Combined Decision Making with Multiple Agents

Thesis submitted for the degree
Doctor of Philosophy

Edwin Simpson

Hertford College

Supervisor: Prof. Stephen J. Roberts

Pattern Analysis and Machine Learning Research Group
Department of Engineering Science

University of Oxford

Michaelmas Term – July 8, 2014

Abstract

In a wide range of applications, decisions must be made by combining information from multiple agents with varying levels of trust and expertise. For example, citizen science involves large numbers of human volunteers with differing skills, while disaster management requires aggregating information from multiple people and devices to make timely decisions. This thesis introduces efficient and scalable Bayesian inference for decision combination, allowing us to fuse the responses of multiple agents in large, real-world problems and account for the agents' unreliability in a principled manner.

As the behaviour of individual agents can change significantly, for example if agents move in a physical space or learn to perform an analysis task, this work proposes a novel combination method that accounts for these time variations in a fully Bayesian manner using a dynamic generalised linear model. This approach can also be used to augment agents' responses with continuous feature data, thus permitting decision-making when agents' responses are in limited supply.

Working with information inferred using the proposed Bayesian techniques, an information-theoretic approach is developed for choosing optimal pairs of tasks and agents. This approach is demonstrated by an algorithm that maintains a trustworthy pool of workers and enables efficient learning by selecting informative tasks.

The novel methods developed here are compared theoretically and empirically to a range of existing decision combination methods, using both simulated and real data. The results show that the methodology proposed in this thesis improves accuracy and computational efficiency over alternative approaches, and allows for insights to be determined into the behavioural groupings of agents.

Acknowledgements

Firstly, I would like to thank Professor Stephen Roberts, my supervisor, for all his advice, help of various kinds and the freedom to pursue my own ideas. I must acknowledge the Engineering and Physical Sciences Research Council (EPSRC) and the Department of Engineering Science for funding my DPhil, and am also very grateful to the ORCHID project for their contributions to travel and providing an excellent forum for sharing and developing ideas.

Huge thanks go to Arfon Smith and Chris Lintott, of the Zooniverse project, who contributed datasets for testing the ideas in this thesis. Many thanks also go to Steve Reece for great discussions, collaboration and the occasional beer, and also to Gopal Ramchurn and Antonio Penta, with whom we built the TREC crowdsourcing apparatus. Also to Ioannis Psorakis for developing the community analysis method applied in this work, and to Abby Levenberg for taking these ideas to a new domain.

Overall, I would like to thank all the members of the Machine Learning group (MLRG) for all your thoughts and talks on topics of mutual interest, as well as making MLRG a great place to work. Similarly, thanks to the members of the ORCHID project for many stimulating conversations, to Hertford MCR and my other friends in Oxford for introducing me to many subjects outside Machine Learning.

Finally, thanks are due to my family and friends for putting up with the writing process, especially Aew, whose ever-wonderful cooking kept me alive through it all.

Related Publications

Some of this work has previously appeared in the following publications:

E. Simpson and S. Roberts and I. Psorakis and A. Smith (2013), Dynamic Bayesian Combination of Multiple Imperfect Classifiers. In *Intelligent Systems Reference Library series: Decision Making with Imperfect Decision Makers*, Intelligent Systems Reference Library series, Springer.

E. Simpson and S. Reece and A. Penta and G. Ramchurn and S. Roberts (2013), Using a Bayesian Model to Combine LDA Features with Crowdsourced Responses. *The Twenty-First Text REtrieval Conference (TREC 2012), Crowdsourcing Track*.

E. Simpson and S. Reece and G. Ramchurn and S. Roberts (2012), An Information Theoretic Approach to Managing Multiple Decision Makers. *Human Computation for Science and Computational Sustainability Workshop, Neural Information Processing Systems (NIPS 2012)*.

E. Simpson and S. Reece and G. Ramchurn and S. Roberts (2012), Dynamic Bayesian Combination of Multiple Imperfect Classifiers. *Human Computation for Science and Computational Sustainability Workshop, Neural Information Processing Systems (NIPS 2012)*.

E. Simpson and S. J. Roberts and A. Smith and C. Lintott (2011), Bayesian Combination of Multiple, Imperfect Classifiers. *25th Annual Conference on Neural Information Processing Systems (NIPS), Workshop on Decision Making with Multiple Imperfect Decision Makers*.

This thesis is entirely my own work, and the code that was used to run the experiments was also produced solely by the author, except for the overlapping community detection method, which was produced by Ioannis Psorakis. The dataset for the TREC Crowdsourcing challenge described in Chapter 5 was obtained using a system implemented collaboratively with Sarvapali Ramchurn, Steven Reece and Antonio Penta.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Decision making is a central problem in many application domains involving multiple autonomous agents. To make optimal decisions, agents must combine information from others whose reliability, skills and motives can vary greatly and may be unknown. The actions each agent takes can affect the ability of the whole system to make informed decisions, so it is important that co-operative agents organise themselves intelligently. This thesis proposes an efficient Bayesian approach to aggregating information, allowing reliable decision-making in *multi-agent systems (MAS)*.

Multi-agent systems are envisaged here in the broadest sense, as any grouping of agents to achieve a single goal, where the agents may be humans, software agents or robots. Here, an *agent* can be any distinct entity that observes and responds to its environment. Thus, the definition encompasses sensors and simple services that respond to queries, as well as rational agents with goal-directed behaviour [Russell and Norvig, 2009]. A range of established algorithms and architectures for computational agents are described in [Russell and Norvig, 2009; Wooldridge and Jennings, 1995]. The diverse strengths and autonomy of individuals enable multi-agent systems to solve problems that monolithic structures cannot. For example, agents with different observation and computation capabilities provide complementary information about their environment or objects of interest. This thesis is concerned with exploiting this diversity by combining responses from multiple agents to make more reliable decisions. The MAS of interest may be a loose, temporary combination

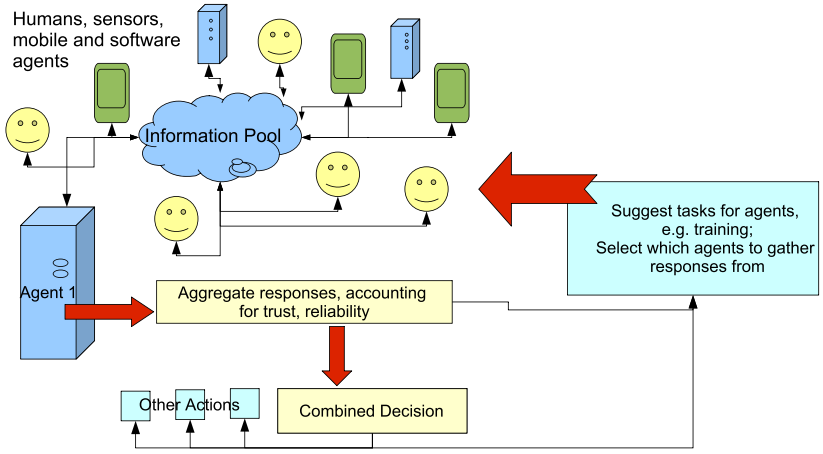


Figure 1.1: A diagram of a multi-agent system (MAS). The actions in the lower part of the image take place inside agent 1, who then exerts weak control on the pool of other agents.

of heterogeneous agents or a fixed team of agents that were programmed to exchange data according to a protocol. Agents may collaborate or have independent, competing goals, and the reliability of information supplied by agents may be inconsistent between agents and over time. This thesis therefore develops techniques for interpreting agents' responses and suggesting actions that best achieve the system's goals in the presence of unreliable agents.

A cartoon of a very general scenario of interest is shown in Figure 1.1. One member of an MAS, "agent 1", retrieves responses from an "information pool", to which several agents in a group contribute. Agent 1 aggregates this information to produce a combined decision, which it may use to take a variety of actions. Possible actions may include influencing the other members of the MAS to perform tasks that provide further information or improve the capabilities of the system, for example, through training.

The multi-agent system offers a number of advantages over alternative paradigms for decision making. One contrasting approach is to train a single classifier to work with the raw data directly, rather than combining the outputs of multiple agents. This allows the user of the system to optimise the classifier to their specific decision-making problem, avoiding the need to aggregate information from agents with potentially differing goals.

An example of this approach can be seen in fields such as computer vision, for example, which use deep belief networks to represent a wide range of complex, hierarchical relationships between the raw data, higher level features and target labels [Lee et al., 2009]. Using a single, sophisticated classifier can be challenging for a number of reasons. Firstly, a particular classifier technique must be chosen to suit the problem in hand, and learning a complex model can be a difficult problem [Adams et al., 2010]. The multi-agent approach developed in this thesis mitigates this problem by combining any number of heterogeneous decision-making methods and accounting for variations in their reliability. Secondly, the classifier must have a sufficient amount of raw data to discriminate between situations that require different decisions, yet access to data can be limited by communication bandwidth, confidentiality or costs. In contrast, individual agents in a MAS make decisions locally based on their own observations, and can communicate these decisions at a lower cost than the entire set of observations. Finally, learning a sophisticated model requires a large amount of training data, which may be unavailable. Even if we can interrogate expert labellers to obtain training examples, they may disagree among each other. Thus the process of building a training set could itself be viewed as a decision combination problem.

For complex decision problems, a single classifier can be computationally expensive to run or re-train in changing environments. Ensemble methods are an effective way to achieve high classification accuracy while avoiding this problem. They work by creating a very large number of fast, weak classifiers, which are combined in a simple manner, for example, by taking the majority vote [Dietterich, 2000; Tulyakov et al., 2008; Ho, 2002; Ho et al., 2002; Ranawana and Palade, 2006]. Ensemble methods and majority voting are further discussed in Chapter 2. The challenge of ensemble techniques is to generate a large and diverse pool of classifiers, so that the method used to aggregate decisions is unimportant. Thus, like single classifiers, ensemble methods require access to raw data and training examples. Ensemble methods are similar to the idea of combining multiple agents in the sense that each base classifier in the ensemble can be viewed as an agent, but require full control over the training of the classifiers. The multi-agent approach, however

places no demands on the number or characteristics of the agents, and needs no centralised access to training data.

Both ensemble methods and single classifiers aim to automate the decision making process without making use of existing agents and the expertise they possess. Often, this means attempting to duplicate the wealth of accumulated knowledge and skills of specialist classifiers or human experts. While a bespoke classifier may require a significant training expense, humans and specialised software agents may already be able to interpret complex problems, so such skills could be reused by a MAS. For example, people are able to perform pattern recognition tasks given only a small set of natural language instructions, and can make emotional and social judgements that may be hard to emulate. Specialised, automated classifiers and feature detectors can also be trained and tested on a wide range of scenarios, so that, in practice, they may outperform new classifiers trained for each specific decision making problem. Therefore, in certain situations, including existing agents in a decision making system may be more effective than attempting to replicate their skills in either a new single classifier or an ensemble. This thesis seeks to address the new challenges that arise when combining multiple heterogeneous, potentially untrusted agents. The next section further illustrates the potential of the MAS paradigm through a number of motivating scenarios.

1.1 Motivations

There is growing interest in several relevant application domains involving multi-agent systems, driven by recent developments in technology and its increasing availability. In these scenarios, robust methods for decision making are critical and a number of common technical challenges must be addressed. This section outlines four influential developments that motivated the work in this thesis.

1.1.1 Distributed Human Computation

Human strengths in pattern matching, natural language understanding and creative tasks continue to outstrip those of computers. For certain types of task, people require only simple instructions, while suitable computer programs are complex and costly to implement. The ubiquity of internet access and the flexibility of web applications allows organisations to tap into the skills of thousands of people, by assigning small computation tasks to different individuals. This parallelisation approach, sometimes known as *distributed human intelligence tasking*, overcomes the limits of processing time per task, while redundant decisions aim to improve reliability. In contrast to employing highly trained experts, distributed human computation aims to process far larger datasets at lower cost, typically using untrained members of a crowd. While distributed human computation is mainly concerned with large volumes of analysis tasks, the more general idea of asking unknown members of a crowd to perform work, then selecting or combining the results, is known as *crowdsourcing*. The people within the crowd can be seen as agents, where a central node distributes tasks then combines responses to make decisions. The agents act autonomously as the central node does not program or control them directly.

Commercial crowdsourcing platforms such as Amazon Mechanical Turk[1] and Crowdflower[2] provide an interface for creating web-based tasks, finding and paying workers. The quality of responses has been shown to be very variable [Bloodgood and Callison-Burch, 2010; Donmez et al., 2010; Raykar and Yu, 2012; Ipeirotis et al., 2010; Liu et al., 2012]. For example, when translating documents, some workers may attempt to cheat by using inferior automatic translation services, whereas others may provide good translations [Bloodgood and Callison-Burch, 2010]. In such systems, other considerations include using Machine Learning techniques to assist humans in decision making [Quinn et al., 2010], designing appropriate incentives [Ramchurn et al., 2013; Tran-Thanh et al., 2013], and recording and analysing the provenance of decisions [Ebden et al., 2012]. The trust that any end user places in the system's combined results may depend on their ability

[1] https://www.mturk.com/mturk/
[2] http://crowdflower.com/

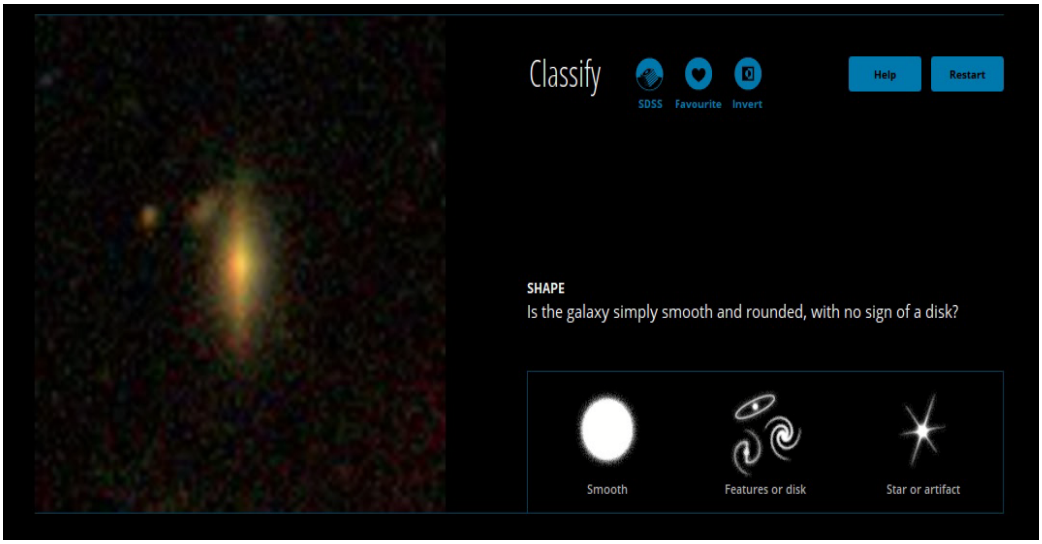


Figure 1.2: The web-based user interface for Galaxy Zoo.

to understand how a decision was made and which agents provided the information.

Related to Crowdsourcing is the field of *citizen science*, where interested members of the public are invited to take part in a science project by helping to analyse or collect large datasets. Citizen scientists have long been involved as observers in biodiversity projects such as the *Great Backyard Bird Count*[3]. Recent work extends the idea of volunteers as mobile observers by exploiting the capabilities of smartphones for recording and analysing data locally [Zilli et al., 2013]. A major collection of citizen science projects is managed by Zooniverse[4], which is a web portal that allows volunteers to participate in projects covering topics such as Astronomy, Biology or climate records. After logging in to a project, a citizen scientist is presented with a series of small tasks, which may involve answering questions about an image, a graph or some hand-written text. Questions with discrete answers are common [Smith et al., 2010] as it is thought to be easier for humans to express simple preferences rather than estimate more complex variables such as probabilities [Brochu et al., 2008]. Multiple participants repeat the same task to alleviate errors, then their decisions are aggregated to form a combined decision. An example of a specific project in Zooniverse is *Galaxy Zoo*, in which the types of galaxies are identified using a large corpus of images. The user interface for Galaxy Zoo is shown in Figure 1.2, display-

[3]http://www.birdsource.org/gbbc/
[4]http://www.zooniverse.org

ing one of several questions that volunteers are asked when analysing a galaxy. Generally, the analysis tasks are hard to automate but require only simple instructions for humans to complete [Smith and Lintott, 2010]. Such projects allow experts' limited time to be used more efficiently, for instance by filtering out interesting galaxies that are more worthy of their attention, but they also allow the scientific community to reach out to educate and involve the public about their work. A successful project may also inform the design of Machine Learning algorithms to eventually replace the human agents, for instance by providing training labels.

A significant challenge with human computation is handling the unreliability of untrained agents in a principled manner, rather than developing ad-hoc approaches for every citizen science application. Major opportunities exist to increase efficiency through intelligent task allocation and by personalising the volunteers' experience. The latter applies particularly to citizen science, where participation is voluntary, so tasks should suit a user's interest but avoid repetitiveness. Deciding which agent is suitable for a particular task would allow reliable workers to concentrate on more difficult tasks that suit their specialisms, while avoiding deploying agents to redundant tasks. More effective integration of Machine Learning algorithms could also increase the speed or scale of datasets that can be processed. When designing human computation systems, it is important to remember that the agents are autonomous people with individual goals, so a centralised system can only exert weak control through nudges and suggestions.

1.1.2 Ubiquitous, Mobile and Pervasive Computing

The increasing ubiquity of networked devices and sensors presents opportunities for services that collate data in real-time and provide contextual, personalised information. Humans may exchange messages, reviews, pictures and video, while sensors provide information about the current state of the environment and appliances in the home through the *Internet of Things* [Berners-Lee et al., 2001]. By combining data from these heterogeneous sources, decisions can be made to control an intelligent home or make recommendations to

a user. As a person moves about and carries out different activities, there is the potential for ad-hoc collaborations involving other people, local sensors, or software agents providing specialised services. Each member of such a collaboration can be seen as an agent in a MAS. However, services may be of variable quality, with better providers charging for use. The increasingly large, dynamic pool of potentially untrusted agents presents a challenge when forming decisions from shared data. When making one particular decision, filtering out the relevant information may present a significant difficulty. An additional problem at present is the limited power of mobile devices, which could limit the quantity of information that can be transmitted and processed. In the field of mobile computing, recent work by [McInerney et al., 2012] develops methods for learning users' routines and identifying unusual behaviour, with the goal of recommending services in unfamiliar locations, which may require the combination of information from sources that change depending on the individual's location and activity.

A large pool of sensor information is managed by platforms such as Sensorpedia [Gorman et al., 2009]. Various tools for publishing information in the Internet of Things are provided by projects such as Xively[5]. Sensor information has been exploited for crowdsourced radiation monitoring after the Fukushima disaster [Venanzi et al., 2013]. Much of the work on sensor networks concerns fusing real-valued data from multiple sensors, but does not advise on the related problem of decision making.

1.1.3 Automation and Communication in Specialist Teams

Robots, unmanned surveillance devices and mobile devices are an increasingly prevalent aid to teams of human specialists managing various situations, such as disaster recovery or medical operations. In the disaster recovery scenario, unmanned aerial vehicles (UAVs) can provide situational awareness to support emergency responders [Rodriguez et al., 2006; Daniel et al., 2009], who will simultaneously be receiving information from their colleagues. Recent work has tackled topics such as coordination of unmanned ve-

[5]https://xively.com

hicles [Rahwan et al., 2011; Michalak et al., 2010] and coalition formation to adaptively determine roles for team members [Rahwan et al., 2012].

Medical situations can also employ robotic devices for precision or remote operations and medical sensing [Lal and Niyogi, 2013; Lum et al., 2007]. The decision-making process can therefore involve consulting various remote specialists on demand at critical moments in a procedure. Earlier work on medical analysis has previously focused on combining experts' decisions [Dawid and Skene, 1979], and provides a foundation for the approaches developed here to handle uncertainty with unreliable decision makers.

Complex situations require efficient deployment of team members and remote devices to gather critical information using different specialist skills. At the same time, agents outside the core team may contribute vital information, either face-to-face or over the web. The information gathered can take different forms, such as location updates, sensor data [Daniel et al., 2009], text messages, pictures and videos [Rodriguez et al., 2006], some of which may be published on social media [Sarcevic et al., 2012; Vieweg et al., 2010]. There is the potential to overload a decision maker with information, so it is imperative to distil the important, trusted notifications from the combination of data sources. Interpreting each data type requires different skills, so there is an opportunity to use a suite of software agents to interpret different data sources, including messages produced by people, using Machine Learning algorithms. In this scenario, decision making not only involves the deployment of agents, but also assessment of events, for example deciding when and where an incident has occurred, or whether to label an area as safe.

1.1.4 Information Overload

The issue of information overload and unreliable agents are faced in many other contexts, including news sources, blogs and microblogging sites such as Twitter[6], as well as in large organisations such as corporations or government departments. Each journalist, blogger or other contributor represents an agent, and there are many possible decision-making prob-

[6]Twitter, see http://www.twitter.com

lems, including the following examples: correctly labelling current events; predicting economic, financial or political trends by pooling opinions (see recent work by [Levenberg et al., 2013]); making recommendations by combining reviews and ratings. The vision of a *Semantic Web* describes computational agents interacting autonomously with web-based services to carry out tasks on behalf of the user [Berners-Lee et al., 2001; Hendler, 2001]. Individual agents may use specialist agents to provide information, for example, deciding which item to buy from a set of reviews. Projects such as *Linked Data* [Berners-Lee, 2006], Freebase[7] and DBPedia[8], publish data using universal identifiers for entities, allowing consumers to compose previously unrelated data from a number of web-based services. This will enable autonomous, web-based agents to interact and form cohorts, requiring decision-making algorithms for handling data from unfamiliar and disparate agents. In this context, the multi-agent system has no central design or control, no fixed structure and the agents may have very varied goals.

1.2 Summary of Technical Challenges

From the application domains described above, we can identify two key decision-making problems in multi-agent systems. Firstly, how to make a combined decision given information from multiple agents. A decision in this sense may be a choice of action, or a decision about belief in a particular proposition, such as the correct label for an object or the current state of the environment. The information provided by agents can also be seen as decisions to respond in a particular way given their observations, so this problem can be seen as one of *decision combination*. The second issue is how to organise or influence a set of agents to best meet the system's goals. The motivating scenarios described several common challenges that will be faced when addressing these two decision-making problems; these are summarised below.

Variation among agents: their abilities, trust, costs and continued availability.

[7] http://www.freebase.com
[8] http://dbpedia.org/

Dynamics: the changing reliability as agents move, learn or their goals and interests change. Dynamics may be gradual drifts and sudden state changes.

Uncertainty: the abilities of agents are themselves not reliably known. Methods for sharing and verifying reliability between heterogeneous agents are typically lacking, and agents – particularly humans – may not supply any measure of their own confidence, nor an explanation of how a decision was reached. The reliability must therefore be learnt from observations and background knowledge of individuals.

Weak control: given a large set of possible tasks and a pool of agents, how do we suggest personalised tasks that meet both the system's aims and the individual's goals? The aim is to obtain information efficiently to enable reliable combined decisions while limiting costs, which may be defined in terms of money, network usage, energy or processing power.

This thesis proposes a set of techniques to address these challenges, enabling multi-agent systems to efficiently make more accurate, combined decisions. The foundation for these approaches is Bayesian inference, which provides an elegant mathematical framework for optimally fusing multiple sources of information in the presence of uncertainty. In many of the target applications described above, Bayesian approaches have not yet been applied, possibly due to their perceived complexity, speed or the focus of system designers on other aspects of the technology. This thesis demonstrates that scalable Bayesian approaches can deliver significant performance gains when working with unreliable agents. Furthermore, these approaches naturally handle uncertainties at all levels of a model, including in the responses of agents and in our understanding of their reliability. The next section introduces the concept of Bayesian inference.

1.3 Bayesian Inference

Bayesian inference is a form of subjective inference for estimating the probability of a hypothesis. Probability theory is the foundation of statistical inference, and is explained

clearly and thoroughly in [Jaynes, 2003]. Statistical inference is a paradigm for soft logical reasoning where observations of the world modify degrees of belief, represented by probabilities. Probabilities take values in the interval $[0, 1]$, where 1 indicates certainty that a hypothesis is true and 0 indicates that the hypothesis is definitely not true. In Bayesian inference, all points of uncertainty are modelled as random variables with a probability distribution that represents an abstract belief in their true value. A probability distribution is a function that describes the weight of belief at each value of a variable. These probabilities may be predictions where the value is yet to be observed, or simply variables whose true value is unknown to the decision maker. The subjectivity arises due to the incorporation of prior knowledge about the random variables. When a decision maker has prior knowledge over the probability distribution of a random variable, Bayesian inference provides the optimal means of combining observations. Prior knowledge is the background expertise, context and previous observations that the decision maker possesses before observing the current set of data, $\boldsymbol{D}$. Given this set of observations, $\boldsymbol{D}$, we perform inference to update our beliefs over a random variable, z, by applying Bayes' theorem, named for early work by Thomas Bayes, which was generalised by Pierre-Simon Laplace [Stigler, 1986]:

$$p(z|\boldsymbol{D}) = \frac{p(z)p(\boldsymbol{D}|z)}{p(\boldsymbol{D})}. \tag{1.1}$$

The terms in Equation 1.1 have the following meanings: $p(z)$ is the *prior* probability of z; $p(z|\boldsymbol{D})$ is the posterior probability of z, combining the decision maker's prior beliefs over z with observations $\boldsymbol{D}$; $p(\boldsymbol{D}|z)$ the likelihood of observing data $\boldsymbol{D}$ given different values of z; and $p(\boldsymbol{D})$ is the marginal likelihood of $\boldsymbol{D}$. The level of support for a particular hypothesis, z, is quantified by $p(z)p(\boldsymbol{D}|z)$.

Calculations using Bayes' theorem require the assumption of certain probability distribution functions for each of the terms in Equation 1.1. The choice of functions determines the probabilistic model. Where the correct model is unknown, Bayes' theorem can be rewritten to *marginalise* this uncertainty over models, which means to integrate or sum over the distribution given each possible model M, multiplied by the prior probability of

the model, M:

$$p(z|\boldsymbol{D}) = \int p(M)\frac{p(z|M)p(\boldsymbol{D}|z,M)}{p(\boldsymbol{D}|M)}\mathrm{d}M. \tag{1.2}$$

The distribution $p(M)$ represents a prior over the model. In parametric modelling, a restricted set of models is considered by assuming particular functional forms with uncertain parameters. Thus, $p(M)$ represents a distribution over the values of model parameters. The model priors themselves usually take a standard form with a fixed set of parameters, $\boldsymbol{h}$, known as the *hyperparameters*. It is often convenient to choose a particular distributional form known as *conjugate distributions* for the model priors over each parameter θ given its hyperparameters $\boldsymbol{h}$. Where x is a variable in the set $\boldsymbol{D}$, conjugate distributions are pairs of prior distributions $p(\theta|\boldsymbol{h})$ and posterior distributions $p(\theta|x,\boldsymbol{h})$ that have the same functional form, so that to find the posterior over model parameter θ simply requires updating the hyperparameters $\boldsymbol{h}$, rather than changing the distribution function itself. The likelihood distribution chosen for $p(x|\theta)$ determines which distributions over θ are conjugate. For example, if $p(x|\theta)$ is a binomial distribution with parameter π, the conjugate prior is a beta distribution $p(\pi|a,b)$ with hyperparameters a and b.

When inferring a distribution over z, any unknown parameters in the model can be marginalised in the manner of Equation 1.2. Marginalisation also allows us to obtain an *expectation*, $\mathbb{E}_M[p(z|\boldsymbol{D})]$ with respect to M, which is the mean value of $p(z|\boldsymbol{D})$ that we would expect to see if we repeatedly sampled M from $p(M)$ and calculated $p(z|\boldsymbol{D},M)$. Equation 1.2 allows us to account for *second-order uncertainty*, i.e. uncertainty in the model itself. While Bayesian inference algorithms account for this uncertainty explicitly, alternative methods choose fixed model parameters, selecting a single model that maximises either the likelihood of the data (*maximum likelihood* techniques, see [Dempster et al., 1977]) or its posterior probability (*maximum a posteriori* or MAP techniques).

A convenient feature of Bayes' theorem is that it can also be rearranged by writing the

marginal likelihood $p(\boldsymbol{D})$ as an integral:

$$p(z|\boldsymbol{D}) = \frac{p(z)p(\boldsymbol{D}|z)}{\int p(\boldsymbol{D}|z)\mathrm{d}z}$$
$$\propto p(z)p(\boldsymbol{D}|z). \tag{1.3}$$

When comparing the distribution over different values of z, it is not necessary to calculate the denominator $p(\boldsymbol{D})$ explicitly, although it can be derived by marginalising over likelihoods given different values of z. The advantage is that it is often easier to work with the likelihood $p(\boldsymbol{D}|z)$ rather than specifying $p(\boldsymbol{D})$ in another way. For example, where $\boldsymbol{D}$ consists of a number of independent observations, $d_1, d_2, ..., d_N$, the likelihood $p(\boldsymbol{D}|z)$ can be broken down into a simple product:

$$p(\boldsymbol{D}|z) = \prod_{i=1}^{N} p(d_i|z). \tag{1.4}$$

Thus, Bayesian inference requires dealing only with this straightforward functional form.

The use of a prior in inference has a number of advantages. Relevant prior knowledge cannot simply be discarded by any rational decision maker, so is encoded in the distribution $p(z)$. In fact, prior information can be obtained from previous observations, so it is intuitive that it should be treated in a similar manner to the likelihood term; thus the prior could be written as $p(z|\boldsymbol{D}_{\mathrm{prior}})$, where $\boldsymbol{D}_{\mathrm{prior}}$ is prior information. The prior probability acts as a regularisation term, so that small numbers of observations do not dominate the posterior distribution, and highly implausible hypotheses that generate the same data will not be accepted without a high quantity of observations. This contrasts with *frequentist* approaches where the probability distribution is determined entirely from the observations.

The use of Bayes' theorem also facilitates the design of *generative models*. Generative models are those that describe how the data was produced, so can be used either to predict or generate new observations, or infer unknown or *latent* variables in the model given observations. In contrast, *discriminative* models learn the distribution $p(z|\boldsymbol{D})$ directly.

In summary, Bayesian inference provides a rational and convenient method for com-

bining prior and observed information, while accounting for uncertainties at all levels.

1.4 Contributions

This thesis builds a framework for combined decision making in multi-agent systems that is founded on Bayesian inference, yet uses principled approximations to allow the algorithms to scale to big data problems. This reduces the reliance on heuristic approaches, which require fine tuning for specific use cases. Specific contributions include:

- An empirical and theoretical comparison of established decision combination methods against a Bayesian approach
- A scalable inference algorithm for Independent Bayesian Classifier Combination using variational inference (IBCC-VB)
- The application of IBCC-VB to citizen science and web attack datasets
- The use of community analysis techniques to analyse behavioural types of citizen scientists
- DynIBCC, a dynamic model for learning the changing behaviour of agents, along with a variational inference algorithm, DynIBCC-VB
- Illustration of how DynIBCC-VB can be used to track dynamic behaviour using synthetic and real-world datasets
- Application of DynIBCC-VB to a mixture of agents' respones and continuous features, allowing decision making with limited numbers of agents' responses
- An information-theoretic approach to choosing pairs of agents and tasks using a utility function based around information gain
- The Hiring and Firing algorithm for automatically maintaining a reliable pool of workers and selecting informative analysis tasks.

