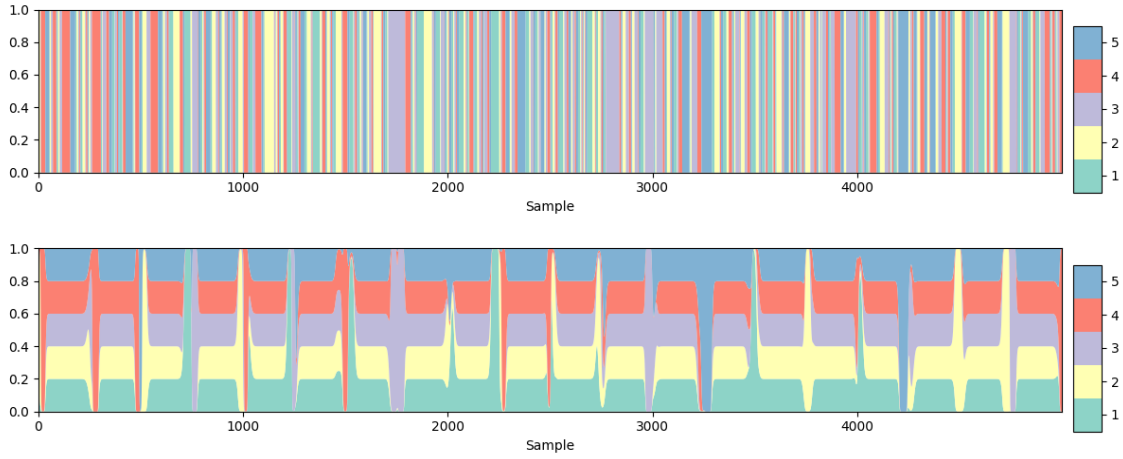


Figure E.1: **DyNeMo with Gaussian Process fails to recover the time courses.** Simulated state time courses (top) and inferred mode time courses (bottom) are plotted. Different colours show activation of different modes

E.3 Simulation result

Here we train this model on HMM generated data with $J = 5, T = 25,600$. Shown in Figure E.1 are the simulated state time courses and inferred mode time courses from DyNeMo with Gaussian Process. We see that although the activations in the inferred mode time courses correspond to state activations in the ground truth state time courses, the proposed model is not able to recover the ground truth accurately.

E.4 Discussion

Although this model is non-parametric in nature and can theoretically infer any temporal dynamics, the use of inducing points still hinders its expressiveness. An exploratory simulation study on data simulated with the HMM generative model, shows that the number of state transitions allowed is constrained by the number of inducing points. Though it could be argued that this model might work well in the case of soft mixing mode time courses where the changes of temporal dynamics are less binarised. It was decided this direction was not worth exploring until there is a good use case for it (for example, in data where very long range dependency

is crucial and the temporal dynamics is smoother). Possible extensions to look at include deep Gaussian processes ([Damianou and Lawrence, 2013]), where multiple layers of SVGPs are stacked in the same manner as dense layers in MLPs to improve the expressiveness of the model.