The methods developed in this work are suitable to be run by either a central control agent, or by individuals in a distributed system where information is shared.

1.5 Overview of Thesis

The next chapter presents a thorough investigation of decision combination methods, contrasting a Bayesian approach to established supervised learning methods and fixed rules, and considering the application of unsupervised methods. The first part of the chapter reviews the theoretical motivations for each method, showing that many methods have related foundations with different assumptions. The second part provides an empirical comparison of key methods, demonstrating how their performance changes in different scenarios. Chapter 3 presents a scalable variational inference algorithm for Bayesian decision combination, IBCC-VB, which is compared on several real-world datasets. The chapter also shows how community analysis techniques can be applied to information learnt by IBCC-VB to detect distinct types of volunteer in a citizen science project. A novel approach to Bayesian decision combination, DynIBCC, that models the evolving behaviour of agents, is proposed in Chapter 4. This tracking ability is exhibited by examining the dynamics of Zooniverse agents and changing clusters of agents. Chapter 5 considers the challenge of weak control through the choice of assignments for agents in a multi- agent system. First, continuous features are used to augment limited numbers of agents' responses, showing how a corpus of documents can be classified using crowdsourced labels. Then, an information-theoretic optimisation approach is proposed and used to design the Hiring and Firing algorithm for selecting particular agents and tasks, obtaining accurate decisions more rapidly from an unreliable crowd. Finally, Chapter 6 specifies some future directions that extend the methods developed in this thesis in several ways, including automatically suggesting training exercises and motivating agents to improve future performance of the system. The mathematical notation in the thesis uses the conventions specified in Table A.2 for the reader's reference. Variables that are consistently used throughout the thesis are listed in Table A.3, while important acronyms are given in Table A.1.

Chapter 2

Decision Combination Methods

This chapter focuses on the task of combining decisions from multiple agents. A range of combination methods are reviewed, including Bayesian Classifier Combination (BCC), which incorporates many of the advantages of other methods within the principled framework of Bayesian inference. The chapter begins by defining the problem of decision combination and the factors that influence its solutions.

The goal of decision combination is to produce aggregate decisions that are more reliable than those obtained from a single individual. A justification for combining decisions can be seen by considering Jensen's inequality, which allows us to place lower bounds on the accuracy of combined decisions [Freund et al., 2004], and investigate when a combination might produce more accurate results [Brown et al., 2005]. Consider a set of K agents, where each agent $k \in \{1, .., K\}$ submits an estimate X_k of a correct decision, which can be any integrable variable, including a posterior probability. Jensen's inequality states that:

$$f(\mathbb{E}[X_k]) \leq \mathbb{E}[f(X_k)], \tag{2.1}$$

where $f()$ is a convex function, such as an error function. Here, we take $f()$ to be the squared error, such that $f(x) = (t - x)^2$, where t is the correct decision, referred to here as the *target value*. The right-hand side represents the expected error of an individual agent, while the left-hand side is the error of the expected decision. Thus, the inequality

states that a simple combination of decisions, i.e. their expectation, $\mathbb{E}[X_k]$, will be at least as accurate as the expected individual agent. This is particularly helpful when the relative performance of individuals is not known, so we cannot simply select one informative agent. The discussion below gives an intuition as to how combinations can also produce higher accuracy than any single agent.

We can write the inequality in a more general form so it can be applied to a wide range of combination methods:

$$f\left(\sum_{k=1}^{K} w_k g(X_k)\right) \leq \sum_{k=1}^{K} w_k f\big(g(X_k)\big), \tag{2.2}$$

where $g()$ is any real-valued measurable function, such as $log(x)$, and w_k is a weight on agent k where $\sum_{k=1}^{K} w_k = 1$. Thus the bounds apply not just to linear combinations of agents' responses, but to other methods that will be discussed in detail in the following sections.

Now, to gain an understanding of when a combined decision will have lower error than the expected individual, we can re-write Equation 2.2:

$$f\left(\mathbb{E}[X_k]\right) = f\left(\sum_{k=1}^{K} w_k g(X_k)\right) = \sum_{k=1}^{K} w_k f\big(g(X_k)\big) - \sum_{k=1}^{K} w_k \big(g(X_k) - \mathbb{E}[X_k]\big)^2, \tag{2.3}$$

This is called the *ambiguity decomposition* [Krogh and Vedelsby, 1995; Brown et al., 2005]. The last term in the equation represents the diversity of errors of the individual agents. The performance of the combination therefore depends on the error of the individuals, the correlation between their errors, and the weights w_k, which can be optimised by methods discussed in Section 2.2. Intuitively, the combined decision is more accurate when uncorrelated errors cancel out. An alternative form of the ambiguity decomposition is given by [Wang et al., 2010] for categorical decisions that cannot be integrated, such as votes in a multi-class problem. Further justifications for the success of decision combination are given in [Dietterich, 2000; Dietterich, 1997]. These include the case where a combination can represent a richer space of hypotheses for decision-making that individ-

ual agents cannot. These justifications also relate to the diversity of the constituent agents, which can be measured in a number of ways [Ranawana and Palade, 2006].

The theory above applies when the agents estimate the target decision, so that both the agents' decisions and the combined decision take values in the same target set, $\mathbb{D}_t$. This scenario is referred to here as *homogeneous* decision combination and is the concern of *multi-classifier systems*, which employ *ensemble methods* to generate a diverse set of *base classifiers* and combine their predictions [Dietterich, 2000; Tulyakov et al., 2008; Ho, 2002; Ho et al., 2002; Ranawana and Palade, 2006]. Many of these ensemble methods focus on the task of creating a diverse set of base classifiers. For example, established methods such as *bagging*, *boosting* and *AdaBoost* train the individuals using different subsets of data [Freund and Schapire, 1995; Dietterich, 2000]. Boosting and AdaBoost generate classifiers iteratively, emphasising training samples that the current set of classifiers did not learn to classify correctly. Since this level of control is not available over human decision makers or pre-existing computational systems, this chapter focuses on the combination functions themselves. However, ensemble methods may provide important insights into how the combination can be improved by influencing the training of agents.

We may wish to combine *heterogeneous* information, where each agent k produces values from a different set, $\mathbb{D}_k \neq \mathbb{D}_t$. For example, the target decision could be whether or not to recommend a song to a particular person, where the information we wish to combine consists of reviews by other people and musical features of the song, such as its musical style and tempo. We can turn this into a homogeneous decision combination problem by learning a *base classifier* that predicts the target decision for each heterogeneous information source. For instance, we could learn a base classifier that predicts whether to recommend a song from its drum rhythm. An alternative is to view the decision combination problem as a general classification problem [Wolpert, 1992], where each agent corresponds to an input to a classifier [Tulyakov et al., 2008]. This chapter compares methods from each of the two perspectives.

Agents can output responses with a range of data types, including discrete labels,

ranked lists, or continuous values. This thesis focuses on categorical decisions and probabilities of such decisions, which arise in a wide range of scenarios where the aim is to combine the local analysis skills or observations of different agents, without transmitting large amounts of sensor data. Working with nominal labels allows humans to produce responses more rapidly and consistently than estimating continuous variables [Ho et al., 2002]. The combined decisions are also taken to have categorical values, so include a choice of action or label for an object or state. However, the methods proposed in this work could be extended to ranked lists and continuous variables.

An important question when designing a decision combination method is how to make use of any additional information besides the decisions themselves. Such information includes any training examples where both the agents' responses and the correct target decisions are known. The combiner may also have prior knowledge about the agents, such as their expertise in a particular problem domain. The input data used by the agents may be visible to the combiner and could indicate likely correlations between their responses. A number of approaches that use this information in different ways are reviewed in this chapter. Broadly, decision combination methods can be categorised according to how they use training examples and how the combination function adapts as the combiner observes the agents' decisions:

1. *Fixed Combination Functions* that do not adapt to data at all

2. *Supervised* learning methods that use a set of training examples to determine a combination function

3. *Unsupervised* learning methods that exploit latent structure in the agents' decisions, such as groupings of similar agents, to adjust a combination function without training examples

4. *Semi-supervised* learning methods that can exploit both training examples and latent structure in the agent decisions, so may be suitable when training data is limited.

An alternative high-level grouping into syntactic, structural and statistical methods is also proposed by [Ho et al., 2002].

The following sections provide background on decision combination methods developed using a range of theoretical foundations: statistical inference, including Bayesian inference; optimisation techniques that search for the most accurate combination function; and techniques that have been shown to work well empirically. The methods are discussed within the categories listed above, starting with fixed combination functions in Section 2.1, then supervised methods in Section 2.2, and unsupervised methods in Section 2.3. Section 2.4 then presents Bayesian Classifier Combination (BCC), a semi-supervised approach, and discusses its theoretical relationship with other methods. Methods chosen for their simplicity can often be seen as approximations of more complex methods under a set of simplifying assumptions, as is discussed in Subsection 2.4.3. Finally, Section 2.5 contributes an empirical comparison of key decision combination methods, showing the benefits of different approaches in a range of controlled scenarios.

2.1 Fixed Combination Functions

There are a number of fixed rules for combining decisions that do not use explicit probabilistic models to adapt to data [Tulyakov et al., 2008]. The inputs to these functions are agents' decisions in the form of scores, probabilities for each decision or ranked lists. Among these, the product and sum combination rules are significant because they approximate more sophisticated combination methods.

The *product rule* can be derived from Bayesian theory, assuming agents are conditionally independent and supply unbiased, calibrated probabilities of the correct decisions [Kittler, 1998]. Mis-calibrated probabilities are those that over or under-exaggerate certainty, potentially placing too much or too little emphasis on a particular piece of evidence; fixed combination functions have no way of adjusting for this problem. For each data point, we assume there exists a *target label*, t, which represents the correct decision for that data point that we wish to infer. Each agent k provides beliefs $\{b_j^{(k)}|j = 1, .., J\}$ that the target

label t has value j, where there are J possible target values. To derive the product combination rule, we assume that these belief values are probabilities that are meaningful to the combiner in the sense that

$$b_j^{(k)} = p\left(t = j | b_j^{(k)}\right). \tag{2.4}$$

This is a simplifying assumption over combination methods that learn a distribution $p\left(t|b_j^{(k)}\right)$ from observing instances of $b_j^{(k)}$, which are described in later sections. Using Bayes' theorem, we can write the posterior probability of t given a set of outputs $\boldsymbol{b_j}$ from agents $1, .., K$ as:

$$\begin{aligned} p(t = j|\boldsymbol{b_j}) &= p(t = j) \prod_{k=1}^{K} \frac{p\left(b_j^{(k)}|t\right)}{p\left(b_j^{(k)}\right)} \\ &= p(t = j) \prod_{k=1}^{K} \frac{b_j^{(k)}}{p(t = j)} \end{aligned} \tag{2.5}$$

The product rule can then be used to determine the most likely value $\hat{t}$ of the target label, t:

$$\hat{t} = \underset{j}{\operatorname{argmax}} \left(p(t = j)^{1-K} \prod_{k=1}^{K} b_j^{(k)} \right) \tag{2.6}$$

The product rule combines independent evidence, so each individual can check for a certain feature or sub-pattern in the data. The combination of these sub-patterns indicates the most probable target label [Hinton, 1999]. This provides a way to break down high-dimensional problems, which may be too complex for a single agent to model accurately and would potentially lead to agents that over-fit or are overly vague. The distinctive feature of the product rule is that any agent with strong beliefs (close to zero or one) can greatly affect the combined decision.

In contrast, the *sum rule* averages over the individual agents' models. The combined probability is estimated by taking the mean:

$$\hat{p}(t = j|\boldsymbol{b}_j) = \frac{1}{K} \sum_{k=1}^{K} b_j^{(k)}. \tag{2.7}$$

The sum rule then selects a decision according to

$$\hat{t} = \underset{j}{\operatorname{argmax}} \left(\frac{1}{K} \sum_{k=1}^{K} b_j^{(k)} \right) = \underset{j}{\operatorname{argmax}} \left(\sum_{k=1}^{K} b_j^{(k)} \right). \tag{2.8}$$

Therefore, sum rule decisions are less affected by a single agent with a strong belief, so rely less on all agents submitting trustworthy belief values. However, the sum rule typically does not allow such sharp decision boundaries as the product rule, as it mixes the individual models and thus decreases their precision.

In situations where the agents are unable to supply continuous belief values, methods are required for combining greedy decisions, such as discrete labels chosen by human decision makers. The product rule is not applicable in such scenarios, since a single zero vote will cause a decision to be rejected. Hence, when there is any disagreement between agents, all decisions will be vetoed. The sum rule can be adapted, however, to deal with discrete labels, giving us the *majority voting* rule. In this case, we assume that agents choose a single decision $c^{(k)} \in [1, J]$ rather than output probabilities of each target value $b_j^{(k)}$. We can alter Equation 2.8 to obtain the majority voting rule:

$$\hat{t} = \underset{j}{\operatorname{argmax}} \left(\sum_{k=1}^{K} \delta \left(j - c^{(k)} \right) \right), \tag{2.9}$$

where $\delta(x)$ is a delta function that indicates when its argument is zero, so in this use case it indicates when j and $c^{(k)}$ agree. The delta function is defined as follows:

$$\delta(x) = \begin{cases} 1 & x = 0 \\ 0 & x \neq 0 \end{cases} \tag{2.10}$$

For multi-class problems, the sum rule can also be extended to combine ranked lists of classes, which agents have sorted according to their preferences. This method is known as Borda count, and involves converting ranks to scores, which are then summed [Ranawana and Palade, 2006]. The sum rule and majority voting assume that each agent's model is

equally likely to be informative of the correct decision. Weighted majority and weighted sum methods explained in Section 2.2 relax this assumption, for example, by adjusting the function according to the probability that each agent is correct. An advantage of the sum rule, product rule and majority voting is their simplicity to implement and calculate, and their potential to deliver performance improvements over individual agents according to Jensen's inequality and the ambiguity decomposition, detailed in Equations 2.2 and 2.3.

2.2 Supervised Methods

Supervised combiners aim to improve the accuracy of a combination by learning the parameters of a function from observations of agents' responses, known target labels, and in some cases, other input data. These functions often involve a weighted sum or weighted product of agent responses, a conversion from agent responses to probabilities, or a combination thereof. However, the motivations and learning algorithms differ greatly.

2.2.1 Weighted Sums and LinOPs

One approach is to learn weights that adjust the contribution of each agent to a sum rule:

$$\hat{t} = \underset{j}{\operatorname{argmax}} \sum_{k=1}^{K} w^{(k)} b_j^{(k)} \tag{2.11}$$

This variant of Equation 2.8 is known as a *weighted sum* rule. Weights can be used to address the issues of unreliable agents or low diversity caused by highly-correlated subsets of agents, by reducing the contribution of some members of the combination. Where the agents supply discrete decisions rather than continuous beliefs, we can also write a *weighted majority* rule:

$$\hat{t} = \underset{j}{\operatorname{argmax}} \left(\sum_{k=1}^{K} w^{(k)} \delta\left(j - c^{(k)}\right) \right), \tag{2.12}$$

where $\delta(x)$ represents the delta function introduced in Equation 2.10. Weighted sums of probabilities supplied by agents are also known as *Linear Opinion Pools (LinOPs)* [Benediktsson and Swain, 2002; Genest and Zidek, 1986]. There are two common theoretical motivations for calculating the weights in a weighted sum or weighted majority.

1. Model selection: assume one agent in our pool can closely match the *data-generating model*, i.e. the most accurate predictor of the target labels that we could possibly attain. Model selection integrates over our uncertainty as to which agent this is. Weighted methods can therefore be seen as *soft selection* methods, as they use probabilistic weights rather than making a hard choice to use a single agent.

2. Model combination: construct a new, more complex model from existing models or reduce the effect of uncorrelated errors in the individual agents, thereby reducing the expected error of the combined decision.

Bayesian methods for soft model selection are known either as *Bayes Optimal Classifiers (BOC)* or *Bayesian Model Averaging (BMA)*. These methods weight each agent by the posterior probability that it is the data-generating model given the data. While BOC searches over an entire space of models with different parameters, BMA is typically applied to a sample or given set of models, often using methods such as Gibbs' algorithm to search for the correct model efficiently [Opper and Haussler, 1991]. If the data-generating model is not present in the pool, BMA tends toward selecting the model closest to the data-generating model, so is not appropriate if we wish to improve on the best model currently available by combining it with others [Minka, 2000; Clarke, 2003; Monteith et al., 2011]. It is also unclear how BMA should handle data points which have responses from only a subset of agents that are known to be unreliable.

Two algorithms proposed by [Littlestone and Warmuth, 1989] provide approximate methods for soft model selection or BMA along with proofs of error bounds, and are referred to here simply as *Weighted Majority* and *Weighted Sum*. Weighted Majority applies to agents that output discrete decisions $c^{(k)} \in [1, J]$, whereas the Weighted Sum applies to beliefs in decisions $\{b_j^{(k)} | j = 1, .., J\}$. Both methods use a common algorithm to update

the weights $w^{(k)}$ in Equation 2.11 or 2.12 in a sequential manner as new decisions are received from agents. The initial weight for all agents, w_0, is usually set to 1. The weight $w_i^{(k)}$ for agent k at data point i is defined by

$$w_i^{(k)} = w_0.\beta^{\epsilon_i^{(k)}}, \tag{2.13}$$

where β is a fixed punishment value $\beta \in [0, 1]$ reflecting the decrease in belief when k makes a mistake that k is the correct model for the target labels. The punishment decreases the weight of k each time k's decision agrees with the majority but is incorrect. For Weighted Majority, where agents make categorical decisions, this count is $\epsilon_i^{(k)}$:

$$\epsilon_i^{(k)} = \sum_{n=1}^{i-1} \delta(t_n - \hat{t}_n)\delta(c_n^{(k)} - \hat{t}_n). \tag{2.14}$$

where $\delta(x)$ is the delta defined in Equation 2.10. For Weighted Sum, where agents supply beliefs $b_j^{(k)}$ of each decision j, we define $\epsilon_i^{(k)}$ as:

$$\epsilon_i^{(k)} = \sum_{n=1}^{i-1} |t_n - \hat{t}_n|.|t_n - c_n^{(k)}|. \tag{2.15}$$

In this case, the discrete delta functions are replaced by the amount of error in the overall decision, $|t_n - \hat{t}_n|$, and the agent's decision, $|t_n - c_n^{(k)}|$. If we normalise the weights to sum to unity, so that $\sum_{k=1}^{K} w_i^{(k)} = 1$, each weight $w_i^{(k)}$ represents the approximate probability that agent k has the data-generating model. This means that the Weighted Sum can be used directly to estimate the probability of a target label according to:

$$\hat{p}(t_i) = \frac{1}{\sum_{k=1}^{K} w_i^{(k)}} \sum_{k=1}^{K} w_i^{(k)} b_j^{(k)}. \tag{2.16}$$

An alternative model combination approach that weights each agent is *stacked generalisation* or *stacking*, proposed by [Wolpert, 1992]. Rather than determine the posterior probability that each agent matches the data-generating model, stacking assesses the error

rate of each agent using a technique based on cross validation. This technique involves testing an agent on data points they have not previously seen, where the assessor knows the correct target label. The agents' responses for any new data points can be combined by any method that takes these error rates into account, such as a weighted sum. As shown by [Clarke, 2003], stacking can outperform BMA when none of the agents are similar to the data-generating model, because it can converge to a combination that better approximates the data-generating model. Building on this idea, *Bayesian model combination* (BMC) searches a space of combination functions to find the best approximation to the data-generating model [Monteith et al., 2011]. Optimising combinations rather than individual models is of particular importance if we want to use non-expert agents or weak decision makers to build a more reliable model. In theory, BMC is equivalent to using BMA to soft-select from an enriched space containing all possible combinations of models. In practice, however, this space is large and requires sampling [Monteith et al., 2011]. Other Bayesian approaches, such as Bayesian Classifier Combination (see Section 2.4), account for the uncertainty over the combination function by updating distributions over each of the parameters of an assumed graphical model.

An alternative approach to model combination is to optimise the true positive rate (TPR) and false positive rate (FPR) to suit a particular application. The TPR is the fraction of positive examples identified correctly and the FPR is the fraction of negative candidates incorrectly labelled as positive. This is particularly applicable in cases where the costs of false positive classifications differ from those of false negatives, such as medical diagnosis. We can attain pairs of TPR-FPR values that none of the individual agents could reach alone by simply selecting one agent at random [Scott et al., 1998]. The TPR-FPR values can be improved in some cases by choosing between AND and OR rules for combining classifier outputs [Haker et al., 2005]. It is also possible to optimise the TPR-FPR pairs by evolving more complex combinations of operators on agents' responses using Genetic Programming [Langdon and Buxton, 2001]. However, this is likely to be costly as Genetic Programming must search a very large space of functions by proposing and evaluating ran-

dom alterations or combinations of functions. It can therefore become difficult to know why a particular combination function was chosen, how well it generalises to new data and whether it is optimal.

A computationally efficient technique for approximating statistical error rates involves using similarity metrics between the agents' responses and the target decisions. In the field of *collaborative filtering*, the decisions of a target user can be predicted by weighting opinions of other users (agents) according to how similar their previous decisions were to those of the target user [Su and Khoshgoftaar, 2009]. Similarity is often calculated using Pearson correlation, vector cosine similarity or locality sensitive hashing [Su and Khoshgoftaar, 2009], giving a computationally inexpensive solution. However, combining decisions weighted by similarity is a heuristic approach to which the theoretical guarantees of Equations 2.2 and 2.3 do not necessarily apply. In contrast, statistical methods use principles such as Bayes' theorem to obtain a meaningful estimate of confidence in an aggregate decision.

2.2.2 Weighted Products and LogOPs

A drawback of the product rule (Equation 2.6) is that mis-calibrated agents may contribute too much or too little to the combined decision. A weighted product addresses this problem, relaxing the need to rely on agents to contribute trustworthy pieces of evidence to the aggregate decision. The general form of a weighted product is:

$$\hat{p}(t_i = j) = \frac{1}{Z} \prod_{k=1}^{K} (b_j^{(k)})^{w_k}, \tag{2.17}$$

where $Z = \sum_{\iota=1}^{J} \prod_{k=1}^{K} (b_\iota^{(k)})^{w_k}$ is the normalising constant. The weights act to calibrate the agents' decisions, so that agents whose responses appear over-confident or overly uncertain are corrected [Lindley, 1983]. We can also down-weight highly correlated agents to avoid counting the same piece of evidence multiple times. The weighted beliefs $(b_j^{(k)})^{w_k}$ should reflect the relative importance of the evidence provided by each agent, given the evidence

provided by the others.

When the agents' decisions are probabilities, the weighted product is also known as the *Logarithmic Opinion Pool (LogOP)* [Bordley, 1982; Givens and Roback, 1999]. This term is used because taking logarithms of Equation 2.17 results in a weighted sum form, similar to that of a linear opinion pool (LinOP):

$$\ln \hat{p}(t_i = j) = \sum_{k=1}^{K} w_k \ln b_j^{(k)} - \ln Z \tag{2.18}$$

Unlike the LinOP, the weights are not constrained to be non-negative nor to sum to one. Unlike LinOPs, LogOPs and do not treat agents as competing models and do not require that they output calibrated probabilities.

Various methods exist for determining the weights, w_k [Benediktsson and Swain, 2002; Genest and Zidek, 1986; Heskes, 1998]. Estimating the weights directly through optimisation is equivalent to discriminative classification techniques such as logistic regression [Kahn, 2004]. Alternatively, we can follow a generative derivation where the weights form part of our model of an agent, as in the Naïve Bayes LogOP, detailed below.

The *Product of Experts* method, proposed by [Hinton, 1999], is similar in form to a LogOP, but takes a different approach. Rather than optimise the weights of the combination function, it uses uniform weights of 1 and trains independent classifiers cooperatively to optimise performance of the product of experts.

Naïve Bayes LogOP

The weights of a logarithmic opinion pool can be learnt in a probabilistic manner by applying the *naïve Bayes* assumption. Naïve Bayes refers to the use of Bayes' theorem with the simplifying assumption that the observed variables – in this case, the agents' decisions – are independent conditioned on the target label. Despite this strong assumption, naïve Bayes methods can produce reliable decisions in many cases [Zhang, 2004]. The naïve Bayes LogOP (NB-LogOP) uses this idea to determine weights, ignoring any correlations between agents and assuming constant error rates over time. The equations below follow

the derivation of the weights given by [Kahn, 2004].

First, we re-express the problem of predicting the target label in terms of the log-odds, $a_{i,j}$, for object i and target value j, which is given by

$$a_{i,j} = \ln\left(\frac{p(t_i = j)}{1 - p(t_i = j)}\right). \tag{2.19}$$

Each agent, k, produces a belief, $b_{i,j}^{(k)}$, that j is the correct decision for object i. This can also be expressed as log-odds $a_{i,j}^{(k)}$:

$$a_{i,j}^{(k)} = \ln\left(\frac{b_{i,j}^{(k)}}{1 - b_{i,j}^{(k)}}\right). \tag{2.20}$$

NB-LogOP assumes that the log-odds, $a_i^{(k)}$, has a Gaussian distribution depending on the target label t_i:

$$p\left(a_{i,j}^{(k)} | t_i = j\right) = N\left(\mu_j^{(k)}, \sigma^{(k)2}\right) \tag{2.21}$$

$$p\left(a_{i,j}^{(k)} | t_i \neq j\right) = N\left(-\mu_j^{(k)}, \sigma^{(k)2}\right). \tag{2.22}$$

To simplify the model, we assume the variance $\sigma^{(k)2}$ is constant for both conditions $t_i = j$ and $t_i \neq j$. The means $\mu_j^{(k)}$ and $-\mu_j^{(k)}$ have the same magnitude, and $\mu_j^{(k)}$ therefore represents the *contrast mean* between the target values [Kahn, 2004], i.e. half the difference between the means for each condition $t_i = j$ and $t_i \neq j$. The contrast mean can be estimated by taking the sample mean from labelled training examples:

$$\mu_j^{(k)} = \frac{1}{N_{tr}} \sum_{i=1}^{N_{tr}} a_{i,j}^{(k)} \delta(t_i - j) - a_{i,j}^{(k)} \big(1 - \delta(t_i - j)\big), \tag{2.23}$$

where N_{tr} is the number of training samples and $\delta(t_i - j)$ is a delta function, equal to 1 if $t_i = j$ and 0 otherwise. The variance, $\sigma^{(k)2}$, can be then be estimated from the training

examples given $\mu_j^{(k)}$:

$$\sigma^{(k)2} = \frac{1}{N_{tr}} \sum_{i=1}^{N_{tr}} \left(a_{i,j}^{(k)} - \delta(t_i - j)\mu_j^{(k)} + \big(1 - \delta(t_i - j)\big)\mu_j^{(k)} \right)^2 . \tag{2.24}$$

We can use Bayes' theorem to find the posterior distribution over the target label:

$$\begin{aligned} p\left(t_i = j | a_{i,j}^{(1)}, .., a_{i,j}^{(K)}\right) &= \frac{1}{Z} p\left(t_i = j, a_{i,j}^{(1)}, .., a_{i,j}^{(K)}\right) \\ &= \frac{1}{Z} p(t_i = j) \prod_{k=1}^{K} p\left(a_{i,j}^{(k)} | t_i = j\right), \end{aligned} \tag{2.25}$$

in which Z is the normalising constant

$$Z = p\left(t_i = j, a_{i,j}^{(1)}, .., a_{i,j}^{(K)}\right) + p\left(t_i \neq j, a_{i,j}^{(1)}, .., a_{i,j}^{(K)}\right), \tag{2.26}$$

and each agent's log-odds has a Gaussian distribution function:

$$\begin{aligned} \hat{p}\left(a_{i,j}^{(k)} | t = j\right) &= \frac{1}{\sigma^{(k)}\sqrt{2\pi}} \exp\left(\frac{-\left(a_{i,j}^{(k)} - \mu_j^{(k)}\right)^2}{\sigma^{(k)2}} \right) \\ &= \frac{1}{\sigma^{(k)}\sqrt{2\pi}} \exp\left(\frac{-a_{i,j}^{(k)2} - \mu_j^{(k)2}}{\sigma^{(k)2}} \right) \exp\left(\frac{2\mu_j^{(k)} a_{i,j}^{(k)}}{\sigma^{(k)2}} \right) \end{aligned} \tag{2.27}$$

$$\hat{p}\left(a_{i,j}^{(k)} | t_i \neq j\right) = \frac{1}{\sigma^{(k)}\sqrt{2\pi}} \exp\left(\frac{-a_{i,j}^{(k)2} - \mu_j^{(k)2}}{\sigma^{(k)2}} \right) \exp\left(\frac{-2\mu_j^{(k)} a_{i,j}^{(k)}}{\sigma^{(k)2}} \right) \tag{2.28}$$

Since the two Gaussian distributions have means $\mu_j^{(k)}$ and $-\mu_j^{(k)}$, a number of terms cancel when Equations 2.27 and 2.28 are substituted into Equations 2.25 and 2.26. This results in an estimated log posterior distribution given by:

$$\begin{aligned} \ln \hat{p}(t_i = j | a_{i,j}^{(1)}, .., a_{i,j}^{(K)}) &= \ln p(t_i = j) + \sum_{k=1}^{K} \frac{2\mu_j^{(k)} a_{i,j}^{(k)}}{\sigma^{(k)2}} - \ln Z \\ &= \ln p(t_i = j) + \sum_{k=1}^{K} w_j^{(k)} a_{i,j}^{(k)} - \ln Z, \end{aligned} \tag{2.29}$$

where each agent k has weights for each decision value j, given by

$$w_j^{(k)} = \frac{2\mu_j^{(k)}}{\sigma^{(k)2}}, \tag{2.30}$$

and the normalisation constant is

$$Z = p(t_i = j) \exp\left(\frac{2\mu_j^{(k)} a_{i,j}^{(k)}}{\sigma^{(k)2}}\right) + p(t_i \neq j) \exp\left(\frac{-2\mu_j^{(k)} a_{i,j}^{(k)}}{\sigma^{(k)2}}\right). \tag{2.31}$$

The weights are dimensionless and need not sum to unity since they are used to weight the log-odds terms, which are themselves unbounded real-valued variables. Furthermore, the subtraction of the normalisation term calculated using the weights ensures that for any weight values, $\ln \hat{p}(t_i = j | a_{i,j}^{(1)}, .., a_{i,j}^{(K)}) < 0$ and thus $0 \leq \hat{p}(t_i = j | a_{i,j}^{(1)}, .., a_{i,j}^{(K)}) \leq 1$.

Equation 2.25 gives us a weighted LogOP combination function of the same form as Equation 2.18, but with the addition of the prior $p(t = j)$, which has the same effect as an additional agent with weight 1. The Naïve Bayes LogOP is thus one derivation of the LogOP from a generative model, where the agents are weighted according to their behaviour over training samples.

Equations 2.21 and 2.22 assume that agents are unbiased. A bias is a constant shift in the log-odds, $a_{i,j}^{(k)}$, that favours one decision over others. A bias decreases the weights $w_j^{(k)}$, since the variance $\sigma^{(k)2}$ in $a_{i,j}^{(k)}$ would be higher according to Equation 2.24. Thus, bias would reduce the contribution of an informative agent to the combined decision. A method for correcting bias was proposed by [Kahn, 2004], but this depends on learning accurate sample means from training data for each target value.

2.2.3 Supra-Bayesian Methods

A *Supra-Bayesian* combiner differs from methods described above, as it treats agents' outputs as raw data, rather than as predictions of the target decision. The Supra-Bayesian learns the likelihood $p(\boldsymbol{b}|t_i = j)$ of the agents' outputs $\boldsymbol{b}$, and combines this with its own prior belief $p(t_i = j)$ using Bayes' theorem, taking the same approach as general-purpose

Bayesian classifiers [Jacobs, 1995; Genest and Zidek, 1986]:

$$p(t_i = j|\boldsymbol{b}) = \frac{p(t_i = j)p(\boldsymbol{b}|t_i = j)}{\sum_{\iota=1}^{J} p(t_i = \iota)p(\boldsymbol{b}|t_i = \iota)} \tag{2.32}$$

Therefore, these methods can naturally handle biased and mis-calibrated agents, or agents that produce the opposite of the expected response, so long as the likelihood function learnt during training remains valid. Supra-Bayesian methods can be developed to deal with agents that output discrete labels, continuous values, or ranked lists. In principle, the likelihood function could be designed to cope with dependencies between agents [Genest and Schervish, 1985; Benediktsson and Swain, 2002; Genest and Zidek, 1986; Givens and Roback, 1999]. However, modelling dependencies is computationally expensive and complex to implement, so approximate models are usually preferred [Jacobs, 1995; Givens and Roback, 1999], such as naïve Bayes, where agents' decisions are assumed to be conditionally independent [Dawid and Skene, 1979; Genest and Schervish, 1985; Genest and Zidek, 1986]. The motivations for supra-Bayesian decision makers underpin the Bayesian approach in Section 2.4, which extends the Supra-Bayesian idea to give a fully Bayesian treatment to all model parameters.

2.2.4 Sample Space Partitioning

The supervised methods described so far involved learning a function that adapts to the overall reliability or importance of the information presented by each agent. However, each agent may have expertise in a specific subset of decision-making tasks. Examples include human experts in different fields of Medicine and automated handwriting recognition systems trained on different languages. Several established approaches predict the error rate of agents for each data point using input patterns to find similar data points in the training set, for which agents' performance has been observed [Wolpert, 1992; Woods et al., 2002; Liu and Yuan, 2001; Kuncheva, 2002]. In particular, *Mixtures of Experts* [Jacobs et al., 1991] produces soft sample space partitions by adjusting a weighted sum according to the input

pattern, so that experts in a particular partition are weighted more highly. Samples can also be partitioned according to difficulty of classification, which can be inferred from the disagreement among classifiers in that area [Ho et al., 2002]. Whilst sample space partitioning potentially allows us to focus on a small, efficient group of experts for each partition, a reliable set of input patterns is required for dividing the sample space in an effective manner. *Locality-based methods* try to avoid this difficulty by partitioning samples using the outputs of the agents rather than the original input features [Tulyakov et al., 2008]. Thus, the weights for a particular data point are determined from the agents' performance on training samples that had a similar set of responses [Giacinto and Roli, 2001].

Samples can also be partitioned according to the state of the environment, which may follow a dynamic process, for example, when a group of agents are visually tracking a moving object. The combination function can be altered according to a hidden Markov model [Hovland and McCarragher, 2002] or a Partially Observable Markov Decision Process (POMDP) [Spaan and Lima, 2009].

2.3 Unsupervised Methods

Supervised methods learn a combination function from a set of training examples, which is not possible for tasks where a labelled dataset is unavailable. Unsupervised methods, however, can find latent structure in unlabelled agent response data that may be useful for decision combination.

Clustering techniques are unsupervised methods for finding groups of related items from their associated data [Jain et al., 1999]. Agents can be clustered according to their responses to find behavioural groupings, from which we can select informative subsets of agents and ignore those that do not contribute to effective combined decisions. When a large pool of agents is available, reducing the number of agents in a combination can reduce communication and computation costs, and allow different subsets of agents to work on independent tasks.

Several clustering strategies have been used to differentiate agents by grouping those

with similar responses. For example, [Tsoumakas et al., 2004] assumes that agents within the same behavioural cluster have similar specialisms. In situations where each agent is only informative in a small portion of a large problem space, each cluster can be assigned to a sub-domain using additional knowledge about the agents' suitability. For example, find suitable data points for each group of agents, [Tsoumakas et al., 2004] assumes that samples from one source database have similar characteristics, then selects data points from sources where at least one member of the cluster is known to perform reliably. A similar idea is used in collaborative filtering [Su and Khoshgoftaar, 2009], where the challenge is to predict a target user's decisions. Users can be clustered according to the similarity of their decisions, so that when we wish to predict a target user's decisions, we only need consider the decisions made by agents in the same cluster. This avoids the need to learn and run a supervised combination function over the entire set of system users, which may be too costly.

Clustering can also be applied to data points to perform sample-space partitioning, grouping items that have similar sets of decisions from agents [Hofmann and Puzicha, 1999; Si and Jin, 2003]. The cluster of an item influences the choice of agent cluster to take decisions from, allowing agents to be selected for tasks that suit their abilities.

2.3.1 Clustering Informative Agents

In general, clustering approaches aim to find an informative subset of agents for a particular decision making task. Agents may also become more or less informative over time, for example, as a physical object moves between sensors. Hence, the clustering may need to be updated in real-time. The remainder of this section demonstrates this idea using a simulation of homogeneous agents predicting a common target decision. Each data point was classified sequentially by all agents. The aim is to separate agents into two clusters at each data point: *informative* and *uninformative* agents. Identifying the two clusters requires additional knowledge about how each group is likely to behave. In this case, we expect uninformative agents to make uncertain predictions or to have little agreement with

other agents, while informative agents should appear as a group of agents that mainly agree.

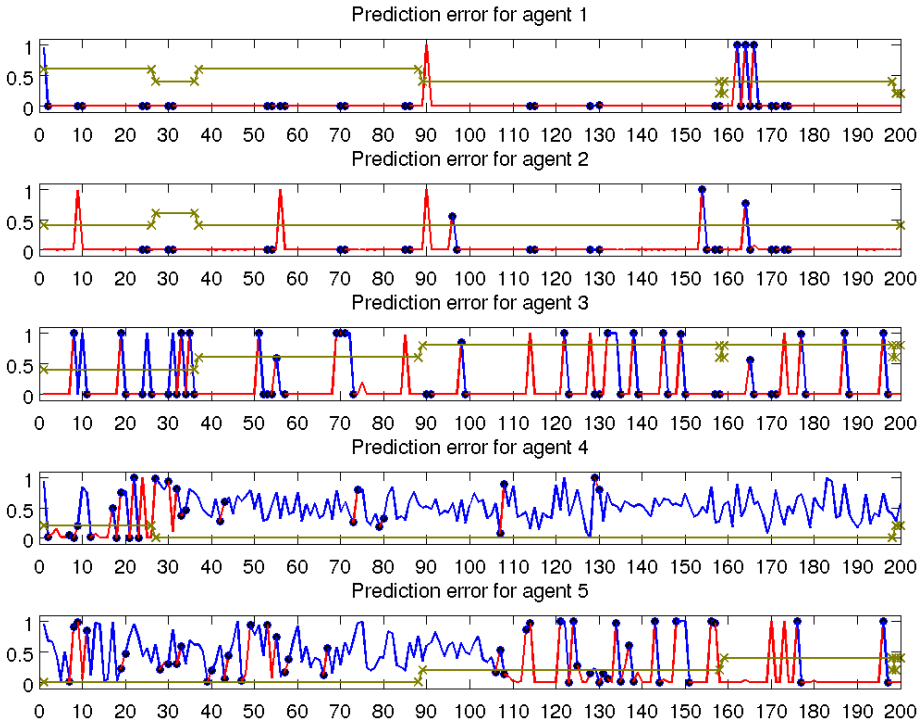
Five agents were simulated using logistic regressors [Bishop, 2006], which perform a binary classification over 200 data points. Two agents were given *informative* but noisy input data, simulating real-world sensors, while three *uninformative* agents were trained using random data. Each agent was switched between the informative and uninformative states at randomly chosen points. To simulate how agents adapt to changing sensors, agents were retrained after classifying each data point using the previous 20 data points.

For each data point i, an agent k produces a belief $b_i^{(k)} = p(t_i = 1)$ that the correct decision is $t_i = 1$. After observing the agents' decisions for data point i, a set of beliefs $\boldsymbol{B}_i$ (described below) for all K agents are clustered using *k-means* clustering [Jain et al., 1999], where the number of clusters is set to $k = 2$. At each iteration, the cluster means are initialised by taking the cluster members from the previous data point and averaging their responses $\boldsymbol{B}_i$ for the current data point. Thus, agents are expected to remain in the same cluster as long as they continue to agree.

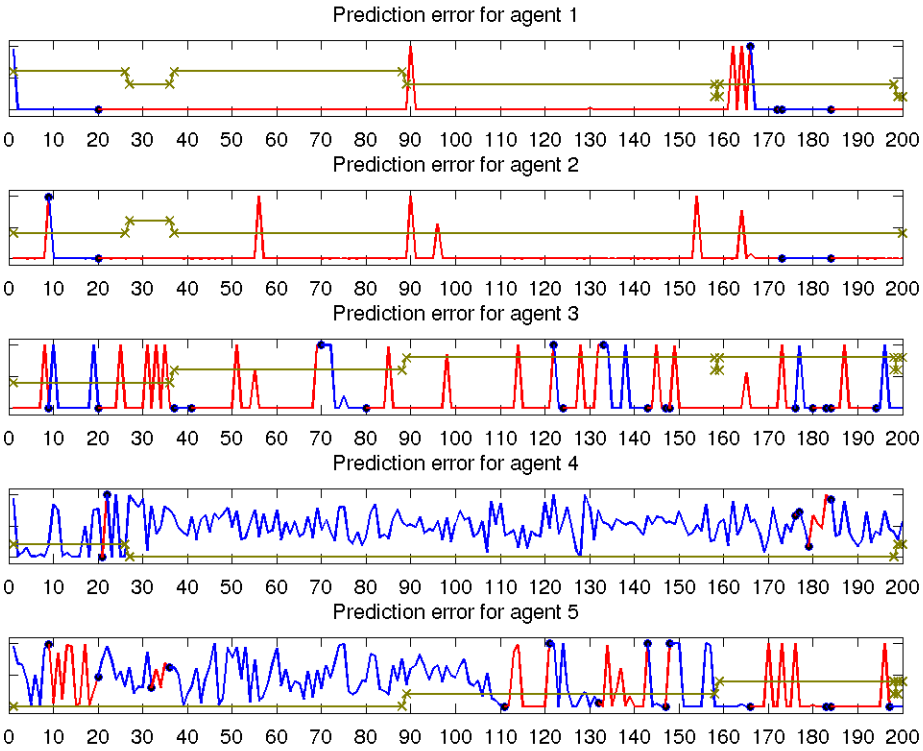
Four variants are tested using different sets of beliefs, $\boldsymbol{B}_i$. The first three variants use sliding windows of size $s = 1$, $s = 10$ and $s = 20$, where clustering is performed over the previous s belief values for each agent, so that $\boldsymbol{B}_i = \{b_\iota^{(k)} | k \in \{1, .., K\}, \iota \in \{i - s + 1, i - s + 2, .., i\}\}$. The fourth method weights the values in the sliding window to take into account the decreasing relevance of older data points. The decision $b_{i-t}^{(k)}$ at point $i - t$ is converted to a weighted value $\hat{b}_{i-t}^{(k)}$ according to:

$$\hat{b}_{i-t}^{(k)} = \begin{cases} b_{i-t}^{(k)} \left(1 - \frac{t}{s}\right), & t \leq s \\ 0, & t \geq s. \end{cases} \tag{2.33}$$

The relevance of data decreases with age, as the probability of a change since data point $i - s$ to sensors, environment or agent increases. Therefore, the probability that agents are similar at point i given they were similar at point $i - s$ decreases. However, while a smaller window size s should increase the speed at which changes are detected, it also increases the influence of individual errors.

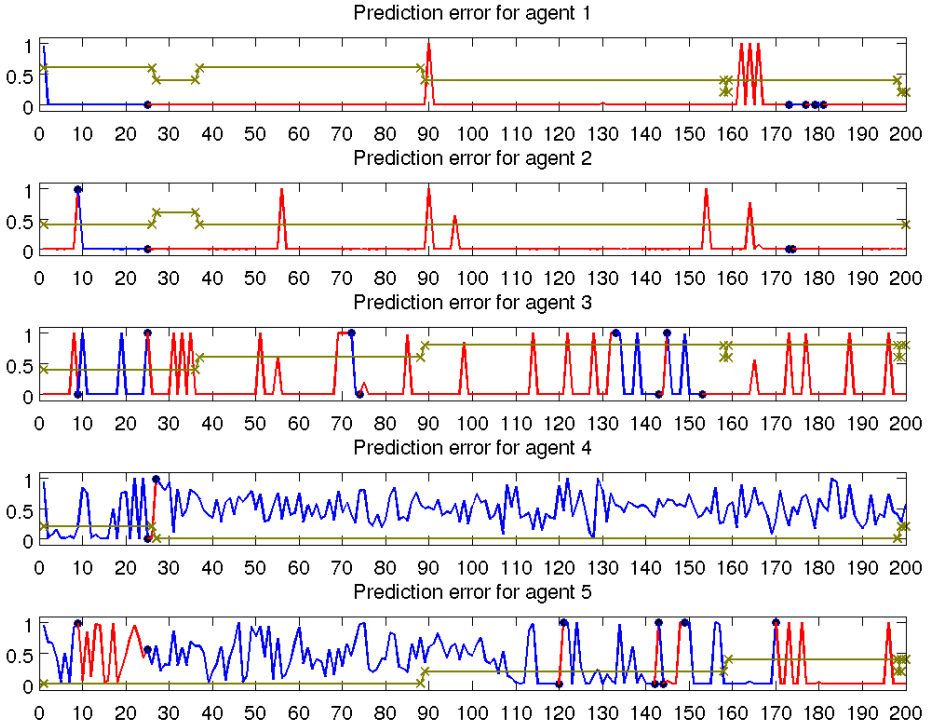


(a) $s = 1$

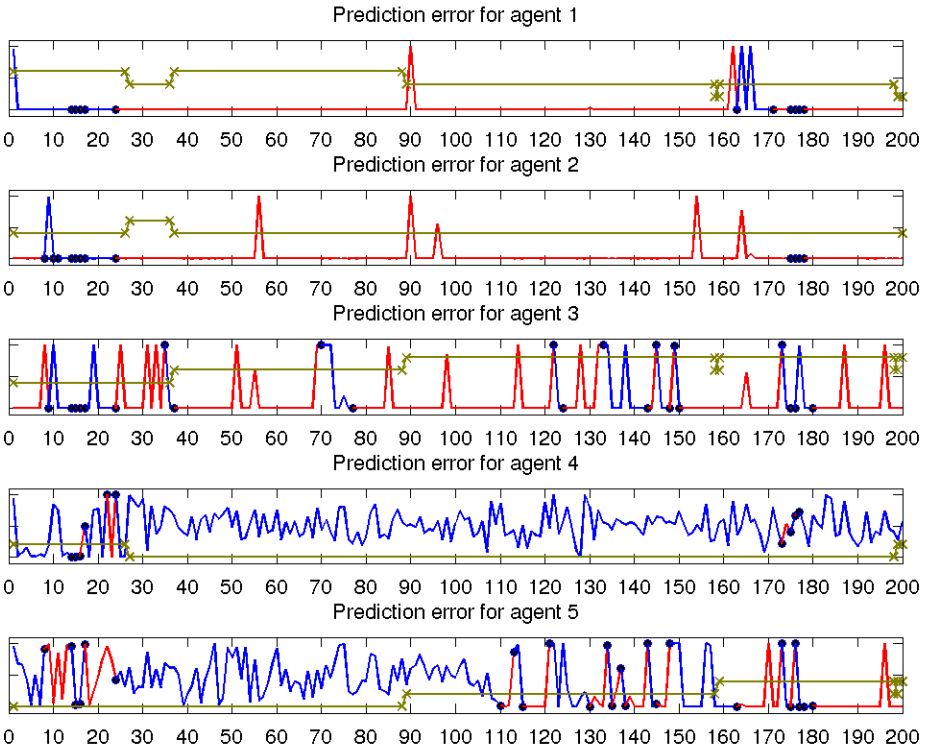


(b) $s = 10$

Figure 2.1: K-means clustering on $b_i^{(k)}$, where s is the sliding window size. The red/blue line shows the absolute difference between the prediction $b_i^{(k)} - t_i$ of each individual agent k at each data point i, where t_i is the target label. The red and blue colouring corresponds to the cluster at that time, with points added to the line to highlight cluster changes. The yellow-brown line indicates fraction of input sensors for the agent that are *informative*, with 'x' symbols marking sensor state changes.



(a) $s = 20$



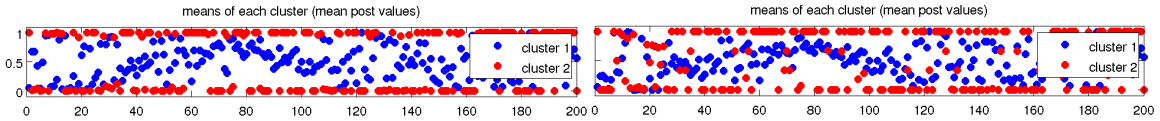
(b) $s = 20$, weighted sliding window.

Figure 2.2: K-means clustering on $b_i^{(k)}$, where s is the sliding window size. The red/blue line shows the absolute difference between the prediction $b_i^{(k)} - t_i$ of each individual agent k at each data point i, where t_i is the target label. The red and blue colouring corresponds to the cluster at that time, with points added to the line to highlight cluster changes. The yellow-brown line indicates fraction of input sensors for the agent that are *informative*, with 'x' symbols marking sensor state changes.

Figures 2.1 and 2.2 track the cluster membership of each agent over time using the four variants of the clustering method on a single dataset. Agents 1, 2 and 3 therefore have informative input data throughout; agents 4 has only a small number of informative inputs up to data point 25; agent 5 has no informative inputs at the start but becomes more informed at changes around 89 and 159 data points. All four clustering variants shown in Figures 2.1 and 2.2 place the informed agents 1, 2, 3 in predominantly in the red cluster, along with agent 5 after point 160, while uninformed agents are mainly in the blue cluster. However, differing patterns are observed for each of the sliding window variants.

Figure 2.1a shows clustering over the most recent responses only. In this case, random errors in one agent's decisions cause informed agents to change to the uninformed cluster, or vice versa. For example, note the spike at data point 97 for agent 2 causing it to move to the uninformative blue cluster despite being in the informative state. Similarly, uninformed agents are sometimes grouped incorrectly when they produce posteriors close to 0 or 1. An example in is agent 5 at data point 39. These rapid cluster changes are visibly reduced in Figures 2.1b to 2.2b, which show the results of using larger sliding windows. With $s = 20$, there is some reduction in the number of changes compared to $s = 10$, e.g. for data points after 130 for agent 3. In some cases $s = 20$ requires longer for a state change to cause a change in membership, e.g. agents 1 and 2 for data points between 10 and 25. For the data points 10 to 20 with $s = 10$, and 10 to 25 with $s = 20$, the blue cluster corresponds to the informed agents, while before and after the red cluster is informed, indicating inconsistent cluster labelling.

The weighted sliding window shown in Figure 2.2b shows more membership switches than with unweighted windows of either $s = 20$ or $s = 10$. Examples of an increased number of changes are agents 2 and 3 for data points 0 to 20, and agent 5 for data points 110 to 150. During this latter period, unweighted clustering with $s = 20$ mostly places agent 5 in the uninformed group, while $s = 10$ and weighted $s = 20$ are less certain. The correct classification is unclear since agent 5 produces many errors yet is not completely uninformed. Using a weighted sliding window increases the effect of recent random errors



(a) $s = 1$ (b) $s = 20$

Figure 2.3: Mean values of $b_i^{(k)}$ for each cluster at the current data point, i. The value s indicates the size of the sliding window over data points used for clustering.

as they are weighted more highly, while decreasing the effect of older data points from before a sensor change. The increased number of cluster changes compared to unweighted $s = 10$ is most likely because the most heavily weighted data points change at each data point with the weighted variant.

Figure 2.3 shows that clusters produced using either $s = 1$ or $s = 20$ clearly distinguish between confident and uncertain agents in most cases. However, Figure 2.3b indicates that some mean values of the red cluster are close to 0.5. These are data points where members of the red cluster made opposing decisions, resulting in a mean close to 0.5; these agents were still clustered together because of similar values for the previous $s = 20$ data points. So by using a sliding window, the informative cluster now corresponds more consistently with the informative agents, but does not remove isolated random errors in these agents.

This simulation shows how K-means clustering using a sliding window of agents' decisions can effectively separate informed and uninformed agents. The latter group may include unreliable agents, those making random guesses, or sensors that cannot currently observe a moving target. An extension to this idea is to increase the number of clusters so that we have several groups of informative agents, from which a diverse set of agents can be chosen to maximise the accuracy of combined decisions according to Equation 2.3.

In the simulation, using a sliding window size $s = 10$ produced more qualitatively meaningful clusters than setting $s = 1$, as the clusters were less affected by random errors in the agents' decisions. Although discounting older data can be motivated intuitively since older data is less likely to be relevant, the weighting scheme tested here is less responsive to state changes and more susceptible to isolated errors than an unweighted sliding window. The data that influences the clustering should therefore be optimised so that state changes

are detected quickly, yet we make use of all relevant data points to reduce the effect of random errors. Chapter 4 addresses the issue of selecting data for clustering by using a dynamic Bayesian method that learns from all observed data in an unsupervised manner. This approach explicitly models a dependency between states from one data point to the next, so that isolated errors do not cause an immediate major state transition, yet data from prior to earlier state changes does not have a direct effect on the current state.

The unsupervised methods discussed in this section show how latent structure provides useful information for decision combination that could supplement training data, especially when the latter is in limited supply. The following section introduces a decision combination approach that combines the benefits of both supervised and unsupervised learning.

2.4 Bayesian Classifier Combination

The fully Bayesian approach to decision combination is similar to the idea of a Supra-Bayesian, introduced in Section 2.2. Both concepts employ a generative model that treats agents' decisions as data, and learn a likelihood function over each agent's responses, combining them using Bayes' theorem. This learning step automatically handles errors and over-confident predictors, and can incorporate heterogeneous information other than predictions of the target decision. As described in Chapter 1, a fully Bayesian treatment allows us to combine new evidence with prior knowledge about agents' behaviour, and accounts for uncertainty over model parameters when making predictions. This is particularly important when training data is insufficient to confidently learn all model variables. The Bayesian inference methods described in this section can augment limited training data by exploiting latent structure in unlabelled data, so can operate in a *semi-supervised* manner, or if no training data is available, in unsupervised mode.

This section focuses specifically on Independent Bayesian Classifier Combination (IBCC), a generative model of agents' decision-making behaviour described by [Ghahramani and Kim, 2003] and based on an earlier idea by [Dawid and Skene, 1979]. IBCC combines discrete, categorical responses, such as labels produced by human decision mak-

ers, but could be extended to handle continuous values. While the original approach optimised point estimates of model parameters using *Expectation Maximisation* [Dempster et al., 1977], IBCC extends this to allow Bayesian inference techniques, such as Gibbs' sampling, as suggested in [Ghahramani and Kim, 2003]. The following subsections first explain the IBCC model in detail, then discuss its relationship to other decision combination methods. The final parts of this section discuss Bayesian inference over IBCC and present a Gibbs' sampling algorithm.

2.4.1 IBCC Model

Assume there are N data points or objects, for which we wish to infer a set of target labels $\boldsymbol{t} = \{t_1, t_2, .., t_N\}$. The target label t_i for the data point i takes a value $j \in \{1, .., J\}$. For some data points i, the target value t_i may be known, in which case i is a training example. Target labels are assumed to be drawn from a multinomial distribution with probability $p(t_i = j) = \kappa_j$. The parameter vector $\boldsymbol{\kappa} = [\kappa_1, .., \kappa_J]$ has a conjugate Dirichlet prior with hyperparameters $\boldsymbol{\nu}_0 = [\nu_{0,1}, .., \nu_{0,J}]$.

Given a set of K agents, each agent $k \in \{1, .., K\}$ can provide a response $c_i^{(k)} = l$ for data point i, where $l \in \{1, .., L\}$ is the set of discrete responses that the agents can make. The responses are assumed to be nominal labels that are not related to each other according to their ordering. Agents' responses may be taken from a set other than the set of target labels, hence the distinction between J, the number of target label values or classes, and L, the number of agent responses. For example, agents may identify features of a target object i rather than the predicting the target decision. Alternatively, human decision makers that predict the target decision may select from an additional "uncertain" category when they are unable to decide. The model therefore places minimal requirements on the type of agents we can include.

IBCC assumes that a response $c_i^{(k)}$ from agent k is generated by a multinomial distribution, taking the value l with probability $\pi_{j,l}^{(k)} = p\left(c_i^{(k)} = l | t_i = j\right)$. Thus, for a given target value, j, the distribution over responses from k has a parameter vector $\boldsymbol{\pi}_j^{(k)}$. Considering

all target values $j \in \{1, .., J\}$, the parameters for agent k can be seen as a *confusion matrix*, $\boldsymbol{\Pi}^{(k)}$, which expresses the dependency between the agent's responses and the target labels:

$$\boldsymbol{\Pi}^{(k)} = \begin{bmatrix} \pi_{1,1}^{(k)}, & \ldots, & \pi_{1,L}^{(k)} \\ \vdots, & \ddots, & \vdots \\ \pi_{J,1}^{(k)}, & \ldots, & \pi_{J,L}^{(k)} \end{bmatrix} \tag{2.34}$$

Each row $\boldsymbol{\pi}_j^{(k)}$ is assumed to be independent of the other rows in the matrix. This explicitly models the situation where agents are less accurate at predicting one of the target values. For example, if classifying birds from photographs, a well-known bird may be easily identified, while an unusual bird without distinctive features may be mis-classified more often. IBCC introduces conjugate prior distributions over $\boldsymbol{\pi}_j^{(k)}$, which is Dirichlet distributed with hyperparameters $\boldsymbol{\alpha}_{0,j}^{(k)}$. These hyperparameters form a matrix $\boldsymbol{A}_0^{(k)}$, where each row j is the vector $\boldsymbol{\alpha}_{0,j}^{(k)}$.

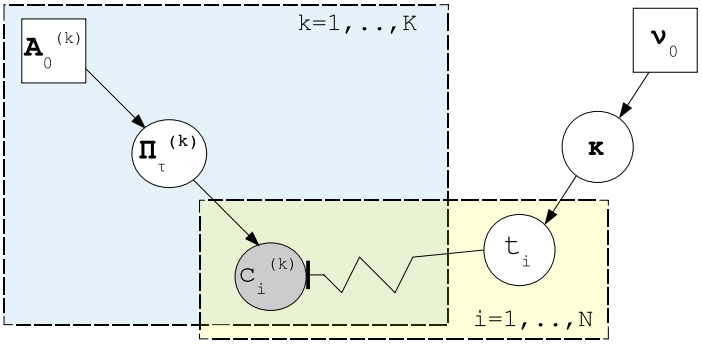


Figure 2.4: Graphical Model for IBCC. The shaded node represents observed values, circular nodes are variables with a distribution and square nodes are variables instantiated with point values. The zig-zag line means t_i is a switch that selects parameters from $\Pi^{(k)}$.

IBCC assumes that agents' responses are conditionally independent given the target labels. The joint distribution over all variables can therefore be written as

$$p\left(\boldsymbol{t}, \boldsymbol{c}, \boldsymbol{\kappa}, \boldsymbol{\Pi} | \boldsymbol{\nu}_0, \boldsymbol{A}_0\right) = \prod_{i=1}^{N} \left\{ \kappa_{t_i} \prod_{k=1}^{K} \pi_{t_i, c_i^{(k)}}^{(k)} \right\} p\left(\boldsymbol{\kappa} | \boldsymbol{\nu}\right) \prod_{j=1}^{J} \prod_{k=1}^{K} p\left(\boldsymbol{\pi}_j^{(k)} | \boldsymbol{\alpha}_{0,j}^{(k)}\right), \tag{2.35}$$

where $\boldsymbol{\Pi} = \left\{ \boldsymbol{\Pi}^{(k)} | k = 1, .., K \right\}$ is the set of confusion matrices for all agents, and

$\boldsymbol{A}_0 = \left\{\boldsymbol{A}_0^{(k)}|1,..,K\right\}$ is the corresponding set of hyperparameter matrices. The graphical model for IBCC is depicted in Figure 2.4. Empirical tests have shown that the performance of IBCC is comparable with that of more computationally-expensive variants that model dependencies between agents [Ghahramani and Kim, 2003], possibly because the more complex models have additional free parameters, so require a larger body of training data before an improvement over IBCC can be realised. For example, to reliably quantify a dependency between a pair of agents, we must be able to confidently identify enough errors and correct responses to measure the degree of correlation. In some situations, the accuracy of classifications is not affected by assuming conditional independence between each piece of evidence, despite this assumption being violated, because the dependencies do not alter the likelihoods enough to cause a change in the discrete combined decision, possibly because they cancel each other out [Zhang, 2004]. Dependencies can cancel if for each dependency that causes an increased belief in a particular target value, there is a dependency that causes a corresponding decrease due to the independence assumption. Increasing the number and diversity of agents can increase the chances of dependencies cancelling. In practice, we can also choose to exclude highly-correlated agents from the same data points. Assuming agent independence is therefore a reasonable trade-off between reducing the amount of training data and computation required, and precise modelling of agents.

The definition of IBCC given in [Ghahramani and Kim, 2003] places a further exponential distribution over $\boldsymbol{\alpha}_{0,j}^{(k)}$ with hyper-hyperparameters $\boldsymbol{\lambda_j}$. However, the exponential distribution is not conjugate to the Dirichlet distribution, which complicates analysis of the posterior distribution over the target labels and model variables. For instance, obtaining the posterior over $\boldsymbol{\pi}_j^{(k)}$ requires marginalising $\boldsymbol{\alpha}_{0,j}^{(k)}$, which cannot be done analytically:

$$p\left(\boldsymbol{\pi}_j^{(k)}|\boldsymbol{t},\boldsymbol{\lambda_j}\right) = \int p\left(\boldsymbol{\pi}_j^{(k)}|\boldsymbol{t},\boldsymbol{\lambda_j},\boldsymbol{\alpha}_{0,j}^{(k)}\right) \mathrm{d}\boldsymbol{\alpha}_{0,j}^{(k)}. \tag{2.36}$$

Using Gibbs' sampling to perform inference, as described in later subsections, involves drawing from the distribution $p\left(\boldsymbol{\alpha}_{0,j}^{(k)}|\boldsymbol{\pi}_j^{(k)},\boldsymbol{\lambda_j}\right)$, which requires an expensive adaptive re-

jection sampling step [Gilks and Wild, 1992]. However, this additional layer of uncertainty is not required. The Dirichlet distribution $p\left(\boldsymbol{\pi}_j^{(k)}|\boldsymbol{\alpha}_{0,j}^{(k)}\right)$ can capture any level of uncertainty over the values of $\boldsymbol{\pi}_{\boldsymbol{j}}^{(\boldsymbol{k})}$ through the choice of values for $\boldsymbol{\alpha}_{0,j}^{(k)}$, which can be understood intuitively as pseudo-counts of prior observations. The pseudo-counts $\boldsymbol{\alpha}_{0,j}^{(k)}$ can encode qualitative prior beliefs, such as that agents are more are more likely to be correct than incorrect. The magnitude of the pseudo-counts quantifies the strength of those beliefs; this strength of belief is equivalent to having observed the same number of real agent decisions. This chapter therefore proposes using point values for $\boldsymbol{\alpha}_{0,j}^{(k)}$ as in other comparable models [Choudrey and Roberts, 2003; Bishop, 2006; Penny and Roberts, 1999].

As well as providing a principled model from which to infer target labels $\boldsymbol{t}$, IBCC quantifies the decision-making behaviour of each agent using the *confusion matrices*, $\boldsymbol{\Pi}$. This information could enable us to identify uninformative agents for re-training or exclusion from future tasks. It could also allow expert decision makers to be assigned to data points that are highly uncertain. Chapter 5 shows how the information value of an agent's response to a particular data point can be predicted from the confusion matrices, enabling an intelligent task assignment mechanism.

2.4.2 Inference using Gibbs' Sampling

The posterior distribution over the unknown variables $\boldsymbol{t}$, $\boldsymbol{\Pi}$, and $\boldsymbol{\kappa}$, given a set of observed agent responses, $\boldsymbol{c}$, is given by:

$$\begin{aligned} p(\boldsymbol{t}, \boldsymbol{\kappa}, \boldsymbol{\Pi}|\boldsymbol{c}, \boldsymbol{\nu}_0, \boldsymbol{A}_0) &= \frac{p(\boldsymbol{t}, \boldsymbol{c}, \boldsymbol{\kappa}, \boldsymbol{\Pi}|\boldsymbol{\nu}_0, \boldsymbol{A}_0)}{\iiint p(\boldsymbol{t}, \boldsymbol{c}, \boldsymbol{\kappa}, \boldsymbol{\Pi}|\boldsymbol{\nu}_0, \boldsymbol{A}_0) \mathrm{d}\boldsymbol{t}\ \mathrm{d}\boldsymbol{\Pi}\ \mathrm{d}\boldsymbol{\kappa}} \\ &= \frac{p(\boldsymbol{t}, \boldsymbol{c}, \boldsymbol{\kappa}, \boldsymbol{\Pi}|\boldsymbol{\nu}_0, \boldsymbol{A}_0)}{\mathbb{E}_{\boldsymbol{\Pi}, \boldsymbol{\kappa}, \boldsymbol{t}}[p(\boldsymbol{c}|\boldsymbol{\nu}_0, \boldsymbol{A}_0)]} \end{aligned} \tag{2.37}$$

The term $\mathbb{E}_{\boldsymbol{\Pi}, \boldsymbol{\kappa}, \boldsymbol{t}}[.]$ refers to the expectation with respect to $\boldsymbol{t}$, $\boldsymbol{\Pi}$ and $\boldsymbol{\kappa}$, i.e. the expected value obtained by marginalising those variables. Inference over one set of variables is performed by marginalising the others from this distribution. To predict target labels $\boldsymbol{t}$, we

marginalise model parameters $\boldsymbol{\Pi}$ and $\boldsymbol{\kappa}$.

$$\begin{aligned} p(\boldsymbol{t}|\boldsymbol{c}, \boldsymbol{\nu}_0, \boldsymbol{A}_0) &= \frac{p(\boldsymbol{t}, \boldsymbol{c}|\boldsymbol{\nu}_0, \boldsymbol{A}_0)}{\int p(\boldsymbol{t}, \boldsymbol{c}|\boldsymbol{\nu}_0, \boldsymbol{A}_0) d\boldsymbol{t}} \\ &= \frac{\iint p(\boldsymbol{t}, \boldsymbol{c}, \boldsymbol{\kappa}, \boldsymbol{\Pi}|\boldsymbol{\nu}_0, \boldsymbol{A}_0) \mathrm{d}\boldsymbol{\Pi} \, \mathrm{d}\boldsymbol{\kappa}}{\iiint p(\boldsymbol{t}, \boldsymbol{c}, \boldsymbol{\kappa}, \boldsymbol{\Pi}|\boldsymbol{\nu}_0, \boldsymbol{A}_0) \mathrm{d}\boldsymbol{\Pi} \, \mathrm{d}\boldsymbol{\kappa} \, \mathrm{d}\boldsymbol{t}} \\ &= \frac{\mathbb{E}_{\boldsymbol{\Pi}, \boldsymbol{\kappa}}[p(\boldsymbol{t}, \boldsymbol{c}|\boldsymbol{\nu}_0, \boldsymbol{A}_0)]}{\mathbb{E}_{\boldsymbol{\Pi}, \boldsymbol{\kappa}, \boldsymbol{t}}[p(\boldsymbol{c}|\boldsymbol{\nu}_0, \boldsymbol{A}_0)]} \end{aligned} \tag{2.38}$$

Since each target label t_i in $\boldsymbol{t}$ has an independent categorical distribution, the probability that $t_i = j$ is equal to the expected value of t_i for category j, given by:

$$\begin{aligned} \mathbb{E}[t_i = j|\boldsymbol{c}, \boldsymbol{\nu}_0, \boldsymbol{A}_0] &= p(t_i = j|\boldsymbol{c}, \boldsymbol{\nu}_0, \boldsymbol{A}_0) \\ &= \frac{\mathbb{E}_{\boldsymbol{\Pi}, \boldsymbol{\kappa}}[p(t_i = j, \boldsymbol{c}|\boldsymbol{\nu}_0, \boldsymbol{A}_0)]}{\mathbb{E}_{\boldsymbol{\Pi}, \boldsymbol{\kappa}, \boldsymbol{t}}[p(\boldsymbol{c}|\boldsymbol{\nu}_0, \boldsymbol{A}_0)]} \end{aligned} \tag{2.39}$$

The expectations in Equations 2.37, 2.38 and 2.39 above cannot be obtained in closed form, so must be estimated using a method such as *Markov chain Monte-Carlo (MCMC)* sampling [Neal, 1993], as explained below.

MCMC is a type of *Monte Carlo* method, named after the famous casino destination for its use of randomly-drawn data points. Monte Carlo methods estimate an expectation by collecting simulated samples of a random variable X, then taking the sample mean of a function $g(X)$ of the random variable X:

$$\mathbb{E}_{\boldsymbol{X}}[g(X)] = \int g(x) f_X(x) dx \approx \tilde{g}(X) = \frac{1}{S} \sum_{i=1}^{S} g(x_i), \tag{2.40}$$

where $f_X(x)$ is the probability density or mass function of X, S is the number of samples and $\boldsymbol{x} = \{x_1, .., x_S\}$ is the set of samples. As the number of samples increases, the error in the estimate from the sample mean decreases. The limit case can be written as follows, according to the weak law of large numbers introduced by [Khintchine, 1929]:

$$\lim_{S \to \infty} p\big(\big|\tilde{g}(X) - \mathbb{E}_{\boldsymbol{X}}[g(X)]\big| \geq \epsilon\big) = 0, \tag{2.41}$$

for any positive error value ϵ. Samples from certain types of distribution can be simulated by mapping the output of a pseudo-random number generator to values that are draws from a probability density or mass function. When performing inference over models such as IBCC, it is possible to evaluate the value of either $f_X(x)$, or its unnormalised form, $\tilde{f}_X(x)$, where $f_X(x) = \frac{1}{Z}\tilde{f}_X(x)$. However, we cannot sample from $f_X(x)$ directly due to the original problem that the posterior distributions over the variables $\boldsymbol{t}$, $\boldsymbol{\Pi}$ and $\boldsymbol{\kappa}$ are not available in closed form. Markov-Chain Monte Carlo methods (MCMC) solve this problem by drawing samples from a simpler distribution that depends on the previous draw, so that the samples form a *Markov chain* with the desired distribution as its *stationary distribution*. A Markov chain is a system that transitions between states, in this case values of the variable x, where the next state depends only on the current state and not on any earlier states. The stationary distribution of a Markov chain is the distribution over states $\pi(x)$ that satisfies:

$$\pi(x') = \int_x \pi(x) T(x'|x) \mathrm{d}x, \tag{2.42}$$

where $T(x'|x)$ is the probability of transitioning from state x to x'. If T is chosen correctly, there is a unique distribution, π, and the distribution of the collected samples tends to π as the number of samples tends to infinity, regardless of the starting state x_0. A potential disadvantage of MCMC methods is the need to obtain large numbers of samples, potentially at a high computational cost, to avoid any effect from the initial value x_0, since the subsequent draws are correlated.

MCMC methods for obtaining samples from a distribution $f_X(x)$ require the transition distribution T to be designed in such a way that $f_X(x)$ is the stationary distribution. The *Metropolis-Hastings* algorithm provides one such MCMC method for obtaining samples from $f_X(x)$. Given an arbitrary starting value of x, a new value x' is proposed by drawing from an arbitrary symmetric probability distribution q, which dependent on x. Then, the sample x' is accepted with probability $a = \min\left(1, \frac{\tilde{f}_X(x')}{\tilde{f}_X(x)}\right)$, otherwise the previous sample is duplicated, setting $x' = x$.

Gibbs' sampling is a special case of MCMC for integrating over more than one random

variable simultaneously [Geman and Geman, 1984]. Each random variable X_i is sampled in turn, drawing from the proposal distribution q_i, which is the conditional distribution over X_i given the current samples for all other variables, $\boldsymbol{x}_{\neg i}$. The acceptance rate is $a = 1$. Gibbs' sampling is suited to models such as IBCC, where we wish to evaluate an expectation over multiple variables, and the conditional probabilities of those variables can be computed easily. Consider that the expectation over the target values in Equation 2.39 requires taking expectations with respect to $\boldsymbol{t}$, $\boldsymbol{\Pi}$ and $\boldsymbol{\kappa}$. Since IBCC places conjugate priors over the model parameters, the conditional distributions for all variables can be sampled without difficulty, as the explanation below will demonstrate[1].

To perform Gibbs' sampling we must first derive the conditional distributions for each variable. For $\boldsymbol{\kappa}$, we can write the conditional distribution in full as

$$p(\boldsymbol{\kappa}|\boldsymbol{t}, \boldsymbol{\nu}_0) = \frac{1}{Z} \prod_{i=1}^{N} \kappa_{t_i} . \frac{1}{\mathrm{B}(\boldsymbol{\nu}_0)} \prod_{j=1}^{J} \kappa_j^{\nu_{0,j}}, \tag{2.43}$$

where $\mathrm{B}(\boldsymbol{a}) = \frac{\prod_{l=1}^{L} \Gamma(a_l)}{\Gamma(\sum_{l=1}^{L} a_l)}$ is the Beta function, $\Gamma(a)$ is the Gamma function [Davis, 1965], and Z is a normalisation constant. We can simplify this by defining N_j as the number of target labels in $\boldsymbol{t}$ with target value $t_i = j$,

$$N_j = \sum_{i=1}^{N} \delta(t_i - j), \tag{2.44}$$

where $\delta(x)$ is a delta function defined as:

$$\delta(x) = \begin{cases} 1, x = 0 \\ 0, x \neq 0 \end{cases} . \tag{2.45}$$

[1]The use of Gibbs' sampling for IBCC was suggested in [Ghahramani and Kim, 2003] without giving the equations required to perform inference. The derivations in this thesis are therefore the work of the present author.

We can now replace the term $\prod_{i=1}^{N} \kappa_{t_i}$:

$$p(\boldsymbol{\kappa}|\boldsymbol{t}, \boldsymbol{\nu}_0) = \frac{1}{Z\mathrm{B}(\boldsymbol{\nu}_0)} \prod_{j=1}^{J} \kappa_j^{\nu_{0,j}+N_j}. \tag{2.46}$$

Recognising this as a Dirichlet distribution, we can write the parameters to the Dirichlet distribution as $\boldsymbol{\nu} = [\nu_1, .., \nu_J]$, where

$$\nu_j = \nu_{0,j} + N_j. \tag{2.47}$$

For completeness, replacing the normalisation constant Z in Equation 2.46 gives the full form of the Dirichlet distribution for $\boldsymbol{\kappa}$ conditioned on $\boldsymbol{t}$:

$$p(\boldsymbol{\kappa}|\boldsymbol{t}, \boldsymbol{\nu}_0) = \frac{1}{\mathrm{B}(\boldsymbol{\nu})} \prod_{j=1}^{J} \kappa_j^{\nu_j}. \tag{2.48}$$

The straightforward parameter update in Equation 2.47 arises because the Dirichlet prior over $\boldsymbol{\kappa}$ is the conjugate of the multinomial from which $\boldsymbol{t}$ was drawn. A similar pattern exists for the distribution over each row of each confusion matrix $\boldsymbol{\pi}_j^{(k)}$:

$$p\left(\boldsymbol{\pi}_j^{(k)}|\boldsymbol{t}, \boldsymbol{c}, \boldsymbol{\alpha}_{0,j}^{(k)}\right) = \frac{1}{Z} \prod_{i=1}^{N} \left(\pi^{(k)}_{t_i, c_i^{(k)}}\right)^{\delta(t_i-j)} \cdot \frac{1}{\mathrm{B}\left(\boldsymbol{\alpha}_{0,j}^{(k)}\right)} \prod_{l=1}^{L} \left(\pi_{j,l}^{(k)}\right)^{\alpha_{0,j,l}^{(k)}}. \tag{2.49}$$

We define $N_{j,l}^{(k)}$ as the number of responses of value l produced by agent k for data points with target value j:

$$N_{j,l}^{(k)} = \sum_{i=1}^{N} \delta\left(t_i - j\right) \delta\left(c_i^{(k)} - l\right). \tag{2.50}$$

We can substitute the response counts $N_{jl}^{(k)}$ into Equation 2.49 and re-write it as a Dirichlet

distribution:

$$p\left(\boldsymbol{\pi}_j^{(k)}|\boldsymbol{t},\boldsymbol{c},\boldsymbol{\alpha}_{0,j}^{(k)}\right) = \frac{1}{Z\mathrm{B}\left(\boldsymbol{\alpha}_{0,t_i}^{(k)}\right)}\prod_{l=1}^{L}\left(\pi_{j,l}^{(k)}\right)^{\alpha_{0,j,l}^{(k)}+N_{jl}^{(k)}}$$
$$= \frac{1}{\mathrm{B}\left(\boldsymbol{\alpha}_{t_i}^{(k)}\right)}\prod_{l=1}^{L}\left(\pi_{j,l}^{(k)}\right)^{\alpha_{j,l}^{(k)}}, \tag{2.51}$$

where $\boldsymbol{\alpha}_j^{(k)}$ is defined by:

$$\alpha_{jl}^{(k)} = \alpha_{0,j,l}^{(k)} + N_{j,l}^{(k)}. \tag{2.52}$$

We thereby update the prior pseudo-counts of the Dirichlet distributions by adding new observations. For the conditional distribution over each target label t_i we apply Bayes rule to obtain a categorical distribution:

$$p(t_i = j|\boldsymbol{\Pi},\boldsymbol{\kappa},\boldsymbol{c}) = \frac{p(\boldsymbol{c},t_i=j|\boldsymbol{\Pi},\boldsymbol{\kappa})}{\sum_{\iota=1}^{J}p(\boldsymbol{c},t_i=\iota|\boldsymbol{\Pi},\boldsymbol{\kappa})} = \frac{\kappa_j\prod_{k=1}^{K}\pi_{j,c_i^{(k)}}^{(k)}}{\sum_{\iota=1}^{J}\kappa_\iota\prod_{k=1}^{K}\pi_{\iota,c_i^{(k)}}^{(k)}}. \tag{2.53}$$

This form arises since the target labels are conditionally independent given $\boldsymbol{\kappa}$ and $\boldsymbol{\Pi}$.

IBCC-Gibbs Algorithm

A Gibbs sampling algorithm, *IBCC-Gibbs*, can then be used as follows to approximate the posterior expectations of the unknown variables $\boldsymbol{t}$, $\boldsymbol{\Pi}$ and $\boldsymbol{\kappa}$, given observations $\boldsymbol{c}$. In particular, we wish to predict each target label t_i by estimating Equation 2.39 from the mean of a set of Gibbs' samples. The algorithm proceeds as follows.

1. Initialise the values of $\boldsymbol{t}$, $\boldsymbol{\Pi}$ and $\boldsymbol{\kappa}$. It is possible to use a less costly method such as Expectation Maximisation [Dempster et al., 1977] to estimate the values. The choice of initialisation can affect the rate of convergence of the sampling algorithm. Any training examples, i.e. data points i where the values of t_i is already known, are fixed at these known values and are not sampled.

2. Draw a value for $\boldsymbol{\kappa}$ from the distribution given in Equation 2.48, using the current

value of $\boldsymbol{t}$. Replace the current value of $\boldsymbol{\kappa}$ with the new sample.

3. Sample $\boldsymbol{\pi}_j^{(k)}$ for $k = 1, .., K$ and $j = 1, .., J$ from the distribution given in Equation 2.51, using current $\boldsymbol{t}$. Replace current value of $\boldsymbol{\pi}_j^{(k)}$ with new samples.

4. Sample $\boldsymbol{t}$ from Equation 2.53 using current values of $\boldsymbol{\Pi}$ and $\boldsymbol{\kappa}$. Known training examples are fixed and are not sampled.

5. Record the current sample values in a list of sample values.

6. Calculate means of variables in the samples recorded so far to find current approximations to the expectation of each variable from the posterior distribution.

7. Repeat all steps from (2) until a sufficient approximation is obtained, or convergence is observed.

8. Outputs: the combined decisions for the target labels, given by the expected value $\mathbb{E}[t_i = j|\boldsymbol{c}, \boldsymbol{\nu}_0, \boldsymbol{A}_0]$ for all data points $i = 1, .., N$ and target values $j = 1, .., J$, estimated by taking the mean of the sample values collected.

While Gibbs' Sampling is guaranteed to converge to the true posterior in the long term, the stopping criteria are difficult to ascertain. If the posterior distribution is highly multi-modal, the Markov chain may appear to converge around a single mode when it has not yet explored others. However, once a large number of samples has been obtained, it may be sufficient in many cases to monitor the convergence of the sample means.

Unlike some traditional Machine Learning algorithms, IBCC-Gibbs does not require a separate training phase to learn the parameters before running a prediction phase to infer the unknown values of $\boldsymbol{t}$. Running the learning and prediction steps in one phase allows for semi-supervised learning, where the latent structure in the test samples can influence the predictions, as well as any training labels. The resulting expected values of target $\boldsymbol{t}$ are probabilities that take into account our uncertainty in the model parameters and our prior knowledge.

2.4.3 Relationships to other Combination Methods

We can relate IBCC to many of the decision combination methods described earlier in this chapter by their use of different assumptions. First, the posterior distribution over a target label t_i, given by Equation 2.53 for IBCC, takes a similar form to the weighted product or LogOP in Equation 2.17. The key difference is that IBCC converts discrete responses to probabilities using the confusion matrix, while weighted products use the beliefs supplied by agents directly, adjusting for bias and miscalibration using weights [Lindley, 1983]. Since IBCC treats the agents' responses as raw data, they need not be homogeneous decisions and may include feature values.

We can also compare IBCC to Equation 2.16 for the weighted sum, by taking the logarithm of Equation 2.53:

$$\ln p(t_i = j|\boldsymbol{\Pi}, \boldsymbol{\kappa}, \boldsymbol{c}) = \ln \kappa_j + \sum_{k=1}^{K} \ln \pi^{(k)}_{j,c_i^{(k)}} - \ln Z, \tag{2.54}$$

where $Z = \sum_{\iota=1}^{J} \kappa_\iota \prod_{k=1}^{K} \pi^{(k)}_{\iota,c_i^{(k)}}$ is the normalising constant. Equation 2.54 is a conditional distribution given known model parameter values. In contrast to the approaches described in Section 2.2, IBCC-Gibbs does not find a point estimate of the parameters, but integrates over possible values. In both Equation 2.54 for IBCC and Equation 2.16 for the weighted sum, contributions corresponding to each agent's responses are summed. The weighted sum combines the weighted responses directly, so treats each agent as a competing decision-making model. In contrast, IBCC and weighted products sum log likelihoods, so each agent contributes a separate piece of evidence.

The weighted sum method, BMA and the NB-LogOP discussed in Section 2.2 use constant weights for all decisions, whereas IBCC uses confusion matrices to model reliability that can vary depending on the target values and values of the agents' responses. Unlike weighted sums, weighted products and confusion matrices can also allow for negative weights when the target label is likely to be the opposite of an agent's prediction (e.g. malicious agents, faulty sensors). In summary, the confusion matrices of IBCC provide a more

expressive and flexible way to map agent responses to evidence of the target label than either weighted products or weighted sums. A further feature of IBCC is the additional κ_j term in Equation 2.54. An equivalent term arises in the Naïve Bayes LogOP in Equation 2.25. IBCC-Gibbs gives a fully Bayesian treatment of model parameters, marginalising over the prior distributions over the parameters κ_j and $\mathbf{\Pi}$.

This section described a number of theoretical advantages of IBCC and presented an inference algorithm using Gibbs' sampling. The following section demonstrates the empirical value of IBCC-Gibbs compared to other key methods discussed in this chapter.

2.5 Empirical Comparison of Methods

No.	Experiment Name	Description	Independent variable
1	Weak Agents	Weak agents have low individual performance.	Error rate of agents
2	Inconsistent Abilities	The reliability of agents' decisions varies between the two target values.	Error rate of agents for objects with target value $t_i = 1$
3	Noise	Uninformative agents produce noise for the combiner.	No. uninformative agents
4	Reversed Decisions	Some agents consistently make incorrect decisions.	Proportion of agents producing reversed decisions
5	Correlated Agents	Includes agents whose responses are correlated, including their errors.	Number of agents that are duplicates of agent 1
6	Training Data	Some combiners can learn from training data.	No. training examples

Table 2.1: Overview of simulated experiments comparing decision combination methods

This section provides an empirical comparison of decision combination methods in a variety of controlled scenarios. Each scenario introduces a different decision combination problem: weak agents; inconsistent abilities across target values; noise; reversed decisions; correlated errors; lack of training data. All the experiments involve evaluating the efficacy of the decision combination methods on simulated data as we vary a particular indepen-

dent variable. The experiments are intended to show how the different assumptions of the decision combination methods allow them to cope differently in each situation. Table 2.1 provides an overview of the experiments and the corresponding independent variables. Further discussion of the motivation and procedure for the individual experiments is provided below alongside the results of each experiment.

No. samples per dataset	1000
No. datasets for each variable setting	25
No. informative agents	5
Error Rate for informative agents	$0.846(\pm 0.154)$
No. uninformative agents	0
Error Rate for uninformative agents	$0.500(\pm 0.017)$

Table 2.2: Default settings for all experiments.

The experiments in this section share the same basic set-up. All involve repeating a binary classification problem for different settings of an independent variable. At each setting of the independent variable, the decision combination methods are tested with 25 different datasets. The data for each run is generated as follows. First, target labels (zero or one) are drawn independently for each data point from a Bernoulli distribution with parameter $p = 0.5$. Then, for each data point, we simulate a number of sensor values by drawing from Gaussian distributions. Sensors can be informative or uninformative: for informative sensors, the mean and variance of the generating Gaussians depends on the target label, whereas for uninformative sensors it does not. Finally, each sensor is used as an input to a weak decision-making agent, which is simulated by a Logistic Regressor [Bishop, 2006]. The Logistic Regressors are first trained on only 5 samples to produce uncertain predictors, then run on the test data. The output predictions form the input dataset for one run of the decision combination methods. The independent variable in an experiment is a parameter that alters the simulated sensors, the logistic regressors or the number of training labels supplied to the decision combination methods. Table 2.2 shows the parameters common to all experiments.

The combination methods tested are listed in Table 2.3 with references to their details in

Method Name	Described in	Assumptions
Majority Voting	Equation 2.9, Section 2.1	All agents equally reliable; discrete decisions.
Mean	Equation 2.8, Section 2.1	All agents equally reliable; agents output belief values; the mean is unbiased and calibrated
Weighted Majority	Equation 2.11, Section 2.2.1	Soft agent selection; discrete decisions; agent reliability is consistent across space of all relevant decisions; reliability constant over time
Weighted Sum	Equation 2.16, Section 2.2.1	Soft agent selection; agents output belief values; soft-selected agents are unbiased and calibrated; agent reliability is consistent across space of all relevant decisions; reliability constant over time
Naive Bayes LogOP	Equation 2.29, Section 2.2.2	Agents output beliefs, which may be biased and mis-calibrated; agent reliability is consistent across space of all relevant decisions; reliability constant over time
Dynamic Logistic Regressor (DLR)	[Penny and Roberts, 1999; Lowne et al., 2010]	Generic adaptive classifier; agent reliability is consistent across space of all relevant decisions
IBCC-Gibbs	Equation 2.35, Section 2.4	Discrete decisions; reliability constant over time

Table 2.3: Overview of decision combination methods tested.

previous sections of this chapter. The dedicated combination methods described in Chapter 2 are compared with a generic adaptive classifier, the *Dynamic Logistic Regressor (DLR)* [Penny and Roberts, 1999; Lowne et al., 2010]. In these experiments, the DLR takes the agents' outputs as its input data, in effect combining these inputs using a LogOP function (Equation 2.18) where weights are learnt in a discriminative manner, in contrast to the NB-LogOP. The Weighted Sum, Weighted Majority and DLR methods update weights sequentially as each data point is received. However, to enable a fair comparison with IBCC, which is run over the complete dataset, the combined decisions from Weighted Sum and Weighted Majority were produced using the final weights after processing all training and test data. The DLR is designed to adapt to changes in the relationship between its inputs and target labels, so does not converge to a final set of weights. Hence these

For IBCC-VB and IBCC-Gibbs	
For all agents, $\boldsymbol{\alpha}_{0,0} =$	$[12, 8]$
For all agents, $\boldsymbol{\alpha}_{0,1} =$	$[8, 12]$
$\boldsymbol{\nu}_0 =$	$[100, 100]$
For IBCC-Gibbs	
No. samples collected	5000
Sampling interval	5
No. burn-in samples	100
For Weighted Sum and Weighted Majority	
$\beta =$	0.5

Table 2.4: Hyperparameters and settings used in all simulated data experiments. Square brackets '[..]' represent vectors.

experiments compare very different learning methods.

The parameters to the decision combination methods are shown in table 2.4. These parameters were not fine-tuned to give the optimal performance in each test, rather they remain fixed for all experiments. They were chosen to give the best performance on the first dataset only so that we can observe how the performance changes as the difficulty of the decision combination problem increases. For IBCC, the $\boldsymbol{\alpha}_0$ hyperparameters represent a belief that the agents are more likely to be correct than incorrect.

2.5.1 Evaluation Method

For each experiment, we evaluate the performance of each combination method using the Receiver Operating Characteristic (ROC) [Fawcett, 2006]. The ROC plots a binary classifier's true positive rate (TPR) against false positive rate (FPR) for different values of a threshold that is used to convert probabilities to binary class values. Each point on the ROC curve corresponds to a different threshold value. Predictions above a given threshold are taken as positive classifications and those below as negative. The true positive rate at a particular threshold is the fraction of positive examples correctly identified by the classifier, while the false positive rate is the fraction of negative candidates incorrectly classified as positive. ROC curves provide a more informative analysis than simply calculating the

proportion of correct answers at a single threshold value of 0.5. They can naturally handle the case where a very small proportion of data points belong to one class, since FPR and TPR are calculated as fractions of the samples in each of two classes. They also allow the threshold to be optimised, which is valuable when false positives and false negatives have different costs e.g. in medical diagnosis.

In these experiments, ROC curves show the performance of decision combination methods at selected settings of the independent variable. The ROC curves are calculated by pooling the data from all repeats of the test at that particular independent variable setting. This indicates whether probability estimates have a consistent meaning over multiple runs of the algorithms, i.e. whether predictions produced by different runs can be used to rank objects in order of probability that the target label is 1. Pooling results to generate the ROC curve naturally penalises methods with a large *threshold variance* for a given pair of TPR/FPR values.

To provide easier comparison between methods, ROC curves are summarised by the area under the ROC curve (AUC). The AUC gives the probability that a randomly chosen positive instance is ranked higher than a randomly chosen negative instance. Thus the AUC provides an overall measure of classification efficacy. The results below show the mean of AUCs calculated separately for each run, summarising the expected performance of each method.

Most decision combination methods also estimate target label probabilities, which can be assessed using the *Brier Score* [Brier, 1950]. The Brier score is the mean squared difference between the forecast and the true outcome:

$$Brier = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{J}\left(\hat{p}(t_i = j) - \delta(t_i - j)\right)^2, \tag{2.55}$$

where δ is the delta function defined in equation 2.45. The Brier score is a *proper* score function, in that the loss is minimised when a classifier supplies their best estimates of the true distribution of data points given the information available [Gneiting and Raftery, 2007]. It cannot be decreased by making overly confident predictions, since any con-

fident mistakes are penalised more strongly than less certain predictions, and cannot be reduced by making overly cautious predictions, since these would result in an accumulation of smaller errors. Hence the Brier score includes an implicit measure of classifier calibration [Blattenberger and Lad, 1985]. An alternative function for evaluating probability estimates is the negative cross-entropy. However, an aim of these experiments is to measure the advantage of outputting probabilities over discrete labels, yet negative cross-entropy penalises discrete mistakes with infinite errors, which can be seen as too strong a loss function [Quinonero-Candela et al., 2006].

2.5.2 Experiment 1: Weak Agents

Sensor Error Rate	AUC	
	mean	S.D.
0	.846	.154
0.1	.782	.099
0.2	.708	.098
0.3	.621	.067
0.4	.567	.029
0.45	.537	.023
0.48	.513	.017
0.5	.500	.017

Table 2.5: Performance statistics for the 5 simulated agents.

Individual agents may be weak decision makers with high error rates, so the primary aim of decision combination is to achieve a lower error rate. This experiment therefore shows how much each decision combination method can boost agents as their error rate varies. The independent variable here is the sensor error rate of the input sensors to the agents. Table 2.5 shows the average error rates of individual agents for each sensor error rate setting. With no sensor error, the agents are still imperfect classifiers, since the sensor values are drawn from overlapping distributions for each target value, and agents are trained on only a small number of data points.

Consider the case where sensor error rate $= 0$ as a baseline scenario, into which we introduce decision combination problems, e.g. by increasing the sensor error rate in Ex-

Method Name	AUC		Brier score	
	mean	S.D.	mean	S.D.
Mean (Sum Rule)	.902	.089	.162	.059
Majority Voting	.783	.096	.329	.036
Weighted Majority	.867	.121	.335	.049
Weighted Sum	.985	.008	.298	.054
Naive Bayes LogOP	.978	.008	.326	.049
DLR	.967	.017	.279	.052
IBCC-Gibbs	.982	.010	.245	.066

Table 2.6: Performance statistics with sensor error rate $= 0$, as in the first test of Experiment 1. Means and standard deviations calculated over 25 repeats.

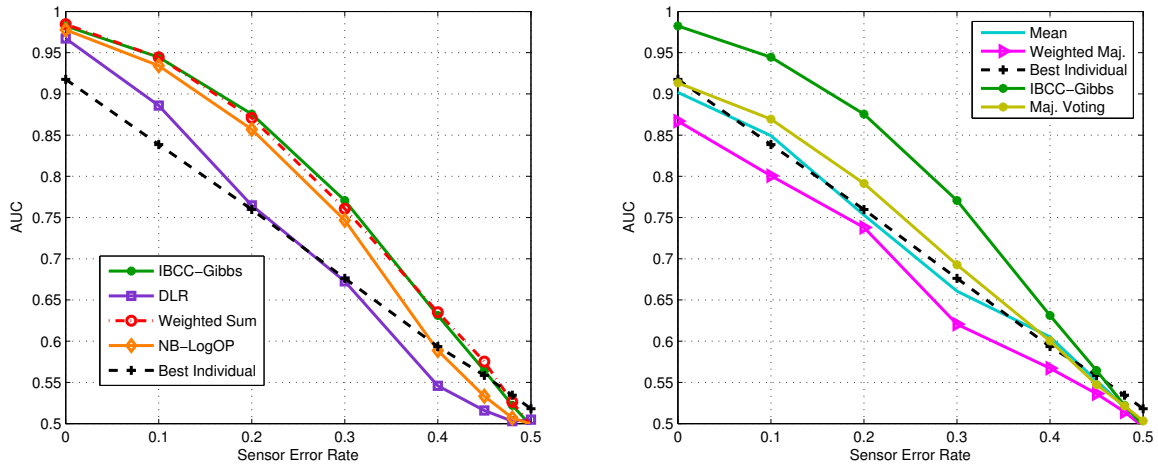


Figure 2.5: Experiment 1, mean AUCs over 25 repeats, varying sensor error rate.

periment 1. The AUC and Brier scores for this baseline are listed in Table 2.6. These show a clear advantage of using adaptive methods over mean and Majority Voting. IBCC-Gibbs produces comparable results with Weighted Sum, NB-LogOP and DLR, despite relying on discrete labels rather than the belief predictions supplied by agents.

Figure 2.5 visualises the mean AUC of each decision combination method as we increase the sensor error rate. For clarity, the figure separates methods into two panels; IBCC-Gibbs is plotted in both panels so it can be compared easily with all methods. The Weighted Sum and IBCC-Gibbs clearly outperform the other methods when the sensor error rate is between 0.1 and 0.45, while NB-LogOP performs comparably well for error rates up to 0.3. These methods significantly boost performance over that of the mean individual (see Table 2.5) and the best individual shown in Figure 2.5. Note that unlike

the Weighted Sum and NB-LogOP, IBCC-Gibbs is only provided with discrete decisions by the agents. Majority Voting improves over the best individual classifier using only the discrete decisions. Weighted Majority does not increase effectiveness compared to simple Majority Voting, and may be the result of overfitting, as the soft selection procedure tends toward selecting the single, most correct agent (see Section 2.2). In contrast, Weighted Sum may produce a softer selection of individuals in these experiments since it combines their probability estimates, resulting in less significant differences between the weight adjustments for different agents at each iteration. Agents that are mostly correct still produce soft decisions and are penalised proportionately, while incorrect agents are penalised less than they would be in a Weighted Majority.

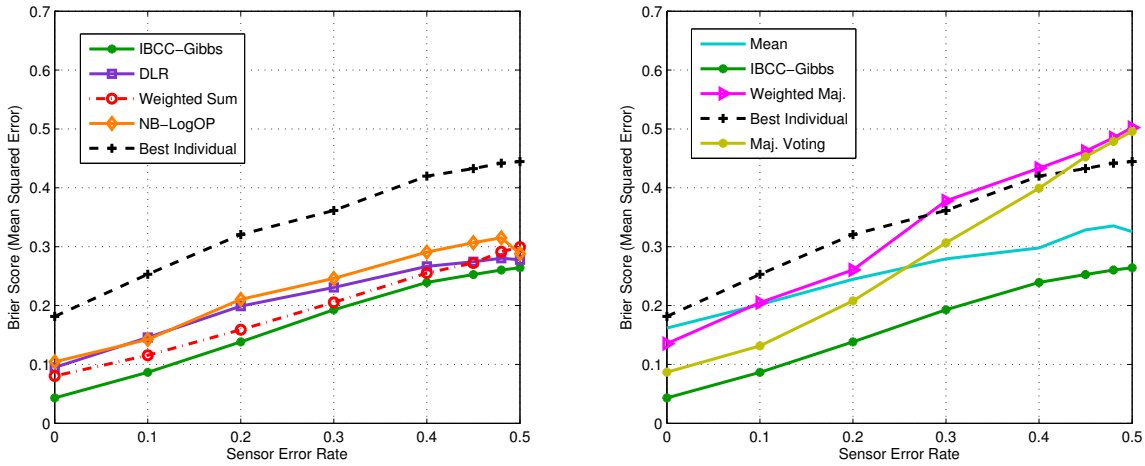
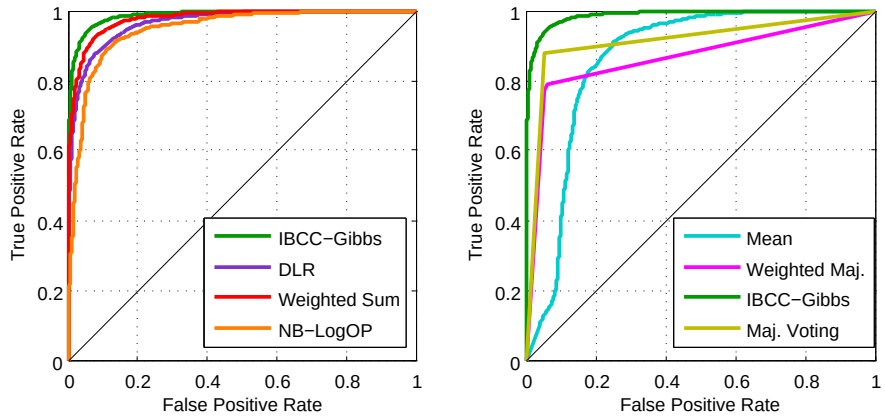


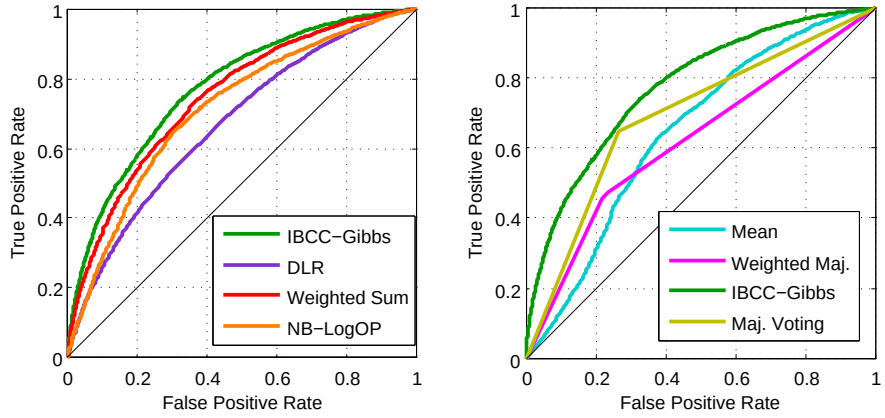
Figure 2.6: Experiment 1, mean Brier score over 25 repeats, comparing the reliability of predictions from each decision combination method with varying sensor error rate.

The Brier score is plotted against sensor error rate in Figure 2.6. IBCC-Gibbs produces a lower and therefore better Brier score at all error rates. The Brier scores of the DLR and mean compare more favourably with Weighted Sum and NB-LogOP than the AUCs, suggesting that they perform similarly at estimating probabilities, but are less effective at separating data points from each class.

Figure 2.7 shows the full ROC curves for sensor error rate $= 0$ and 0.3 respectively. Here it is possible to more clearly distinguish each of the methods described. Since Majority Voting and Weighted Majority emit only discrete combined decisions, it is only possible



(a) Sensor error rate $= 0$ (no sensor errors).



(b) Sensor error rate $= 0.3$.

Figure 2.7: Experiment 1, ROCs at selected sensor error rates.

to choose one threshold value, creating angular ROC plots. Despite having less information available, Majority Voting performs comparably to the Mean, producing better Brier scores. This suggests that agents' probability estimates may be unhelpful if, e.g. they are inaccurate due to insufficient training of agents.

2.5.3 Experiment 2: Ability Varies by Target Value

This scenario considers the case where the reliability of agents' decisions varies depending on the target value of the object they are classifying. For example, in a two class situation, an agent that is averse to assigning the label *positive* may often incorrectly assign *negative* to objects in the positive class, but always assign *negative* correctly to objects in the negative class. In this situation, the reliability of agents' decisions is not constant, since *negative* decisions cannot be trusted as they may be false negatives, whereas *positive* deci-

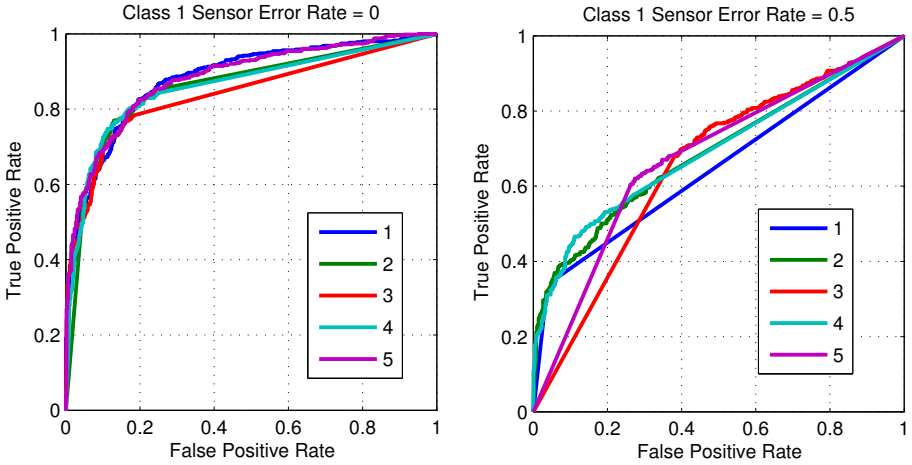


Figure 2.8: Experiment 2, ROC curves for base agents showing the effect of increasing sensor error rate for target value $t_i = 1$ only.

sions are reliable. In a multi-class situation, greater variation is also possible. This violates the assumptions of Weighted Majority, Weighted Sum, NB-LogOP and the DLR. The experiment tests this scenario by increasing the error rate of decisions about data points with target label $t_i = 1$ (the *class 1 error rate*). The effect that this has on the individual agents' error rates can be seen in Figure 2.8.

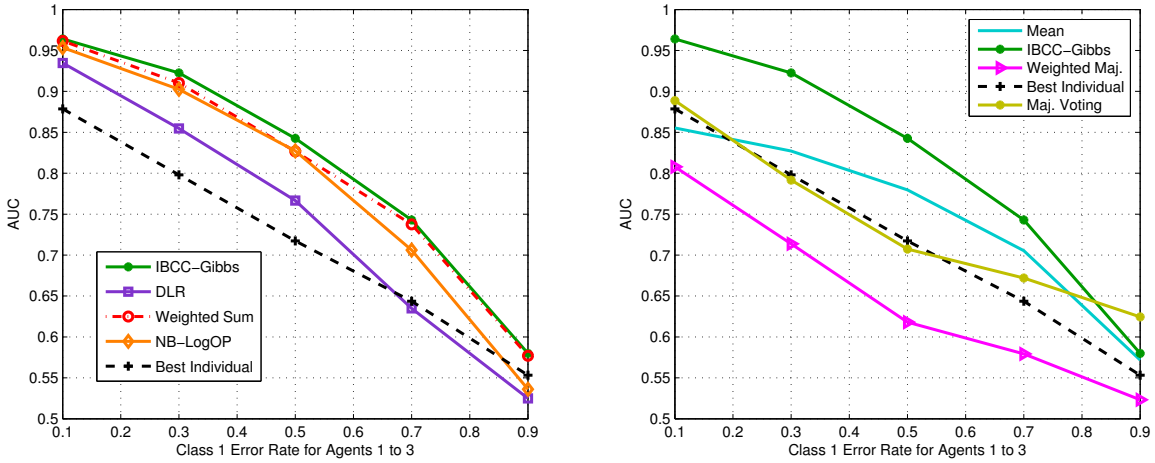


Figure 2.9: Experiment 2, mean AUCs over 25 repeats, varying sensor error rate for target value $t_i = 1$ only.

Figure 2.9 shows similar trends as in Experiment 1. However, Mean and Majority Voting perform comparatively well and appear less affected by the increased class 1 error than the best individual agent or the adaptive methods. In IBCC, the confusion matrices, $\boldsymbol{\Pi}$, explicitly model the variations in ability. However, IBCC nonetheless has only slightly

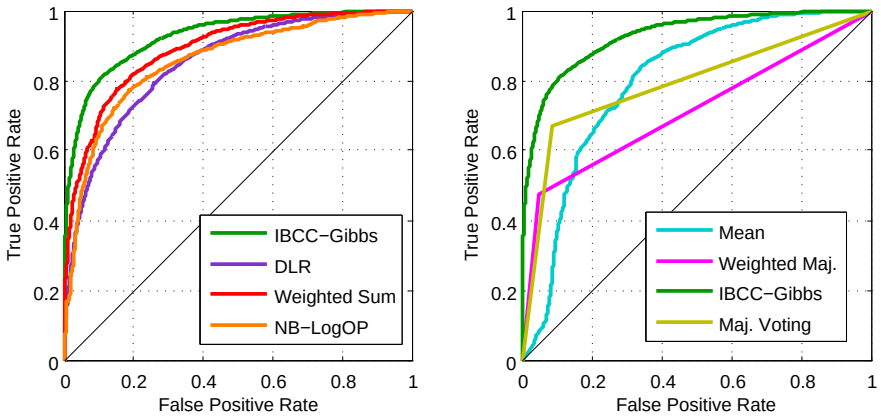


Figure 2.10: Experiment 2, receiver operating characteristic of each decision combination method with sensor error rate for class $1 = 0.3$.

better AUCs than Weighted Sum and NB-LogOP in this scenario. The improvement of IBCC is more clearly visible if we examine the ROC curves for the error rate 0.3, which are depicted in Figure 2.10. Figure 2.11 plots the Brier Scores, showing more clearly the advantage of IBCC over Weighted Sum and mean at high class 1 error rates. It is likely that the Weighted Sum and mean produce more inaccurate probability estimates in this experiment since they cannot model the different error rates for each target value. However, their AUCs are less affected by the different error rates because when presented with a class 1 object, there is usually at least one agent who will correctly predict $t_i = 1$ as the most likely target label, thus increasing the (weighted) mean for $p(t_i = 1)$.

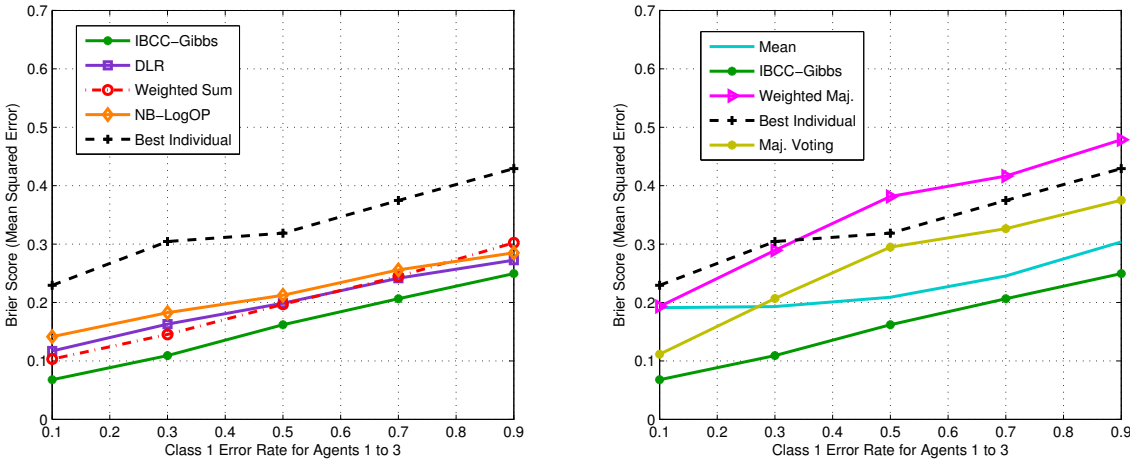


Figure 2.11: Experiment 2, mean Brier score over 25 repeats, comparing the reliability of target value predictions from each decision combination method with varying sensor error rate for target value $t_i = 1$ only.

2.5.4 Experiment 3: Noise

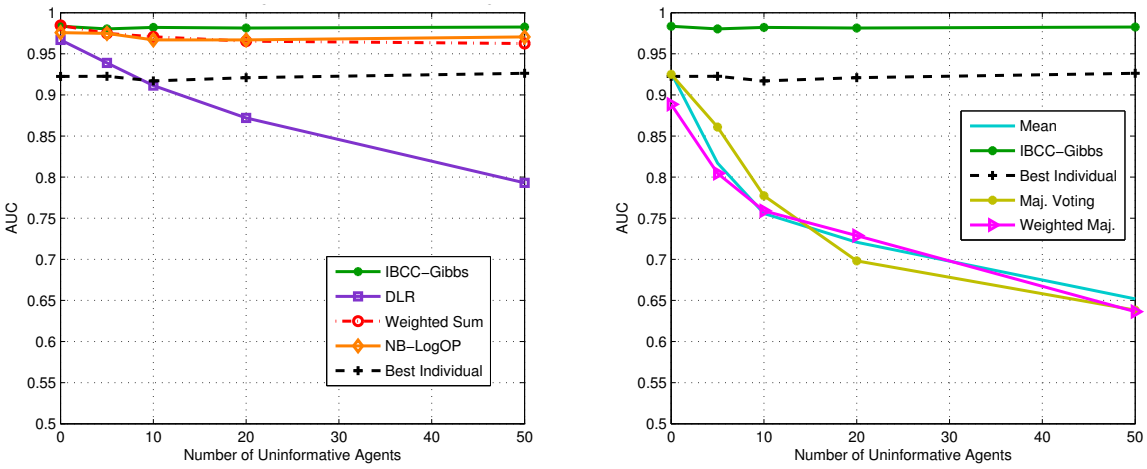


Figure 2.12: Experiment 3, mean AUCs over 25 repeats, varying number of uninformative agents (noise).

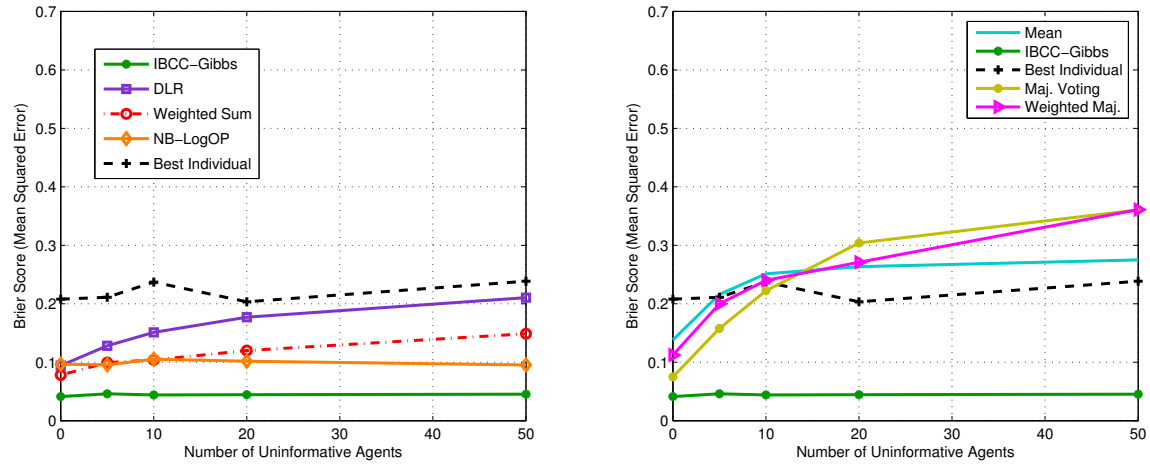


Figure 2.13: Experiment 3, mean Brier score over 25 repeats, comparing the reliability of target value predictions from each decision combination method with varying number of uninformative agents (noise).

Uninformative agents produce random responses that act as noisy input data to the decision combination methods. Such a situation may occur in a real-world scenario where human agents are not interested in the same decisions as the combiner, so their decisions do not relate to the same set of target values. Here, the scenario is simulated by introducing increasing numbers of uninformative agents that produce decisions entirely at random.

Figures 2.12 and 2.13 show that the methods most affected by noise are the DLR, mean, Majority Voting and Weighted Majority, while the others remain largely unaffected. The

Brier score for Weighted sum increases slightly, possibly because the uninformative agents still have non-zero weights and so continue to affect the decisions.

2.5.5 Experiment 4: Reversed Agents

Reversed agents are those that produce the opposite response to the target label for the majority of data points. This simulates antagonistic agents, sensor bit-flip errors, and misinterpretations between agents and combiners. The independent variable in Experiment 4 is the number of agents from the pool of 5 agents whose decisions are reversed.

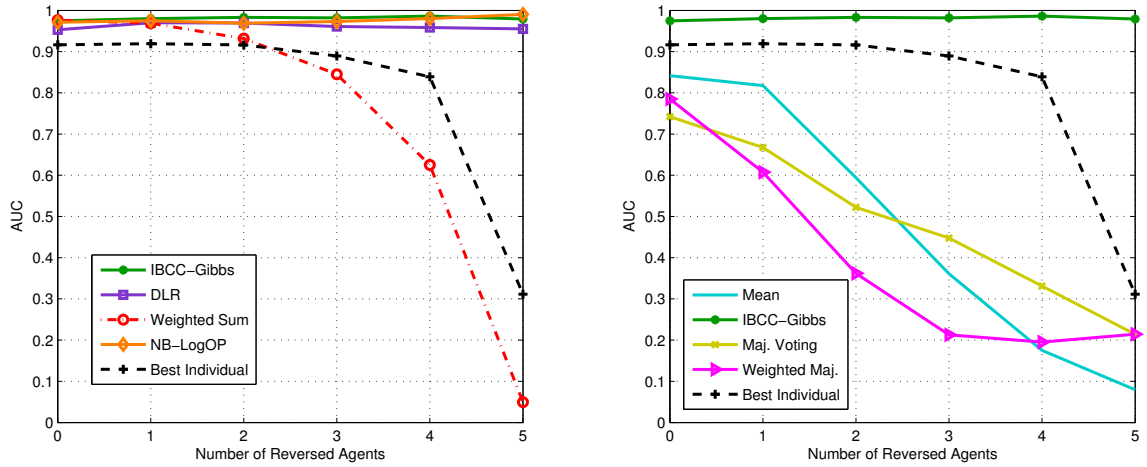


Figure 2.14: Experiment 4, mean AUCs over 25 repeats, varying number of reversed agents.

Figures 2.14 and 2.15 show that the performance of IBCC, DLR and NB-LogOP remains almost constant as agents' decisions are reversed, since these methods model responses probabilistically, rather than by pooling. The success of Weighted Sum, Weighted Majority, mean and Majority Voting deteriorates as more agents become reversed. The mean and Majority Voting start to produce reversed results themselves, which can be seen in the ROC curves below the diagonal in Figure 2.16. The Weighted Sum does not produced reversed results until all agents are reversed, which can be seen from the AUC, which is greater than 0.5 until all 5 agents become reversed. Weighted Sum can discount the reversed agents by down-weighting them, but in doing so it ignores the information they provide, leading to the poorer ROC curve in Figure 2.16. Weighted Majority might be ex-

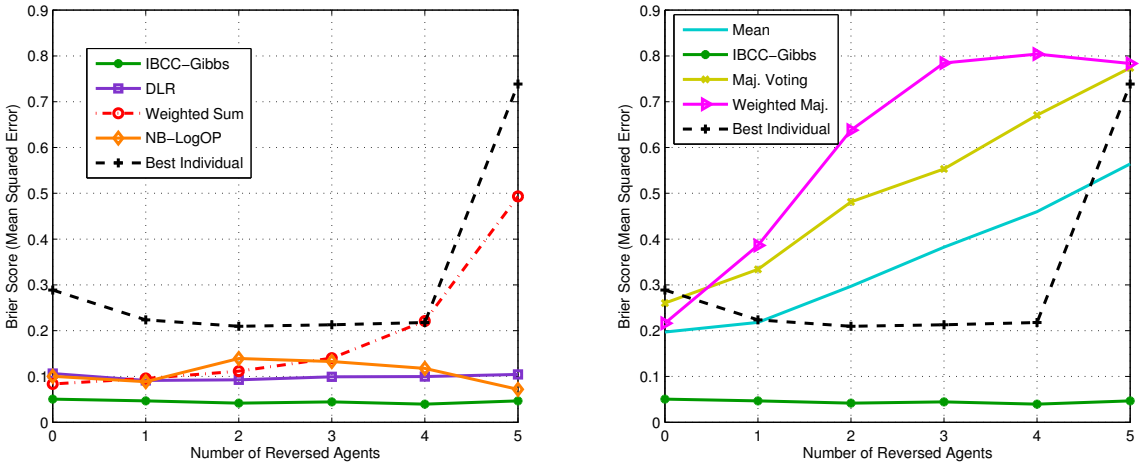


Figure 2.15: Experiment 4, mean Brier score over 25 repeats, comparing the reliability of target value predictions from each decision combination method with varying number of reversed agents.

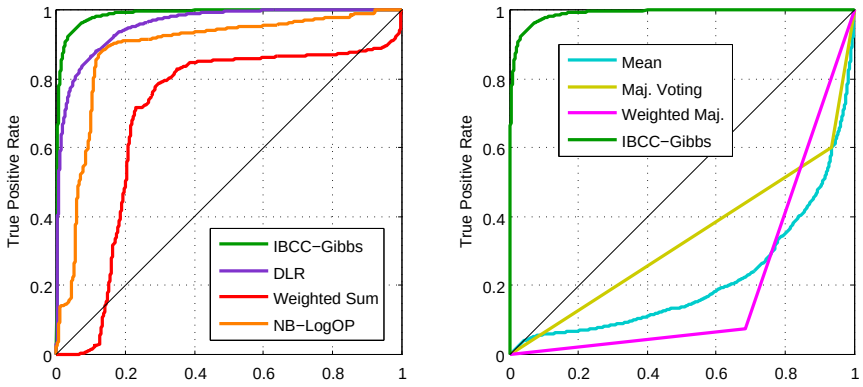


Figure 2.16: Experiment 4, receiver operating characteristic of each decision combination method with 4 reversed agents.

pected to perform similarly, but appears to down-weight some of the agents that are actually correct when more than one reversed agent was present. The ROC curve for NB-LogOP in Figure 2.16 is noticeably worse than the mean AUC in Figure 2.14. This discrepancy arises because the ROC curve was generated by combining data from all repeats with 4 reversed agents, whereas the mean AUC is calculated by averaging over AUCs calculated separately for each repeat. If the same points on the separate ROC curves correspond to different threshold values, the combined ROC curve will be worse than the separate curves. This suggests that the probability estimates of NB-LogOP are not compatible between different runs of the algorithm, as they have different bias or calibration.

2.5.6 Experiment 5: Correlated Agents

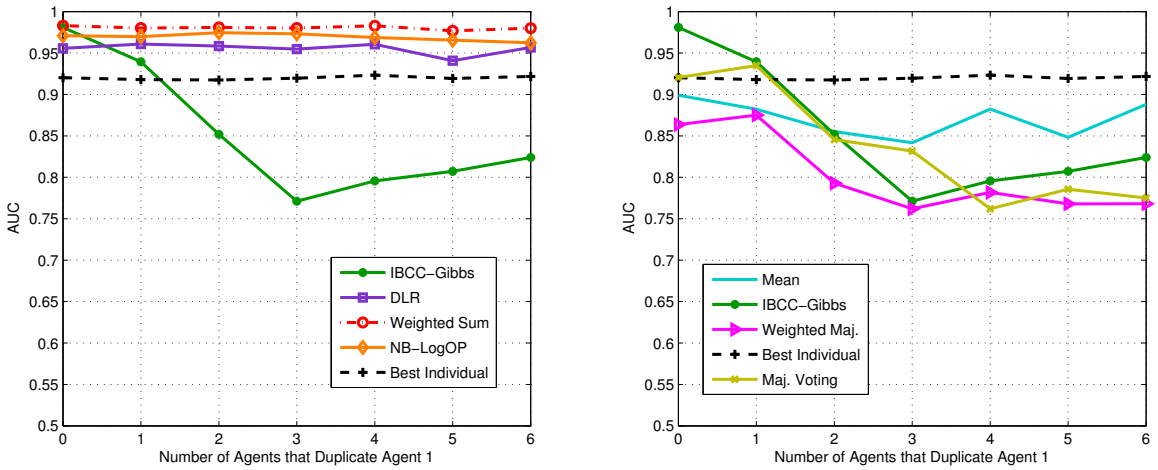


Figure 2.17: Experiment 5, mean AUCs over 25 repeats, varying number of duplicate agents.

Of the decision combination methods tested here, IBCC and NB-LogOP assume that agents' errors are uncorrelated. Experiment 5 tests the effect of this assumption by adding agents to the pool of 5 agents. The additional agents produce identical decisions to one of the previous agents. Figure 2.17 shows that the AUC of IBCC is significantly affected when muliple correlated agents are present. The Brier score is also increased, but remains below that of the best individual agent in figure 2.18. Weighted Majority and Majority Voting also decrease as the correlated agents gain dominance.

In Figure 2.19 we can more clearly see that the ROC curve for NB-LogOP is also poorer than for Weighted Sum and DLR, although it is not affected as much as IBCC-Gibbs. An interesting feature in Figure 2.19 is the shape of the ROC curve for IBCC-Gibbs, and to a lesser extent, for NB-LogOP. The ROC is more angular and more similar in shape to Majority Voting, which outputs a discrete combined decision. This shape would result from the combination methods producing more extreme probabilities when correlated agents are present.

The learning process in the two methods is very different, with NB-LogOP using only the labelled training points to learn its weights, whereas IBCC-Gibbs also exploits the latent structure in the unlabelled test points. IBCC-Gibbs is likely to infer more strongly

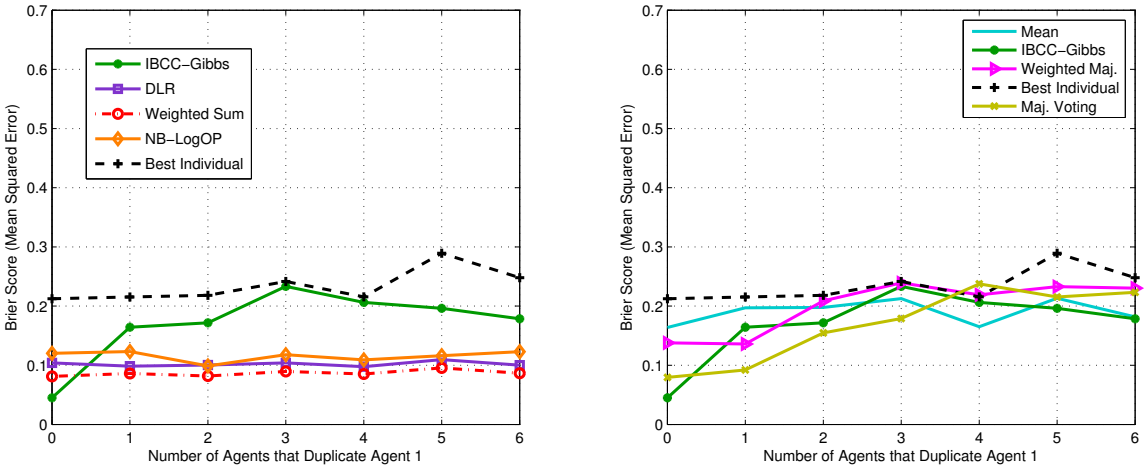


Figure 2.18: Experiment 5, mean Brier score over 25 repeats, comparing the reliability of target value predictions from each decision combination method with varying number of duplicate agents.

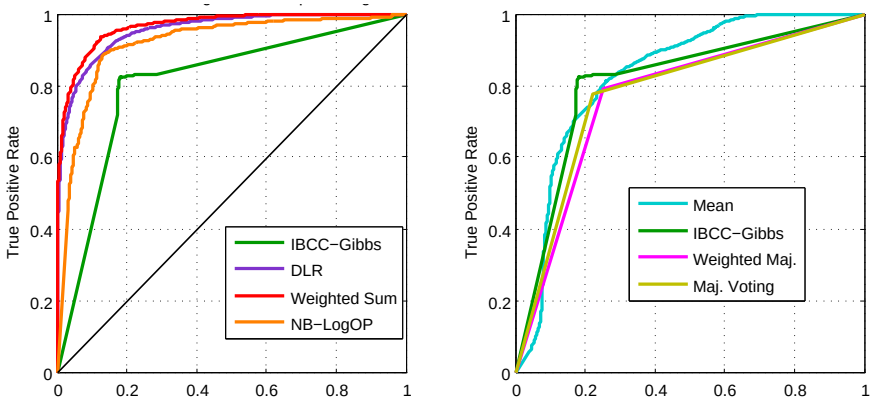


Figure 2.19: Experiment 5, receiver operating characteristic of each decision combination method with 6 duplicates of agent 1.

that the duplicated agents are highly reliable. The duplicated agents agree with each other, so are more often in agreement with the majority decision. Given that all agents have equal error rates, the majority decision is accepted for unlabelled data points. This leads to IBCC-Gibbs inferring that the duplicated agents are correct on far more of the unlabelled data points than is actually the case. In contrast, the other agents are inferred to be incorrect whenever they disagree with the majority. Thus the contribution of the duplicated agents is exaggerated, while that of the others is diminished. This dominance of the duplicated agents is reflected in mean AUCs close to 0.8, which is similar to that of the mean individual agent shown in Table 2.5. NB-LogOP is less affected by this as the reliabil-

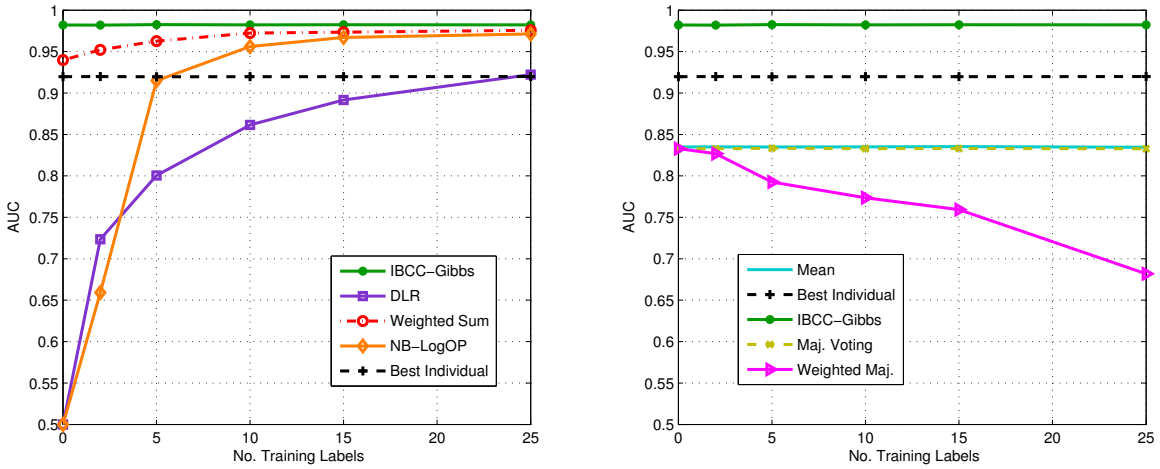


Figure 2.20: Experiment 6, mean AUCs over 25 repeats, varying number of training labels.

ity is not inferred from an agents' correlation with the majority. The Weighted Sum and Weighted Majority also avoid the problem because weights are only adjusted by decreasing the weights of agents in the majority when the majority is incorrect. Correlations between agents therefore do not cause an increase in their weights, but rather an increase the frequency with which the correlated agents' weights are decreased. Recalling that Weighted Sum and Weighted Majority perform soft selection over agents, when faced with two identical agents, after a number of iterations, the total weight apportioned to the pair of agents combined should equal that apportioned when only one of the agents was present.

2.5.7 Experiment 6: Training the Combiner

Training labels may be available to combiners to allow supervised learning of weights or distributions. IBCC-Gibbs, Weighted Majority and Weighted Sum can also operate in semi-supervised or unsupervised modes, where few or no training labels are provided. This experiment examines the way that each method responds with different amounts of training data. Four reversed agents are added to the five standard agents used in previous experiments, to test the ability of decision combination methods to cope with these differences with little training data.

According to the mean AUC in Figure 2.20 and Brier score in Figure 2.21, IBCC-Gibbs and Weighted Majority are not negatively affected when training labels are reduced, and

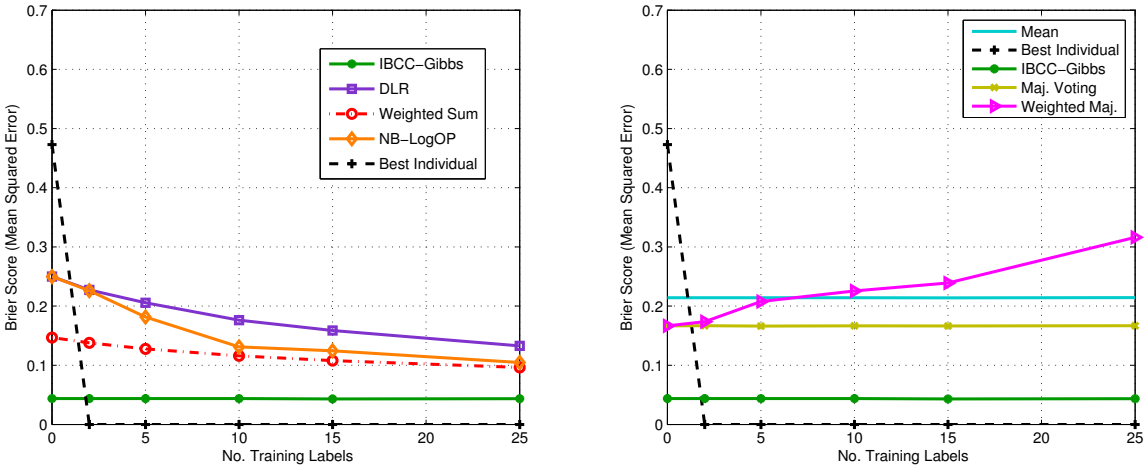


Figure 2.21: Experiment 6, mean Brier score over 25 repeats, comparing the reliability of target value predictions from each decision combination method with varying number of training labels.

Weighted Sum is only slightly affected. Perhaps surprisingly, Weighted Majority performs better with fewer labels, as it tends toward simple Majority Voting. As discussed with Experiment 1, Weighted Majority is a selection method, so the weights increase as training data is provided until one agent is selected. For example, the un-normalised weights of the agents in two sample test runs with 100 labels are:

Weighted Majority	Weights for Good Predictors					Weights for Reversed Agents			
First example	77	19	77	77	38	155	2	19	2
Second example	144	36	144	36	3	144	4	3	12

This shows significant emphasis on a small number of agents, sometimes even the reversed agents, which may be accidentally correct on a number of training points. In contrast, an example of the Weighted Sum shows weights that are more evenly distributed, with reversed agents correctly down-weighted:

Weighted Sum	Weights for Good Predictors					Weights for Reversed Agents			
Example	9	58	55	26	59	7	2	6	9

Since Weighted Sum calculates weights based on the probabilities they output rather than discrete decisions, we expect less focus on a single model.

2.5.8 Discussion of Experimental Results

These experiments illustrate the advantage of methods that learn the reliability of individual agents over fixed methods such as Mean and Majority Voting. However, the Weighted Majority mostly failed to improve over other methods. This result confirms that models that perform soft selection, such as Weighted Majority, are not ideal for decision combination, although the Weighted Sum can perform well if the weights do not collapse to focus on a single agent.

In most experiments, IBCC-Gibbs produced the best results, with AUCs above that of the nearest contenders, NB-LogOP and Weighted Sum. Also note that in most cases, Majority Voting produces better Brier scores than the Mean and comparable AUCs, yet requires only discrete decisions from agents. Therefore probability estimates from agents do not appear necessary for good performance, and the combiner's model and learning algorithm appear to influence the results more significantly. The DLR seems not to produce such good results in this setting, either due to an unsuitable model or because it learns in a sequential manner. The Brier scores also show a notable improvement when using IBCC-Gibbs over all other methods, suggesting that this algorithm may provide more accurate probability estimates. IBCC is also robust to noise and reversed agents. Theoretically, the confusion matrices, $\boldsymbol{\Pi}$, allow IBCC to better model agents whose error rates vary between classes, as in Experiment 2. IBCC demonstrated a small performance gain relative to the other methods in Experiment 2 when compared to Experiment 1. However, the difference may be more apparent in multi-class problems where a single weight or accuracy estimate is less expressive. The main weakness encountered in IBCC is in Experiment 5, where correlated agents reduce the AUC from 0.97 to 0.82. In cases where correlation reduces accuracy, a pre-processing step could be used to detect and remove highly-correlated agents from the combination entirely.

The methods tested here all use different inference algorithms as well different underlying models. Using Gibbs sampling to learn a posterior distribution, as with IBCC-Gibbs, can produce good results even when training labels are unavailable, as shown in Experiment 6. It may be interesting to explore whether a Bayesian treatment of NB-LogOP, Weighted Majority or Weighted Sum would improve their efficacy, particularly in unsupervised settings.

The scenarios tested here picked out a selection of individual difficulties when combining decisions. Many other scenarios and complex combinations of these challenges are possible, which would require further testing should they arise in a real application. However, these experiments showed that IBCC is best able to handle most of these problems, offering a substantial performance advantage while requiring only discrete decisions from agents.

2.6 Conclusions

This chapter introduced the field of decision combination, first describing some of the factors that influence the design of decision combination methods. The next section reviewed established approaches to decision combination, covering fixed combination functions, supervised learning methods, unsupervised methods. A principled, Bayesian approach to decision combination, Independent Bayesian Classifier Combination (IBCC), was then presented in detail. The experiments in this chapter using simulated data demonstrated the strengths of IBCC compared to established alternatives. Aside from the strong empirical performance of IBCC-Gibbs, this approach has a number of advantages. Using Bayesian inference gives flexibility, since different priors can be set for each agent, e.g. when we know *a priori* that both experts and non-experts are present. Within IBCC, the confusion matrices, $\boldsymbol{\Pi}$, provide a more detailed model of agent behaviour than the single weights or accuracy measures given by other models. Later chapters will explore how the confusion matrices facilitate optimal agent selection, task assignment and training. The next two chapters address some of the shortcomings of IBCC-Gibbs. Firstly, Gibbs sampling is

computationally expensive, which presents problems in real-world applications. Chapter 3 therefore proposes a more efficient inference algorithm and applies IBCC to real-world problems. Secondly, behaviour of agents is not necessarily constant over time, e.g. due to learning or physical agents moving. Chapter 4 therefore develops IBCC into a dynamic Bayesian model for decision combination.

Chapter 3

Efficient Application of Bayesian Classifier Combination

This chapter develops Independent Bayesian Classifier Combination for practical application to a large real-world citizen science project, Galaxy Zoo Supernovae (GZSN). First, we describe GZSN as a motivating example for work throughout the thesis, explaining how Independent Bayesian Classifier Combination (IBCC) could present an opportunity to improve the performance of the system. Citizen science applications typically require decision combination methods to be run very frequently over large datasets, so a computationally efficient approach is required. The second section of this chapter therefore proposes a variational inference algorithm for IBCC, *VB-IBCC* to provide rapid execution. The following sections evaluate *VB-IBCC* empirically using both synthetic data, Galaxy Zoo Supernovae data and two further test applications. The final section demonstrates how the IBCC model can be used to further analyse the behaviour of agents through community detection. By examining groups of similar agents, we observe a number of diverse behavioural groups in this application, which may in future be targeted with particular tasks and training. We also use community detection to examine whether the specific tasks completed by users affect our model of their behaviour.

3.1 Application: Galaxy Zoo Supernovae

Galaxy Zoo Supernovae (GZSN) is a citizen science project within the umbrella group *Zooniverse*[1], in which human agents carry out classification tasks. The aim of GZSN is to classify candidate objects as either a supernova or not a supernova using telescope images [Smith et al., 2010]. Untrained volunteers act as citizen scientists who attempt to decide on the correct classifications. Volunteers can log into the system at any time they choose and are then presented with images of a candidate object, similar to the example in Figure 3.1. The user then answers a series of questions according to a decision tree, or

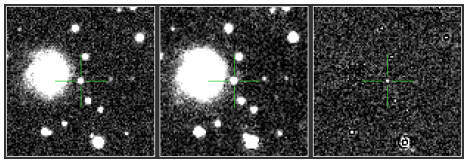


Figure 3.1: Example of a set of images presented to volunteers in Galaxy Zoo Supernovae (GZSN).

may skip the object. Questions asked of the users include "is there a candidate centred in the crosshairs of the right-hand image?" and "is the candidate centred in a circular host galaxy?". Each set of answers given by a single volunteer corresponds to a score of either -1 (not supernova), 1 (uncertain) or 3 (supernova). This score characterises the agent's response to an object. The abilities of agents can vary greatly, with no guarantee over any individual's performance, as each user can have radically different levels of domain expertise. The reliability of volunteer agents is unknown to GZSN, and there is often little performance data available for individual users, since many users complete only small numbers of tasks per session. To mitigate for the mistakes made by individual participants, GZSN passes each object to multiple volunteers at random. The total of their scores is used to assign an estimated decision. Objects with total scores below 0 are assumed to be negative, while those above 1 are assumed positive. Those that have not crossed either threshold are assigned to more decision makers until they are either classified as negative or positive or have been classified more than 20 times and are finally marked as unknown. The

[1] http://www.zooniverse.org

project aims to identify genuine astronomical entities and events, so positive classifications lead to domain experts reviewing the images and requesting more expensive analysis of the objects. Thus, there is a need to maximise the accuracy of combined classifications from GZSN. This presents an opportunity to deploy a more principled decision combination method that can learn and account for the varying abilities of agents.

To process large datasets, GZSN reaches out to thousands of volunteer citizen scientists, each acting as a decision-making agent. Their collective performance is comparable to that of experienced human scanners performing similar tasks as part of the related project, Palomar Transient Factory (PTF) [Smith et al., 2010]. The scale is large: another Zooniverse project, Galaxy Zoo, received over 60,000 classifications per hour after running for forty hours [Fortson et al., 2011], and any decision combination method that informs task allocation may need to be re-run frequently as new decisions are received. Thus one challenge for a decision combination method – especially where regular updates are needed – is to operate quickly at scale. The quantity of data that must be processed also motivates further automation by introducing machine learning algorithms that can learn from the human classifiers.

Aside from the need for accurate classifications, there is also a strong desire to use the volunteers' time efficiently, both for their own satisfaction and the performance of GZSN. This motivates the need for efficient agent/task allocation methods explored in Chapter 5, which necessitates learning a model of the individual agents' abilities.

Independent Bayesian Classifier Combination (IBCC) can be used to address many of the challenges described above. Firstly, it can combine volunteers' decisions to a produce a classification with a probabilistic measure of certainty, and does not require large training sets. Secondly, through the confusion matrices for each agent, $\pi^{(k)}$, IBCC is able to model the behaviour of individuals. However, a major disadvantage of the existing Gibbs' sampling implementation is computational cost. It would be desirable to be able to run the algorithm repeatedly as new classifications arrive, and over far larger datasets than have been tested in Chapter 2. For example, we may use data from agents who have worked on

multiple projects within Zooniverse, meaning we must deal with hundreds of thousands of agents. The next section addresses this problem by proposing an alternative, more scalable inference algorithm. The following section then presents an empirical comparison with both the Gibbs' sampling algorithm and other combination methods, first on our synthetic dataset from Chapter 2, then on datasets taken from GZSN.

3.2 Variational Bayesian IBCC

As described in Chapter 2, the goal of IBCC is to perform inference for the unknown variables $\boldsymbol{t}$, $\boldsymbol{\Pi}$, and $\boldsymbol{\kappa}$, where $\boldsymbol{t}$ is the vector of target labels for the objects we wish to classify, $\boldsymbol{\Pi}$ is the set of confusion matrices characterising the agents' behaviour, and $\boldsymbol{\kappa}$ is a vector listing the proportions of each target value. To predict a target label t_i for a test dataset in a Bayesian manner, we wish to approximate the expectation $\mathbb{E}[t_i = j] = p(t_i = j|\boldsymbol{c}, \boldsymbol{\nu}_0, \boldsymbol{A}_0)$, where $\boldsymbol{c}$ is the set of agent responses, and $\boldsymbol{\nu}_0$ and $\boldsymbol{A}_0$ are the hyperparameters. The exact posterior distribution cannot be obtained analytically, so must be estimated using sampling techniques or approximated by an analytical distribution. This section first discusses the options for inference, then proposes and derives a principled, approximate approach using variational Bayes. The resulting algorithm is described using pseudo-code in Appendix B.

Chapter 2 investigated Gibbs' sampling for IBCC, as suggested by [Ghahramani and Kim, 2003], showing promising performance on synthetic data. Assuming a specific model, Gibbs' sampling [Geman and Geman, 1984] is a Markov-Chain Monte Carlo method that estimates the exact posterior distributions over the variables. As the number of samples S grows, $S \to \infty$, the numerical estimate becomes more accurate. Despite this theoretical guarantee, Gibbs' sampling suffers from slow convergence, and it can be difficult to ascertain when the set of samples obtained is a good representation of the posterior distribution [Neal, 1993]. The sampling algorithm should explore the complete distribution, potentially including local maxima, so may require a large number of samples to provide confidence in the estimate obtained.

As a computationally inexpensive alternative to estimating the posterior distribu-

tions over the variables, we can also compute point estimates using a method such as *Expectation-Maximisation* (EM) algorithm [Dempster et al., 1977]. Such methods can be used to find the maximum likelihood estimate, which does not consider prior distributions, or the maximum a posteriori (MAP) estimate. These estimates are modes of the variables rather than a distribution over their values. MAP estimation is deployed in a model similar to IBCC by [Raykar et al., 2010].

An efficient alternative to sampling methods and point estimates is *variational Bayes* (VB) [Jordan et al., 1999; Attias, 2000]. This principled Bayesian method uses an analytical approximation to the posterior distribution, allowing us to replace non-analytic marginal integrals in the original model with analytic updates in the *sufficient statistics* of the variational approximation. A sufficient statistic for a parameter θ is a statistic that summarises the distribution of a data sample $\boldsymbol{X}$ in such a way no further information can be obtained about the value of θ from the original data $\boldsymbol{X}$. These updates are performed in an iterative manner, which can be seen as a fully Bayesian generalisation of EM. The remainder of this section explains the workings of VB before presenting the IBCC-VB derivation.

3.2.1 Variational Bayes

Given a set of observed data $\boldsymbol{X}$ and a set of latent variables and parameters $\boldsymbol{Z}$, the goal of variational Bayes (VB) is to find a tractable approximation $q(\boldsymbol{Z})$ to the posterior distribution $p(\boldsymbol{Z}|\boldsymbol{X})$ by minimising the KL-divergence $KL(q||p)$ [Kullback and Leibler, 1951] between the approximate distribution and the true distribution [Attias, 2000; Fox and Roberts, 2011]. We can write the log of the marginal likelihood $p(\boldsymbol{X})$ as

$$\begin{aligned}\ln p(\boldsymbol{X}) &= \int q(\boldsymbol{Z}) \ln \frac{p(\boldsymbol{X}, \boldsymbol{Z})}{q(\boldsymbol{Z})} \mathrm{d}\boldsymbol{Z} - \int q(\boldsymbol{Z}) \ln \frac{p(\boldsymbol{Z}|\boldsymbol{X})}{q(\boldsymbol{Z})} \mathrm{d}\boldsymbol{Z} \\ &= \mathcal{L}(q) - \mathrm{KL}(q||p), \end{aligned} \tag{3.1}$$

where $\mathcal{L}(q)$ is a lower bound on $\ln p(\boldsymbol{X})$. As $q(\boldsymbol{Z})$ approaches $p(\boldsymbol{Z}|\boldsymbol{X})$, the KL-divergence disappears and the lower bound $\mathcal{L}(q)$ on $\ln p(\boldsymbol{X})$ is maximised. Variational Bayes selects a restricted form of $q(\boldsymbol{Z})$ that is tractable to work with, then seeks the distribution within this restricted form that minimises the KL-divergence. A common restriction is to partition $\boldsymbol{Z}$ into groups of variables, then assume $q(\boldsymbol{Z})$ factorises into functions of single groups:

$$q(\boldsymbol{Z}) = \prod_{i=1}^{M} q_i(\boldsymbol{Z}_i). \tag{3.2}$$

For each factor $q_i(\boldsymbol{Z}_i)$ we then seek the optimal solution $q_i^*(\boldsymbol{Z}_i)$ that minimises the KL-divergence. Consider partitions of variables $\boldsymbol{Z}_i$ and $\bar{\boldsymbol{Z}}_i$, where $\bar{\boldsymbol{Z}}_i = \{\boldsymbol{Z}_j | j \neq i, j = 1...M\}$. *Mean field theory* [Parisi, 1988] shows that we can find an optimal factor $q_i^*(\boldsymbol{Z}_i)$ from the conditional distribution $p(\boldsymbol{Z}_i|\boldsymbol{X}, \bar{\boldsymbol{Z}})$ by taking the expectation over all the other factors $j|j \neq i, j = 1...M$. We can therefore write the log of the optimal factor $\ln q_i^*(\boldsymbol{Z}_i)$ as the expectation with respect to all other factors of the log joint distribution over all variables plus a normalisation constant:

$$\ln q_i^*(\boldsymbol{Z}_i) = \mathbb{E}_{q(\bar{\boldsymbol{Z}})}[\ln p(\boldsymbol{X}, \boldsymbol{Z})] + \text{const.} \tag{3.3}$$

In our notation, we take the expectation with respect to the variables in the subscript. In this case, $\mathbb{E}_{q(\bar{\boldsymbol{Z}})}[...]$ indicates that we take an expectation with respect to all factors except $q(\boldsymbol{Z}_i)$. This expectation is implicitly conditioned on the observed data, $\boldsymbol{X}$, which we omit from the notation for brevity.

We can evaluate these optimal factors iteratively by first initialising all factors, then updating each in turn using the expectations with respect to the current values of the other factors. Unlike Gibbs sampling, each iteration is guaranteed to increase the lower bound on the log-likelihood, $\mathcal{L}(q)$, converging to a local maximum in a similar fashion to standard EM algorithms. In practice, once the optimal factors $q_i^*(\boldsymbol{Z}_i)$ have converged to within a given tolerance, we can approximate the distribution of the unknown variables and calculate their expected values.

The choice of initial values can still be important with VB, since it may alter the number of iterations required to converge. Initialising the factors to values close to the converged values means fewer changes will be required. This is particularly significant when we wish to add a small amount of new data, having already run the algorithm, since the values we can initialise the factors to the values inferred previously, assuming the new data will cause only small alterations to these values. We would then expect to need far fewer iterations to update the model given the new data points than were required to run the algorithm the first time. This contrasts with Gibbs' sampling, where samples collected before more data was added become invalid.

A further technique for increasing computational efficiency is to initialise the variational factors using a very fast but less accurate approximation, such the Expectation Maximisation (EM) [Dempster et al., 1977] algorithm discussed earlier. This works by the same principle, assuming that the rough algorithm will find values that are reasonably close to the optimal values, so VB will need fewer iterations to refine those values.

3.2.2 Variational Equations for IBCC

This subsection presents a variational approximation for IBCC and derives the equations for calculating the updates to the sufficient statistics of the parameters. The following subsection then explains the iterative update algorithm that uses these equations, which is also detailed in pseudo-code in Appendix B.

To provide a variational Bayesian treatment of IBCC, IBCC-VB, we first propose the form for our variational distribution, $q(\boldsymbol{Z})$, that factorises between the model parameters and latent variables. In this case, the model parameters are $\boldsymbol{\Pi}$ and $\boldsymbol{\kappa}$, and the latent variables are the target labels of $\boldsymbol{t}$, so we can write the following variational distribution:

$$q(\boldsymbol{\kappa}, \boldsymbol{t}, \boldsymbol{\Pi}) = q(\boldsymbol{t})q(\boldsymbol{\kappa}, \boldsymbol{\Pi}) \tag{3.4}$$

This is the only assumption we must make to perform VB on this model; the forms of the

factors arise from our model of IBCC. We can use the joint distribution in Equation (2.35) to find the optimal factors $q^*(\boldsymbol{t})$ and $q^*(\boldsymbol{\kappa}, \boldsymbol{\Pi})$ in the form given by Equation (3.3). For the target labels we have

$$\ln q^*(\boldsymbol{t}) = \mathbb{E}_{\boldsymbol{\kappa},\boldsymbol{\Pi}}[\ln p(\boldsymbol{\kappa}, \boldsymbol{t}, \boldsymbol{\Pi}, \boldsymbol{c})] + \text{const.} \tag{3.5}$$

We can rewrite this into factors corresponding to independent data points, with any terms not involving t_i being absorbed into the normalisation constant. To do this we define ρ_{ij} as

$$\begin{aligned} \ln \rho_{ij} &= \mathbb{E}_{\kappa_j,\boldsymbol{\Pi}}[\ln p(t_i, \boldsymbol{c}|\boldsymbol{\Pi}, \kappa_j)] \\ &= \mathbb{E}_{\boldsymbol{\kappa}}[\ln \kappa_j] + \sum_{k=1}^{K} \mathbb{E}_{\boldsymbol{\Pi}}[\ln \pi^{(k)}_{j,c_i^{(k)}}] \end{aligned} \tag{3.6}$$

then we can estimate the probability of a true label, which also gives its expected value:

$$q^*(t_i = j) = \mathbb{E}_{\boldsymbol{t}}[t_i = j] = \frac{\rho_{ij}}{\sum_{\iota=1}^{J} \rho_{i\iota}} \ . \tag{3.7}$$

To simplify the optimal factors in subsequent equations, we define expectations with respect to $\boldsymbol{t}$ of the number of occurrences of each true class, given by

$$N_j = \sum_{i=1}^{N} \mathbb{E}_{\boldsymbol{t}}[t_i = j] \ , \tag{3.8}$$

and the counts of each classifier decision $c_i^{(k)} = l$ given the true label $t_i = j$, by

$$N_{j,l}^{(k)} = \sum_{i=1}^{N} \delta(c_i^{(k)} - l)\mathbb{E}_{\boldsymbol{t}}[t_i = j] \tag{3.9}$$

where $\delta(c_i^{(k)} - l)$ is unity if $c_i^{(k)} = l$ and zero otherwise.

For the parameters of the model we have the optimal factors given by:

$$\begin{aligned}\ln q^*(\boldsymbol{\kappa}, \boldsymbol{\Pi}) &= \mathbb{E}_{\boldsymbol{t}}[\ln p(\boldsymbol{\kappa}, \boldsymbol{t}, \boldsymbol{\Pi}, \boldsymbol{c})] + \text{const} \\ &= \mathbb{E}_{\boldsymbol{t}}\left[\sum_{i=1}^{N} \ln \kappa_{t_i} + \sum_{i=1}^{N}\sum_{k=1}^{K} \ln \pi^{(k)}_{t_i, c_i^{(k)}}\right] + \ln p(\boldsymbol{\kappa}|\boldsymbol{\nu}_0) + \ln p(\boldsymbol{\Pi}|\boldsymbol{A}_0) + \text{const}.\end{aligned} \tag{3.10}$$

In Equation (3.10) terms involving $\boldsymbol{\kappa}$ and terms involving each confusion matrix in $\boldsymbol{\Pi}$ are separate, so we can factorise $q^*(\boldsymbol{\kappa}, \boldsymbol{\Pi})$ further into

$$q^*(\boldsymbol{\kappa},\boldsymbol{\Pi}) = q^*(\boldsymbol{\kappa}) \prod_{k=1}^{K}\prod_{j=1}^{J} q^*\left(\boldsymbol{\pi}_j^{(k)}\right). \tag{3.11}$$

In the IBCC model (Section 2.4.1) we assumed a Dirichlet prior for $\boldsymbol{\kappa}$, which gives us the optimal factor

$$\begin{aligned}\ln q^*(\boldsymbol{\kappa}) &= \mathbb{E}_{\boldsymbol{t}}\left[\sum_{i=1}^{N} \ln \kappa_{t_i}\right] + \ln p(\boldsymbol{\kappa}|\boldsymbol{\nu}_0) + \text{const} \\ &= \sum_{j=1}^{J} N_j \ln \kappa_j + \sum_{j=1}^{J} (\nu_{0,j} - 1) \ln \kappa_j + \text{const}.\end{aligned} \tag{3.12}$$

Taking the exponential of both sides, we obtain a posterior Dirichlet distribution of the form

$$q^*(\boldsymbol{\kappa}) \propto \text{Dir}(\boldsymbol{\kappa}|\nu_1, ..., \nu_J) \tag{3.13}$$

where $\boldsymbol{\nu}$ is updated in the standard manner by adding the data counts to the prior counts ν_0:

$$\nu_j = \nu_{0,j} + N_j. \tag{3.14}$$

The expectation of $\ln \boldsymbol{\kappa}$ required to update Equation (3.6) is therefore:

$$\mathbb{E}_{\boldsymbol{\kappa}}\left[\ln \kappa_j\right] = \Psi(\nu_j) - \Psi\left(\sum_{\iota=1}^{J} \nu_\iota\right) \tag{3.15}$$

where Ψ is the standard digamma function [Davis, 1965].

Each row of each confusion matrix $\boldsymbol{\Pi}$ is independent, so the variational distribution $q^*(\boldsymbol{\Pi})$ factorises further:

$$q^*(\boldsymbol{\Pi}) = \prod_{k=1}^{K} \prod_{j=1}^{J} q^*(\boldsymbol{\pi}_j^{(k)}) \tag{3.16}$$

For a row of a confusion matrix, $\boldsymbol{\pi}_j^{(k)}$, the priors are Dirichlet distributions giving the factor

$$\begin{aligned} \ln q^* \left(\boldsymbol{\pi}_j^{(k)}\right) &= \sum_{i=1}^{N} \mathbb{E}_{t_i}[t_i = j] \ln \pi^{(k)}_{j,c_i^{(k)}} + \ln p \left(\boldsymbol{\pi}_j^{(k)} | \boldsymbol{\alpha}_{0,j}^{(k)}\right) + \text{const} \\ &= \sum_{l=1}^{L} N_{jl}^{(k)} \ln \pi_{jl}^{(k)} + \sum_{l=1}^{L} \left(\alpha_{0,jl}^{(k)} - 1\right) \ln \pi_{jl}^{(k)} + \text{const}. \end{aligned} \tag{3.17}$$

Again, taking the exponential gives a posterior Dirichlet distribution of the form

$$q^* \left(\boldsymbol{\pi}_j^{(k)}\right) = \text{Dir} \left(\boldsymbol{\pi}_j^{(k)} | \alpha_{j1}^{(k)}, ..., \alpha_{jL}^{(k)}\right) \tag{3.18}$$

where $\boldsymbol{\alpha}_j^{(k)}$ is updated by adding data counts to prior counts $\alpha_{0,j}^{(k)}$:

$$\alpha_{jl}^{(k)} = \alpha_{0,jl}^{(k)} + N_{jl}^{(k)}. \tag{3.19}$$

The expectation required for Equation (3.6) is given by

$$\mathbb{E}_{\boldsymbol{\Pi}} \left[\ln \pi_{jl}^{(k)}\right] = \Psi \left(\alpha_{jl}^{(k)}\right) - \Psi \left(\sum_{m=1}^{L} \alpha_{jm}^{(k)}\right). \tag{3.20}$$

The equations derived above are used by the algorithm described in the next subsection 3.2.3 to optimise the approximate solution to the posterior distribution.

3.2.3 The IBCC-VB Algorithm

The basic pattern of the VB algorithm is to iterate between updating the latent variables, in this case the target labels, $\boldsymbol{t}$, and the model parameters, which for IBCC are the confusion matrices $\boldsymbol{\Pi}$ and the target value proportions $\boldsymbol{\kappa}$. The algorithm takes as *input data* a set of agents' responses, $\boldsymbol{c}$, and where available, a set of known target labels, $\boldsymbol{t}_{known}$, i.e. training

labels. To run the algorithm, we must also select prior hyperparameter values, $\boldsymbol{A}_0$ and $\boldsymbol{\nu}_0$. The algorithm then proceeds as follows.

1. *Initialisation*: set arbitrary initial values for $\mathbb{E}_{\boldsymbol{\Pi}}[\ln \boldsymbol{\pi}_j^{(k)}]$ for all values $j = 1, .., J, k = 1, .., K$ and $\mathbb{E}_{\boldsymbol{\kappa}}[\ln \boldsymbol{\kappa}]$.

2. Update the distribution over the *target labels* $\mathbb{E}_{t_i}[t_i = j]$. For $i = 1, .., N, j = 1, .., J$:

   - Calculate ρ_{ij} by inserting current values of $\mathbb{E}_{\boldsymbol{\Pi}}[\ln \boldsymbol{\pi}_j^{(k)}]$ and $\mathbb{E}_{\boldsymbol{\kappa}}[\ln \boldsymbol{\kappa}]$ into Equation 3.6
   - Use ρ_{ij} to calculate $\mathbb{E}_{t_i}[t_i = j]$ according to Equation 3.7.

3. Update the expectations over the *model parameters*, $\mathbb{E}_{\boldsymbol{\Pi}}[\ln \pi_{jl}^{(k)}]$ and $\mathbb{E}_{\boldsymbol{\kappa}}[\ln \kappa_j]$. For $j = 1, .., J$ and $k = 1, .., K$:

   - Use current values of $\mathbb{E}_{t_i}[t_i = j]$ for $i = 1, .., N$ along with agents' responses $\boldsymbol{c}$ to update the counts N_j and $N_{jl}^{(k)}$ for $l = 1, .., L$ according to Equations 3.8 and 3.9
   - Use N_j to update hyperparameters $\boldsymbol{\nu}$ according to Equation (3.14)
   - Use $N_{jl}^{(k)}$ to update hyperparameters $\boldsymbol{\alpha}_j^{(k)}$ according to Equation 3.19
   - Insert $\boldsymbol{\nu}$ and $\boldsymbol{\alpha}_j^{(k)}$ into Equations 3.15 and 3.20 to find $\mathbb{E}_{\boldsymbol{\Pi}}[\ln \pi_{jl}^{(k)}]$ and $\mathbb{E}_{\boldsymbol{\kappa}}[\ln \kappa_j]$

4. *Check convergence*: if the target label distributions $\mathbb{E}_{t_i}[t_i = j]$ have not converged, repeat from step 2. Alternatively, check for convergence of the lower bound, $\mathcal{L}(q)$, calculated using equations in Subsection 3.2.4.

The *outputs* of the algorithm are:

- Target label predictions, given by the current values of the posterior expectations of the latent variables, $\mathbb{E}_{t_i}[t_i = j], i = 1, .., N, j = 1, .., J$. Each value is calculated by Equation 3.7.

- Approximate distributions over the model parameters, parametrised by the current values of the posterior hyperparameters $\boldsymbol{\nu}$ and $\boldsymbol{\alpha}_j^{(k)}$ for $j = 1, .., J$ and $k = 1, .., K$.

A pseudo-code representation of this algorithm is given in Listing B.1 to show how this may be implemented.

Convergence is guaranteed, regardless of the initial values for the parameter expectations. However, careful choice of initial values can reduce the number of iterations required to converge. Rather than select values at random, we can run a fast parameter estimation algorithm such as EM to set the initial values, or take expectations from the prior distributions of the parameters. The experiments below use the latter approach.

The choice of priors is important, especially when little training data is available. As was also explained in Chapter 2, the prior hyperparameters $\boldsymbol{\nu}_0$ and $\boldsymbol{A}_0$ are pseudo-counts, meaning they are equivalent to data that was observed *a priori*. Thus any prior beliefs, such as the expertise of an agent or the rarity of a target class, must be encoded as if we had previously observed a number of data samples. The strength of those beliefs is equivalent to the magnitude of the pseudo-counts. A number of techniques exist for optimising the hyperparameters [Bergstra and Bengio, 2012].

If there are known training labels, the distribution over these labels is not updated in step 2. Instead, the expected value is fixed at the known value, and is still used to update the counts N_j and $N_{jl}^{(k)}$ as described above.

The IBCC-VB algorithm is a *mean field VB* algorithm, since it applies mean field theory to optimise each factor (see Section 3.2.1). This type of algorithm consists of iterative steps that are comparable to the Expectation Maximisation (EM) algorithm [Dempster et al., 1977]. Step 2 is the variational equivalent of the E-step in EM, while step 3 corresponds to the M-step. In EM, the M-step finds maximum-likelihood or maximum-a-posteriori parameter estimates, whereas in VB, step 3 approximates the posterior expectation of the parameters. Unlike related EM algorithms, the output predictions of IBCC-VB naturally incorporate uncertainty from all levels of the model, as they marginalise approximately over unknown variables.

3.2.4 Variational Lower Bound

$\mathcal{L}(q)$ represents the lower bound of the log marginal likelihood, $\ln p(\boldsymbol{X})$ (see Equation (3.1)), which is also known as log *model evidence*. As such it is useful for model selection, assuming that models with greater evidence are a better fit for the problem. The lower bound can also be used to check for convergence of the algorithm, as it should always increase after a pair of *E-step* and *M-step* updates.

$$\begin{aligned}
\mathcal{L}(q) = & \iiint q(\boldsymbol{t}, \boldsymbol{\Pi}, \boldsymbol{\kappa}) \ln \frac{p(\boldsymbol{c}, \boldsymbol{t}, \boldsymbol{\Pi}, \boldsymbol{\kappa} | \boldsymbol{A}_0, \boldsymbol{\nu}_0)}{q(\boldsymbol{t}, \boldsymbol{\Pi}, \boldsymbol{\kappa})} \mathrm{d}\boldsymbol{t} \mathrm{d}\boldsymbol{\Pi} \mathrm{d}\boldsymbol{\kappa} \\
= & \ \mathbb{E}_{\boldsymbol{t}, \boldsymbol{\Pi}, \boldsymbol{\kappa}}[\ln p(\boldsymbol{c}, \boldsymbol{t}, \boldsymbol{\Pi}, \boldsymbol{\kappa} | \boldsymbol{A}_0, \boldsymbol{\nu}_0)] - \mathbb{E}_{\boldsymbol{t}, \boldsymbol{\Pi}, \boldsymbol{\kappa}}[\ln q(\boldsymbol{t}, \boldsymbol{\Pi}, \boldsymbol{\kappa})] \\
= & \ \mathbb{E}_{\boldsymbol{t}, \boldsymbol{\Pi}}[\ln p(\boldsymbol{c} | \boldsymbol{t}, \boldsymbol{\Pi})] + \mathbb{E}_{\boldsymbol{t}, \boldsymbol{\kappa}}[\ln p(\boldsymbol{t} | \boldsymbol{\kappa})] + \mathbb{E}_{\boldsymbol{\Pi}}[\ln p(\boldsymbol{\Pi} | \boldsymbol{A}_0)] + \mathbb{E}_{\boldsymbol{\kappa}}[\ln p(\boldsymbol{\kappa} | \boldsymbol{\nu}_0)] \\
& - \mathbb{E}_{\boldsymbol{t}}[\ln q(\boldsymbol{t})] - \mathbb{E}_{\boldsymbol{\Pi}}[\ln q(\boldsymbol{\Pi})] - \mathbb{E}_{\boldsymbol{\kappa}}[\ln q(\boldsymbol{\kappa})]. \qquad (3.21)
\end{aligned}$$

The expectation terms relating to the joint probability of the latent variables, observed variables and the parameters are

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{t}, \boldsymbol{\Pi}}[\ln p(\boldsymbol{c} | \boldsymbol{t}, \boldsymbol{\Pi})] &= \sum_{i=1}^{N} \sum_{k=1}^{K} \sum_{j=1}^{J} \mathbb{E}_{t_i}[t_i = j] \mathbb{E}_{\boldsymbol{\Pi}} \left[\ln \pi^{(k)}_{j c_i^{(k)}} \right] \\
&= \sum_{k=1}^{K} \sum_{j=1}^{J} \sum_{l=1}^{L} N_{jl}^{(k)} \mathbb{E}_{\boldsymbol{\Pi}} \left[\ln \pi_{jl}^{(k)} \right] \qquad (3.22) \\
\mathbb{E}_{\boldsymbol{t}, \boldsymbol{\kappa}}[\ln p(\boldsymbol{t} | \boldsymbol{\kappa})] &= \sum_{i=1}^{N} \sum_{j=1}^{J} \mathbb{E}_{t_i}[t_i = j] \mathbb{E}_{\boldsymbol{\kappa}} [\ln \kappa_j] \\
&= \sum_{j=1}^{J} N_j \mathbb{E}_{\boldsymbol{\kappa}}[\ln \kappa_j] \qquad (3.23) \\
\mathbb{E}_{\boldsymbol{\Pi}}[\ln p(\boldsymbol{\Pi} | \boldsymbol{A}_0)] &= \sum_{k=1}^{K} \sum_{j=1}^{J} \left\{ - \ln \mathrm{B} \left(\boldsymbol{\alpha}_{0,j}^{(k)} \right) + \sum_{l=1}^{L} \left(\alpha_{0,jl}^{(k)} - 1 \right) \mathbb{E}_{\boldsymbol{\Pi}} \left[\ln \pi_{jl}^{(k)} \right] \right\} \\
\mathbb{E}_{\boldsymbol{\kappa}}[\ln p(\boldsymbol{\kappa} | \boldsymbol{\nu}_0)] &= - \ln \mathrm{B}(\boldsymbol{\nu}_0) + \sum_{j=1}^{J} (\nu_{0,j} - 1) \mathbb{E}_{\boldsymbol{\kappa}}[\ln \kappa_j], \qquad (3.24)
\end{aligned}$$

where $\mathrm{B}(\boldsymbol{a}) = \frac{\prod_{l=1}^{L}\Gamma(a_l)}{\Gamma(\sum_{l=1}^{L} a_l)}$ is the Beta function and $\Gamma(a)$ is the Gamma function [Davis, 1965]. Terms in $\mathcal{L}(q)$ relating to the expectation of the variational distributions $\boldsymbol{q}$ are

$$\mathbb{E}_{\boldsymbol{t}}[\ln q(\boldsymbol{t})] = \sum_{i=1}^{N}\sum_{j=1}^{J} \mathbb{E}_{t_i}[t_i = j] \ln \mathbb{E}_{t_i}[t_i = j] \tag{3.25}$$

$$\mathbb{E}_{\boldsymbol{\Pi}}[\ln q(\boldsymbol{\Pi})] = \sum_{k=1}^{K}\sum_{j=1}^{J} \left\{ -\ln \mathrm{B}\left(\boldsymbol{\alpha}_{\boldsymbol{j}}^{(\boldsymbol{k})}\right) + \sum_{l=1}^{L} \left(\alpha_{jl}^{(k)} - 1\right) \mathbb{E}_{\boldsymbol{\Pi}}\left[\ln \pi_{jl}^{(k)}\right] \right\} \tag{3.26}$$

$$\mathbb{E}_{\boldsymbol{\kappa}}[\ln q(\boldsymbol{\kappa})] = -\ln \mathrm{B}\left(\boldsymbol{\nu}\right) + \sum_{j=1}^{J} \left(\nu_j - 1\right) \mathbb{E}_{\boldsymbol{\kappa}}[\ln \kappa_j] \tag{3.27}$$

where $\boldsymbol{N} = [N_1, ..., N_J]$ is a vector of counts for each true class. These equations simplify to give the following equation:

$$\mathcal{L}(q) = \sum_{j=1}^{J}\sum_{k=1}^{K} \left\{ \ln \frac{\mathrm{B}\left(\boldsymbol{\alpha}_{j}^{(k)}\right)}{\mathrm{B}\left(\boldsymbol{\alpha}_{0,j}^{(k)}\right)} - \sum_{i=1}^{N} \mathbb{E}[t_i = j] \ln \mathbb{E}[t_i = j] \right\} + \ln \frac{\mathrm{B}\left(\boldsymbol{\nu}\right)}{\mathrm{B}(\boldsymbol{\nu}_0)} \tag{3.28}$$

When performing the iterative update algorithm, calculating the lower bound is likely to be a more costly way to measure convergence than simply checking the expected target labels for convergence. However, monitoring the lower bound gives a clear measure of the rate of convergence and is a useful sanity check for any implementation of IBCC-VB.

3.3 Synthetic Data Experiments

This section assesses the efficacy of IBCC-VB using simulated data from Chapter 2. As before, the experiments evaluate the performance of the combination methods by plotting their *Receiver Operating Characteristic* (ROC) curves, calculating the Area Under Curve (AUC), and calculating Brier scores, as described in Subsection 2.5.1. As with IBCC-Gibbs, the outputs of IBCC-VB are the posterior probabilities of the target labels, $\boldsymbol{t}$, given by their expected values.

This section repeats only Experiments 5 (correlated agents) and 6 (training the combiner) from the previous chapter. In Experiment 6, there is no discernible difference in the

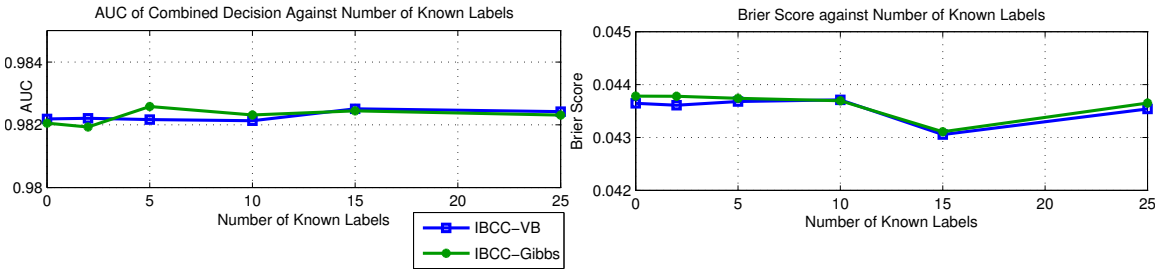


Figure 3.2: Performance of IBCC-VB with Experiment 6 with varying number of duplicate agents. Mean AUCs and Brier scores over 25 repeats.

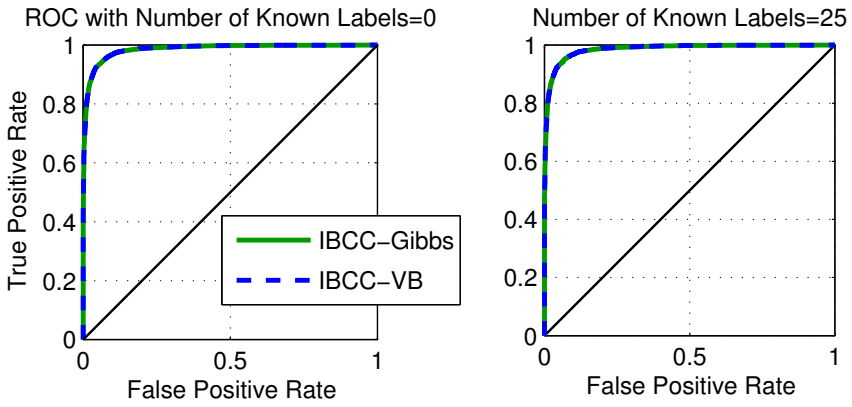


Figure 3.3: Receiver operating characteristics for two datasets from Experiment 6, showing performance of IBCC-VB.

AUC, Brier score or ROC curves of the VB and Gibbs' methods. These can be seen in Figures 3.2 and 3.3. In these tests, VB provided a good approximation in both unsupervised and supervised learning modes. Experiments 1 to 4 were also repeated and showed no difference in the results between the two methods.

In Experiment 5, we note a significant improvement in the mean AUCs of IBCC-VB over those of IBCC-Gibbs, shown in Figure 3.4. This difference is also reflected in the ROC curves in Figure 3.5, where the Gibbs' sampling approach results in a more angular plot. This ROC curve shape follows a similar pattern to Majority Voting (see Figure 2.7, for example). Probability estimates from the Gibbs' sampling algorithm may be more extreme, so that only a small number of threshold values change the true or false positive rates. In the ROC curve for IBCC-VB, this effect is reduced, although IBCC-VB still has a different angular shape. The Brier scores for the two methods are also almost identical, so probability estimates are likely to differ only by very small amounts between the two

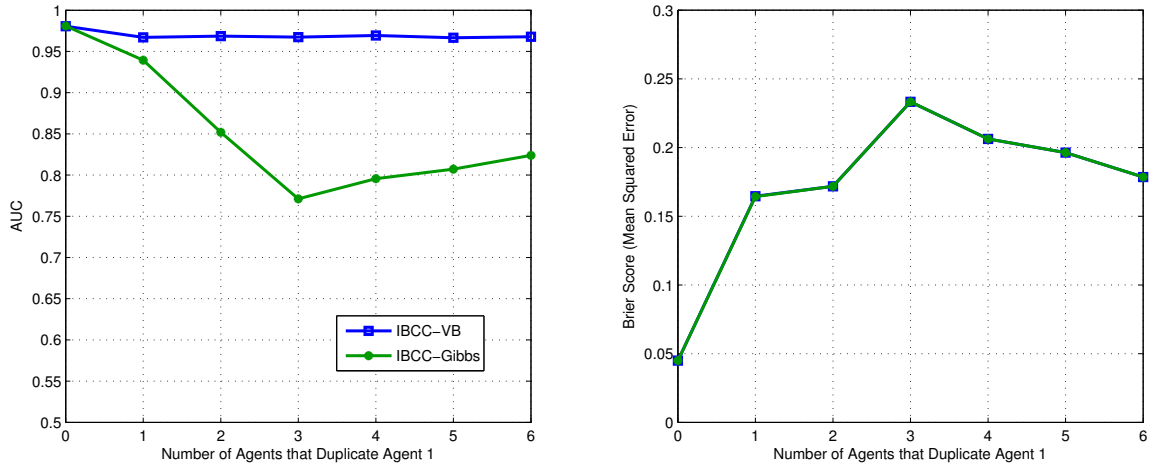


Figure 3.4: Performance of IBCC-VB with Experiment 5 with varying number of duplicate agents. AUCs and Brier scores are the means of 25 repeats.

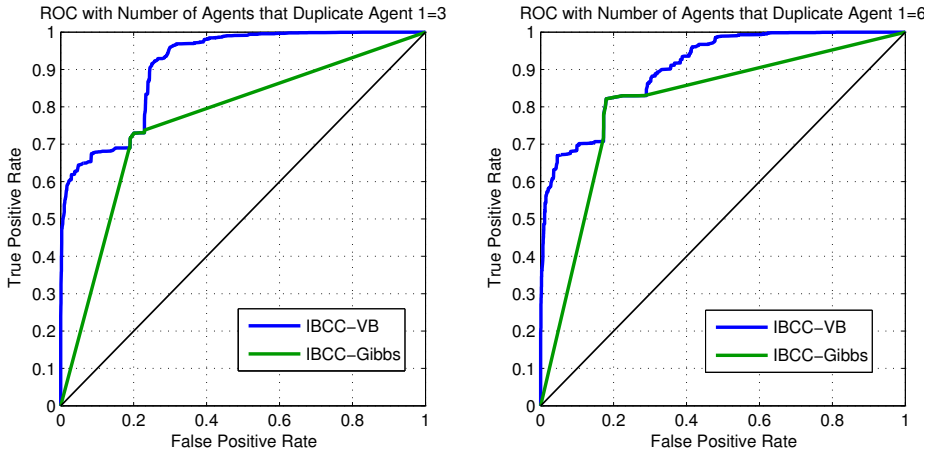


Figure 3.5: Receiver operating characteristics for two datasets from Experiment 5.

methods.

Note that the mean AUC in Figure 3.4 is approximately 0.97, whereas the ROC curves in Figure 3.5 have AUCs of approximately 0.91 and 0.90. The difference can be accounted for because the mean AUC was calculated by first drawing different ROC curves for each dataset, each of which may have different performances at different thresholds. Table 3.1 shows that the AUCs calculated separately for each test run are consistently high, as they have a low standard deviation for IBCC-VB. When combined in Figure 3.5, the same thresholds are applied to data from all datasets, which brings about a decrease in AUC. This is due to *threshold variance*, i.e. the variance in the true positive and false positive

rates at a given threshold value, over a number of test runs.

The AUC differences may result from the uni-modal nature of the variational distribution assumed by IBCC-VB. The true posterior of IBCC is bimodal in this application, with one mode corresponding to the correct target label assignment, and the other mode corresponding to the case that we swap the labels around. Gibbs sampling can explore both modes, which could increase confusion with some uncertain data points, whereas in this case the mode chosen by VB is correct. In any case, the VB algorithm appears to have produced a good approximation to IBCC, estimating a similar level of uncertainty in its predictions to Gibbs' Sampling, as shown by the similar entropies over the target labels $\boldsymbol{t}$, shown in Table 3.1.

Figure 3.6 compares the number of iterations required for each method to converge, showing the changing mean AUCs and entropy over the target labels $\boldsymbol{t}$. These graphs show an example using data from Experiment 1 in Chapter 2, although the same pattern was observed on various datasets, as later sections of this chapter will show. The plot shows the means over 25 runs, with error bars indicating one standard deviation. In comparison to Gibbs' sampling, the VB algorithm converges very quickly to a stable AUC and Entropy $H(\boldsymbol{t})$. When Gibbs' sampling is continued beyond 50 iterations, a slight improvement in the AUC over VB can be seen, which corresponds to slightly higher uncertainty $H(\boldsymbol{t})$. While a single iteration of each algorithm involves different calculations, in both cases the computational costs of a single iteration scale linearly with the number of variables that must be sampled or updated. Thus the substantial difference in convergence rates confirm

	Number of duplicate agents						
	0	1	2	3	4	5	6
Method Name	Standard deviation of AUC						
IBCC-VB	0.012	0.011	0.015	0.012	0.010	0.012	0.009
IBCC-Gibbs	0.012	0.032	0.058	0.076	0.081	0.078	0.065
	Mean entropy over target labels $H(\boldsymbol{t})$ (nats)						
IBCC-VB	150.106	34.183	24.905	35.911	17.900	8.488	17.408
IBCC-Gibbs	149.979	34.388	24.939	36.158	17.916	8.566	17.429

Table 3.1: Standard deviation in the AUC and mean entropy over target labels over 25 repeats of 10-fold cross-validation with Experiment 5.

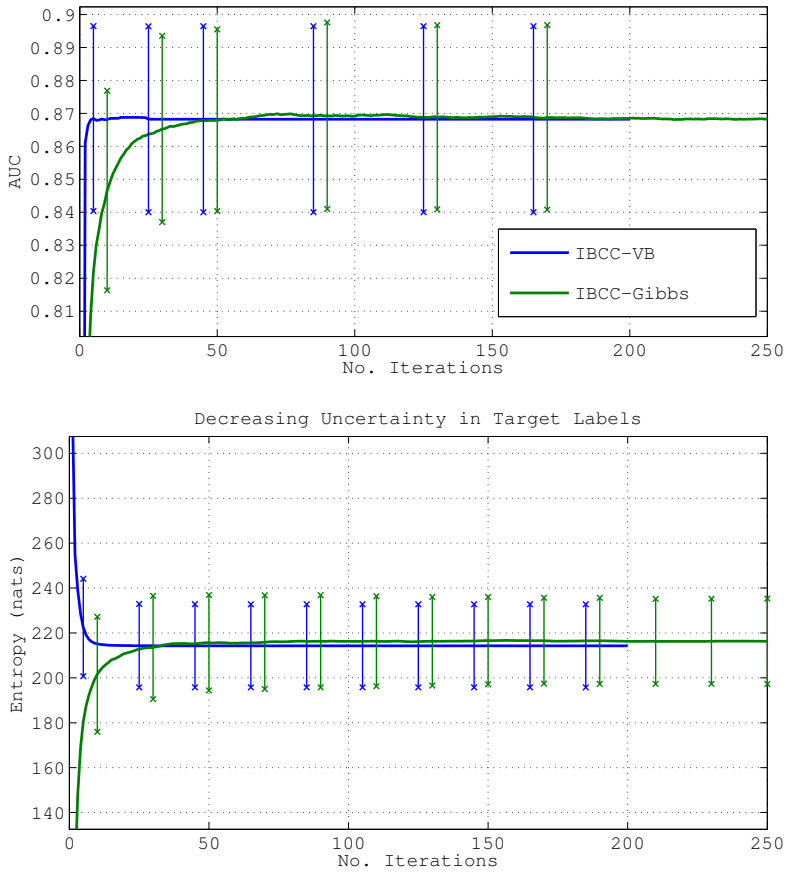


Figure 3.6: The improvement in the AUC (top) and entropy (bottom), $H(\boldsymbol{t})$, with increasing iterations of each IBCC algorithm. Run over data from the second test from Experiment 1, with sensor error rate $= 0.2$. Plot shows the means of 25 repeats, with error bars indicating one standard deviation, which are offset in each plot for clarity.

that IBCC-VB is a more computationally efficient approach.

With Gibbs' sampling, the entropy increases with each iteration until convergence, since we are initially averaging a small number of correlated samples from one part of the posterior distributions over each target t_i. Within this small sample, many draws of each target label t_i produce the same value $t_i = j$, so that the initial estimate of $\mathbb{E}[t_i = j|\boldsymbol{c}, \boldsymbol{\nu}_0, \boldsymbol{A}_0] = 1$ and hence t_i has low entropy. As more samples are received from the complete distribution over t_i, we estimate $\mathbb{E}[t_i = j|\boldsymbol{c}, \boldsymbol{\nu}_0, \boldsymbol{A}_0]$ by taking the mean of samples with both $t_i \neq j$ and $t_i = j$, increasing the entropy in uncertain target labels. With the VB algorithm, the entropy decreases as information is passed between variables in each iteration and the pseudo-counts of initially uncertain distributions increase until convergence.

3.4 Galaxy Zoo Supernovae Experiments

This section evaluates IBCC-VB using datasets taken from Galaxy Zoo Supernovae (GZSN)[2]. Each sample in the dataset is a triplet consisting of the Zooniverse user ID (a unique identifier for each agent), object ID and the score assigned by the user. In these experiments, the scores represent the agents' decisions $\boldsymbol{c}$ and may be -1, 1 or 3, indicating respectively that the agent has decided the candidate is not a supernova, may be a supernova, or is definitely a supernova. The results here compare the efficacy of IBCC using variational Bayes (IBCC-VB) and Gibbs sampling (IBCC-Gibbs) with Majority Voting, Weighted Majority, Weighted Sum and mean. The running implementation of Galaxy Zoo Supernovae used a sum of scores to filter the candidate objects, which is equivalent to taking the mean. For majority voting methods we treat scores of 1 as a single positive vote, and scores of 3 as three positive votes for the supernova class. This means that confident agents have more than one vote, but also that we do not discard uncertain scores of 1.

To verify the efficacy of our approach and competing methods, we use reliable target classifications obtained from full spectroscopic analysis, undertaken as part of the Palomar Transient Factory (PTF) collaboration [Law et al., 2009]. We note that this information was not available to the volunteers (the agents) as it was obtained retrospectively.

In the PTF-labelled GZSN dataset, there are approximately 3.5 times as many negative examples as positive examples. If trained on imbalanced data, the performance of classifiers can decrease [Weiss and Provost, 2001; He and Garcia, 2009]. To see why, consider that test data points are classified as either the majority or the minority class, depending on which region of input space they fall into. In this case the input space is the space of combinations of agents' decisions. If there are fewer examples of the minority class in the training data, some regions where the minority class is more likely are not labelled as such, since the training data does not contain minority examples from these regions. This can lead to misclassification. Therefore, this section consists of two experiments using

[2]Courtesy of Arfon Smith, Department of Physics (Astrophysics), University of Oxford, DWB, Keble Road, Oxford OX1 3RH, UK.

different datasets from GZSN:

1. *Balanced data* experiment: so that it is not affected by the class distribution particular to the GZSN data, we randomly select four equally-sized datasets, each containing the same number of positive and negative examples.

2. *Imbalanced data* experiment: the data is not resampled, hence the class distribution corresponds with the distribution of the PTF-labelled data points.

Real-world applications may use re-sampling to attain better performance so long as sufficient training data is available.

In both experiments, the decision combination methods were tested using ten-fold cross validation. With k-fold cross validation, the dataset is divided randomly into k partitions, then the algorithms are run k times, each with a different partition designated as the test data and the other partitions used as training data. In the test partition, the true labels are withheld from the decision combination algorithms and are used only to measure performance. Measuring the performance of the algorithms on a range of datasets gives a clearer understanding of their performance on other GZSN data in future [Bishop, 2006].

The hyperparameters $\boldsymbol{\alpha}_0$ for IBCC were chosen by the following process. First, calculate the proportions of each type of response for all agents in the complete dataset, without knowing the ground truth target labels. Then, adjust the proportions for each target class to reflect the belief that scores of -1 are more likely for the "not supernova" class, while scores of 3 are more likely for the supernova class. Finally, determine the magnitude of the counts. This should be large enough that when we update the hyperparameters with observed data points in Equation 3.19, very small numbers of observations do not overwhelm the priors. At the same time, the priors must be small enough that when we have many observations for an agent, the data can start to dominate. This process for setting the priors is therefore not an exact science, and it would be possible to further optimise them using techniques such as searching for the maximum model evidence. This was not necessary for the purposes of these experiments, where we were able to demonstrate the advantages of IBCC and compare inference methods without fine-tuning.

For IBCC-VB and IBCC-Gibbs	
For all agents, $\boldsymbol{\alpha}_{0,0} =$	$[25, 9, 2]$
For all agents, $\boldsymbol{\alpha}_{0,1} =$	$[23, 9, 3]$
$\boldsymbol{\nu}_0 =$	$[1000, 1000]$
For IBCC-Gibbs	
No. samples collected	5000
Sampling interval	5
No. burn-in samples	100
For Weighted Sum and Weighted Majority	
$\beta =$	0.9

Table 3.2: Hyperparameters and settings used in all GZSN experiments. Numbers in square brackets "[...]" represent elements in a vector.

3.4.1 Balanced Data Results

No. datasets	4			
No. positive samples per dataset	330			
No. negative samples per dataset	330			
Possible values of agent decisions	-1, 1, 3			
Target label values	0='not supernova', 1= 'supernova'			
Dataset ID	1	2	3	4
No. agents	1,986	1,644	1,691	1,921
No. responses from agents	10,280	9,901	9,623	9,502
Mean no. responses per agent	5.18	6.02	5.69	4.95
(± 1 standard deviation)	(19.43)	(27.59)	(24.94)	(24.62)

Table 3.3: Details of the balanced GZSN datasets.

Details of the datasets used for the balanced data experiment are given in Table 3.3. In this experiment, the ten-fold cross validation procedure was repeated four times, once for each of the balanced datasets specified in Table 3.3.

Figure 3.7 shows the ROC curves for each of the decision combination methods. The ROC curves were calculated from combined test data from all folds and all balanced datasets. The varying performance between each run of the algorithm is summarised by the mean and standard deviation of the area under the ROC curve (AUC). This is shown in Table 3.4, alongside the mean and standard deviation of the Brier score, which signifies

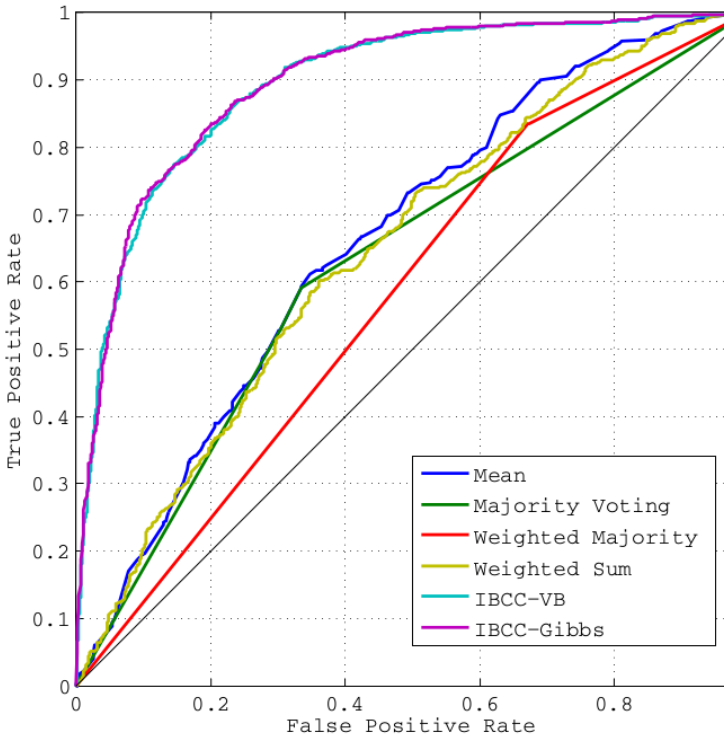


Figure 3.7: Galaxy Zoo Supernovae, balanced datasets: receiver operating characteristic (ROC) curves with 10-fold cross validation.

Method	AUC		Brier Score	
	Mean	S.D.	Mean	S.D.
Mean of Scores	0.654	0.135	0.193	0.105
Simple Majority	0.620	0.115	0.298	0.177
Weighted Majority	0.581	0.099	0.340	0.193
Weighted Sum	0.643	0.113	0.232	0.126
IBCC-VB	0.897	0.040	0.146	0.039
IBCC-Gibbs	0.897	0.040	0.137	0.033

Table 3.4: Galaxy Zoo Supernovae, balanced datasets: performance metrics of decision combination methods tested using 10-fold cross validation on 4 datasets.

the reliability of the probability estimates produced by each method. Both IBCC methods clearly outperform the alternatives, with IBCC-VB producing only a slightly higher Brier score. In contrast with the simulated datasets in the previous section, Weighted sum does not improve on the mean. The IBCC results both have lower variance than other methods, indicating that they may be more robust to changes in the dataset.

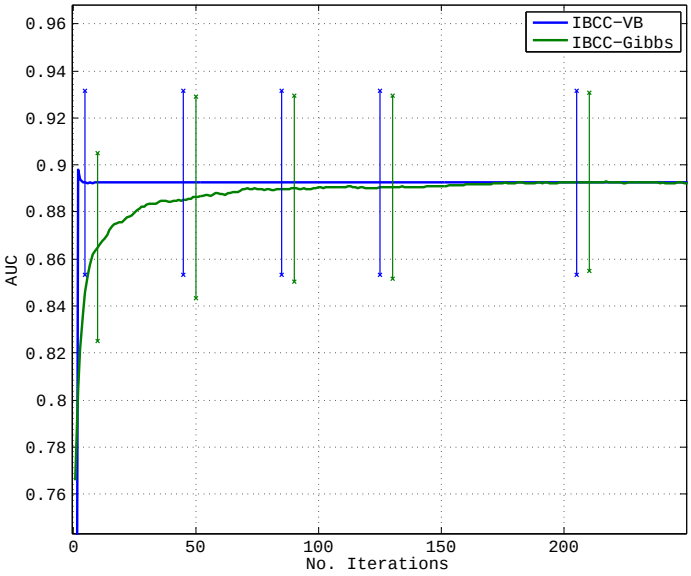


Figure 3.8: Galaxy Zoo Supernovae, balanced datasets: improvement in the AUC with increasing numbers of iterations. Continuous lines show the mean of the AUCs for each fold of each dataset, i.e. mean over AUCs calculated during each run of the algorithm. Error bars show one standard deviation from this mean and are offset in each plot for clarity.

The graph in Figure 3.8 shows that the AUC for IBCC-VB converged in 15 iterations, which is more than thirteen times quicker than the 200 iterations required for IBCC-Gibbs to reach the same AUC. Note that the AUC for IBCC-VB actually peaks before it has converged, which is the result of a slight initial over-fitting to the labelled training examples. Table 3.5 lists the mean times taken to run each algorithm for a single fold of one of the balanced datasets, showing that IBCC-VB is the fastest of the adaptive methods. Although these times seem very short, scalability becomes an issue when working with larger datasets where real-time updates are required. If computational cost does not prohibit running more than 200 iterations of Gibbs' sampling, it is possible to obtain further small improvements using IBCC-Gibbs, which lead to the marginally improved Brier scores in Table 3.4 when running up to 5000 iterations.

A second view of convergence is shown in Figure 3.9, which plots the entropy over the target labels $\boldsymbol{t}$ against the number of iterations. The VB method estimates the final entropy to be lower than that of Gibbs' sampling, which is able to fully explore the multi-modal posterior distribution, rather than using a uni-modal approximation. Since running Gibbs'

Method	Mean run-time (seconds)
Mean of scores	0.01
Simple majority voting	0.00
Weighted majority voting	2.39
Weighted sum	2.58
IBCC-VB (15 iterations)	0.40
IBCC-Gibbs(200 iterations)	3.33

Table 3.5: Time taken to obtain converged results for each decision combination method.

sampling to thousands of iterations produced little improvement, we can infer that for these datasets the variational approach produces a very accurate approximation with a far smaller computational cost and far out-performs the other methods tested.

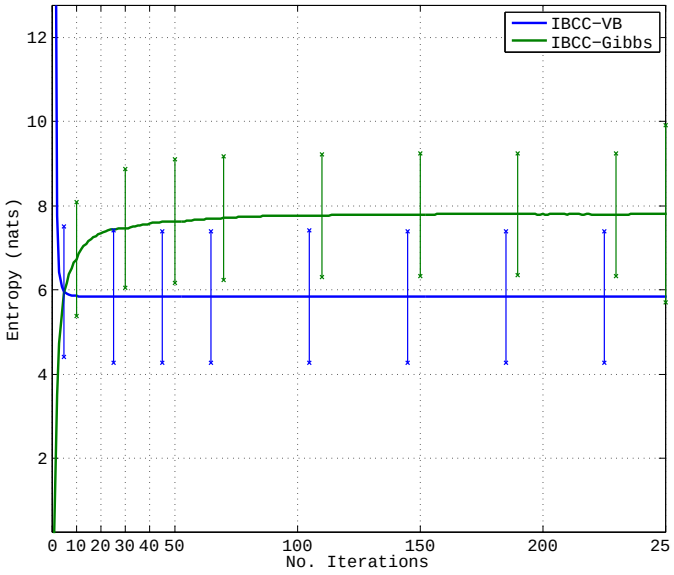


Figure 3.9: Galaxy Zoo Supernovae, balanced datasets: changing entropy of target labels with increasing numbers of iterations. Continuous lines show the mean of entropy for each fold of each dataset, i.e. mean target label entropy calculated during each run of the algorithms. Error bars show one standard deviation from this mean and are offset in each plot for clarity.

No. datasets	1
No. positive samples	330
No. negative samples	1,091
No. agents	2,726
No. responses from agents	21,113
Mean no. responses per agent (± 1 standard deviation)	7.75 (41.29)

Table 3.6: Details of the imbalanced GZSN dataset.

3.4.2 Imbalanced Data Results

In this experiment, rather than use re-sampling to balance the datasets, the combination methods are tested using 10-fold cross validation over the complete set of labelled data. In real-world applications, it is often possible to obtain nearly-balanced datasets by running the algorithm multiple times over small subsets of the test data, each with time using a much larger balanced training dataset. However, the re-sampling step adds complications and sufficient training data may not always be available. Dealing with imbalanced data is important for citizen science projects such as Galaxy Zoo Supernovae, where the aim is to filter candidate objects by identifying rare events or objects, in this case the occurrence of a supernova. Here, we show the performance of the algorithms on such a dataset with its inherent class proportions, i.e. the proportions of positive and negative examples equal to those of the complete set of ground truth target labels. Here, we are treating the reliable labels produced by the Palomar Transient Factory (PTF) [Law et al., 2009] using more expensive analysis techniques and additional data as the ground truth labels. The details of the dataset are given in Table 3.6.

Table 3.7 lists the AUCs and Brier Scores. In comparison with the Balanced datasets, the AUCs for IBCC-Gibbs and IBCC-VB have decreased by 0.064 and 0.073 respectively, while the Weighted Majority and Weighted Sum have increased slightly, but still not exceeded the un-weighted Mean. The Brier scores have changed similarly in the imbalanced dataset. Despite the performance changes, IBCC continues to significantly outperform the other methods. The VB algorithm has a slightly lower mean AUC, which can also be seen

Method	AUC		Brier Score	
	Mean	S.D.	Mean	S.D.
Mean of Scores	0.672	0.041	0.191	0.030
Simple Majority	0.633	0.057	0.247	0.089
Weighted Majority	0.593	0.034	0.251	0.091
Weighted Sum	0.660	0.044	0.200	0.040
IBCC-VB	0.824	0.027	0.166	0.016
IBCC-Gibbs	0.833	0.033	0.152	0.022

Table 3.7: Galaxy Zoo Supernovae, imbalanced dataset: performance metrics of decision combination methods tested using 10-fold cross validation.

in the ROC curve in Figure 3.10, but is nonetheless very close.

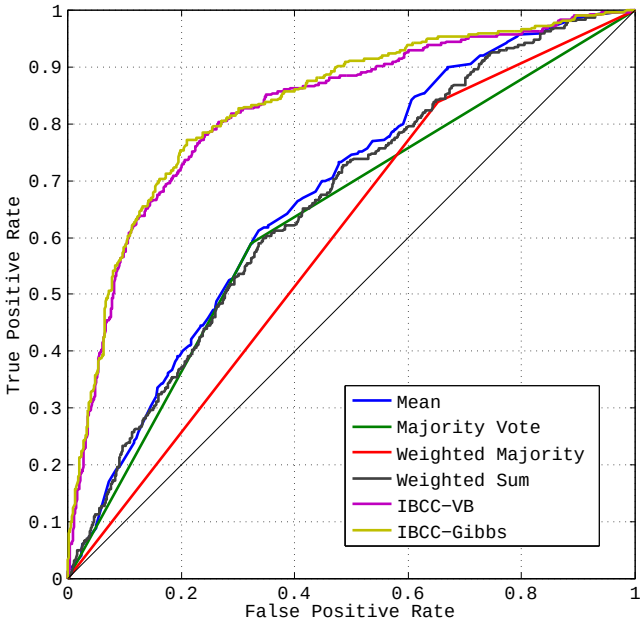


Figure 3.10: Galaxy Zoo Supernovae, imbalanced dataset: receiver operating characteristic (ROC) curves with 10-fold cross validation.

With the imbalanced dataset, VB also converges far more rapidly than Gibbs' sampling, and again shows a small drop after two iterations, which may be due to slight over-fitting. Figure 3.11 shows that the two methods converge at similar rates to with the balanced data, although Gibbs' sampling catches up with VB after 30 iterations and continues to improve very slowly until around 100 iterations.

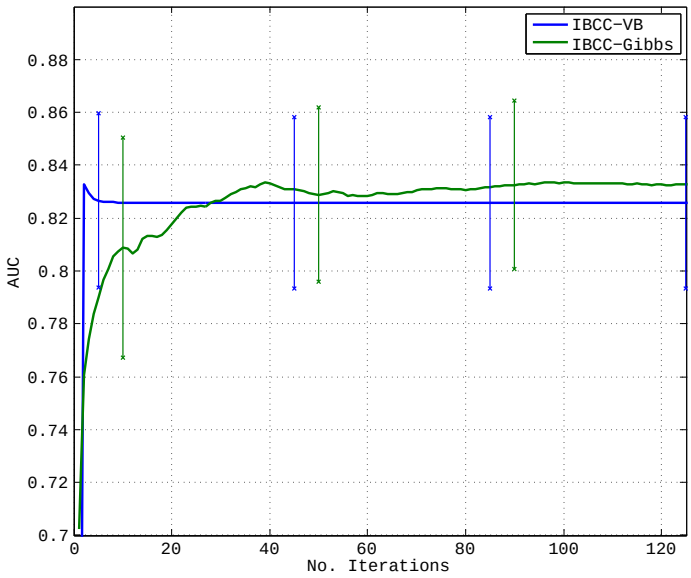


Figure 3.11: Galaxy Zoo Supernovae, imbalanced dataset: improvement in the AUC with increasing numbers of iterations. Continuous lines show the mean of the AUCs for each fold of each dataset, i.e. mean over AUCs calculated during each run of the algorithm. Error bars show one standard deviation from this mean.

3.5 Galaxy Zoo Mergers Experiment

This section provides further empirical results for IBCC-VB using another citizen science project, Galaxy Zoo Mergers[3]. This project operates in the same manner as Galaxy Zoo Supernovae, with volunteer agents answering questions to produce one of three possible scores, with the aim of identifying merging galaxies from images. A balanced dataset was sampled at random for this experiment and is detailed in Table 3.8.

No. datasets	1
No. positive samples per dataset	1628
No. negative samples per dataset	1629
Possible values of agent decisions	1, 2, 3
Target label values	0=’no galaxy merger’, 1=’galaxy merger’
No. agents	18626
No. responses from agents	146529
Mean no. responses per agent (± 1 standard deviation)	7.87 (18.12)

Table 3.8: Details of the Galaxy Zoo Mergers dataset.

[3]Data provided courtesy of Chris Lintott. See also http://mergers.galaxyzoo.org/

Method	AUC		Brier Score	
	Mean	S.D.	Mean	S.D.
Mean of Scores	0.718	0.040	0.279	0.028
Simple Majority	0.655	0.026	0.346	0.039
Weighted Majority	0.653	0.016	0.346	0.041
Weighted Sum	0.711	0.034	0.285	0.031
IBCC-VB	0.771	0.035	0.238	0.025
IBCC-Gibbs	0.764	0.035	0.237	0.026

Table 3.9: Galaxy Zoo Mergers: performance metrics of decision combination methods tested using 10-fold cross validation.

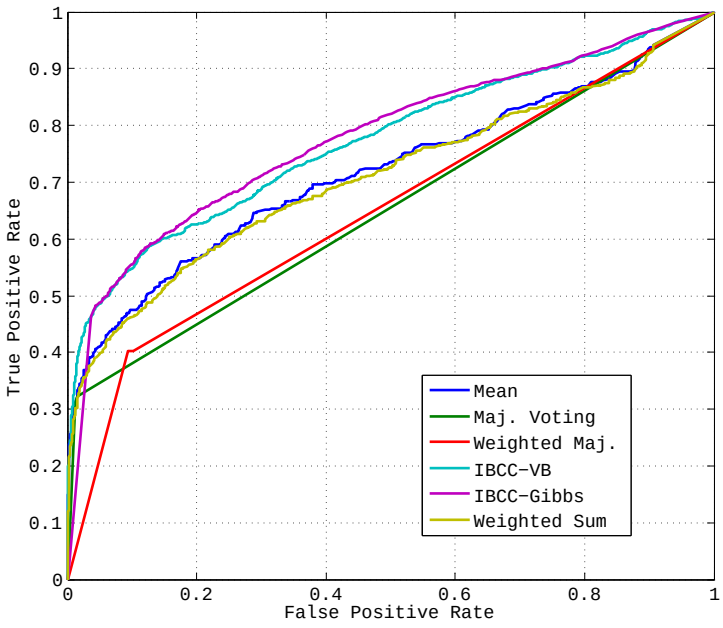


Figure 3.12: Galaxy Zoo Mergers, balanced datsets: receiver operating characteristic (ROC) curves with 10-fold cross validation.

The resulting AUCs and Brier scores are given by Table 3.9 and the ROC curve is shown in Figure 3.12. In comparison to Galaxy Zoo Supernovae, we again see significant improvements when using IBCC over alternative methods, and in this case the VB approximation marginally outperforms the Gibbs' sampling algorithm. The advantage of IBCC over other methods is smaller for Galaxy Zoo Mergers. This may be because agents' responses are simply less correlated with target labels, so contain more random noise and

less information, meaning that the *Bayes' error*, which refers to the minimum attainable error, is higher. Thus, more informative responses would be needed to produce better results. Alternatively, there may simply be fewer responses per agent, meaning that the IBCC model of the confusion matrices $\mathbf{\Pi}$ is less accurate.

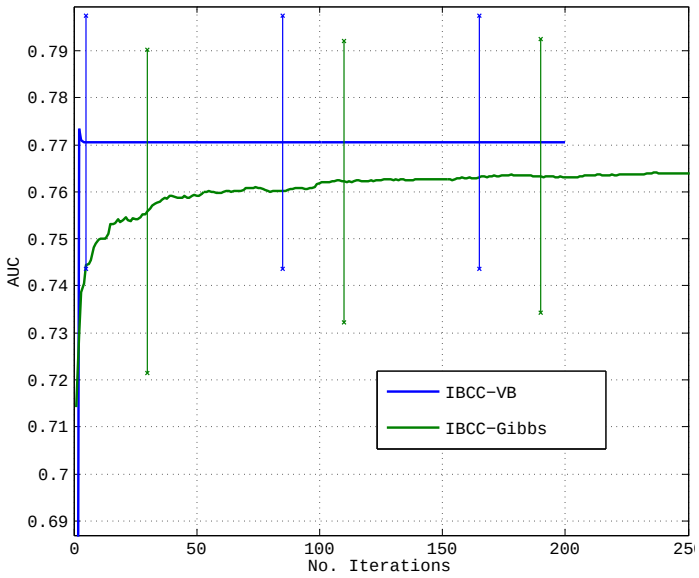


Figure 3.13: Galaxy Zoo Mergers: improvement in the AUC with increasing numbers of iterations. Continuous lines show the mean of the AUCs for each fold of each dataset, i.e. mean over AUCs calculated during each run of the algorithm. Error bars show one standard deviation from this mean.

The convergence of the IBCC algorithms is charted in Figure 3.13, showing a similar pattern to the GZSN datasets. Gibbs' sampling appears a little slower, with slight improvements in the AUC continuing beyond 250 iterations.

3.6 HTTP Web Attack Classification

No. training samples	17828
No. testing samples	24630
No. binary features	1328
No. classes (attack types)	8

Table 3.10: The HTTP Web Attack dataset.

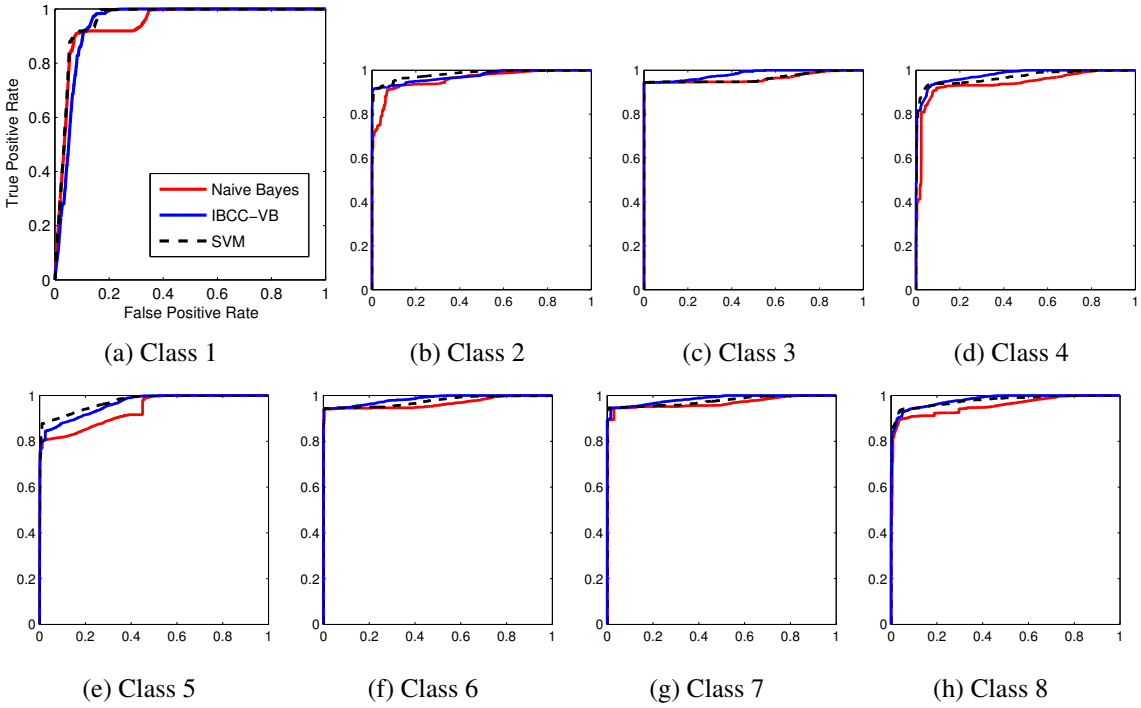
The applications of IBCC extend beyond purely combining decisions from human agents. Since IBCC effectively treats agents' decisions as data, other kinds of data can be combined in a similar manner, such as features extracted by computers rather than humans. This example considers a scenario where HTTP web requests are analysed to detect whether they are attempts to attack a web server. Each web request takes the form of a text document, from which a number of tokens were extracted. The presence of a particular token indicates that a corresponding binary feature is true. There are 7 attack types, plus a non-attack class. Unlike the agents' decisions in the Citizen Science scenarios, we have no prior knowledge of whether a particular feature indicates a particular attack type, and while there are 8 target classes, each feature may only be 0 or 1. Instead we rely entirely on training data to learn confusion matrices for the binary text features using the IBCC-VB and IBCC-Gibbs algorithms. Basic statistics for the web attack dataset are shown in Table 3.10. The dataset was first proposed for a Machine Learning challenge, in which it was divided into training data and test samples [Gallagher and Eliassi-Rad, 2008]. This experiment evaluates IBCC-VB over the same division of samples.

The IBCC-VB algorithm is compared with a *Multinomial Naïve Bayes* classifier (MNB), which has been established as very effective with bag-of-words feature models, including binary feature models [Schneider, 2004]. As with IBCC, the MNB classifier assumes independence between features given the target class label. However, this multinomial variant of Naïve Bayes does not learn confusion matrices, but learns the probability that the next feature drawn will be a particular token. So, for one observation (an HTTP request document), the observed data (the tokens) represent a number of independent draws

from a single multinomial distribution, with outcomes corresponding to each possible feature. With IBCC, on the other hand, the tokens are assumed to be draws from independent Bernoulli distributions. Another variant of the Naïve Bayes classifier, the *Multivariate Bernoulli* model also models features in this way [Metsis et al., 2006], but does not do so in a Bayesian paradigm. Importantly, the confusion matrices mean that IBCC takes into account non-occurrences of features, which are assumed to be draws with an outcome of 0. MNB considers only the features present in a document to be draws from the multinomial likelihood distribution. Each of these models is suited to different situations. For example, when each object contains only a small fraction of possible features, the evidence from features that are not present can dominate the multivariate Bernoulli model. This has been shown to lead to biased predictions [Schneider, 2004] when the number of features present differs between classes, since absent features affect one class more than the other. Intuitively, the IBCC or multivariate Bernoulli models would be suitable if the absence of a feature can be considered as negative evidence rather than as no evidence at all. In situations where the size of the document can be small, it might be more reasonable to assume that the absence of a feature provides no evidence, i.e. it is equivalent to an agent providing no response. Given a fixed size of document and a set of features that are already present, it is not possible for any more features to occur in that document, so the assumption of conditional independence between features no longer holds. In the case of HTTP web attacks, it is not obvious which model is more suitable for the features, since the absence of some tokens might be strong evidence that a particular attack type is not taking place, whereas tokens not related to an attack might not provide negative evidence. Thus, this experiment is intends to test the applicability of the IBCC model in the cyber security setting.

A further distinction between the Naïve Bayes approaches and IBCC-VB is that Naïve Bayes uses separate training and prediction phases, operating in a fully supervised manner. The semi-supervised IBCC-VB simultaneously learns from training data and latent structure in the test data, while inferring target labels. The Naïve Bayes classifier used here also differs from the NB-LogOP used in earlier experiments, since the NB-LogOP models the

likelihood of log-odds estimates from a base classifier, whereas the Naïve Bayes classifier used here models the likelihood of discrete features.



(a) Class 1 (b) Class 2 (c) Class 3 (d) Class 4

(e) Class 5 (f) Class 6 (g) Class 7 (h) Class 8

Figure 3.14: Web attack dataset, ROC curves for each attack class.

MNB and IBCC-VB were evaluated against a support vector machine (SVM) with a radial basis kernel function, which is an established classification algorithm described in [Bishop, 2006]. A detailed comparison between the three methods can be seen in the ROC curves in Figure 3.14, showing how IBCC-VB outperforms MNB, and for some classes, the SVM. The results in Table 3.11 compare MNB and IBCC-VB to and current state-of-the-art results published in [Gallagher and Eliassi-Rad, 2008] using the AUC and accuracy of discrete decisions obtained by assigning the most probable class. The mean AUCs given here enable a comparison with the previously-published results, for which only a summary was provided. The method used by [Gallagher and Eliassi-Rad, 2008] built a canonical example of each class by combining feature vectors from training data. New examples were classified by finding the most similar canonical example. Features were extracted using *term-frequency inverse-document-frequency (TF-IDF)* [Baeza-Yates and Ribeiro-Neto, 1999]. Despite using the raw tokens without TF-IDF pre-processing, IBCC-VB achieves an equally high AUC and accuracy, while the MNB approach is slightly worse.

Method	Mean AUC	Accuracy
IBCC-VB	0.97	0.94
IBCC-VB, 250 test features only	0.89	0.89
Multinomial Naïve Bayes	0.95	0.88
TF-IDF	0.97	0.94
SVM	0.97	0.90

Table 3.11: HTTP Web Attack dataset: performance over test data. The mean AUC was calculated by averaging over all classes, weighted by the number of instances in each class. Mean AUC values for TF-IDF were given in [Gallagher and Eliassi-Rad, 2008]; the method was not re-run here.

While the SVM produces a comparable AUC to IBCC-VB, IBCC-VB attains a higher accuracy, possibly due to better probability estimates resulting from the fully-Bayesian approach that the SVM does not use. It is also worth noting that the training phase of the SVM took several times longer than that of IBCC-VB.

An advantage of IBCC-VB over the TF-IDF method is that it provides a natural way to perform feature selection. We can calculate the expected information gain that each feature provides over the target labels, then select features that maximise the expected information gain. We start by learning the IBCC model on the training data only, then calculate the expected information gain over test target labels from each feature. We can calculate the expected information gain $\mathbb{E}[I(\boldsymbol{t}; \boldsymbol{c}^{(k)})]$ about the target labels $\boldsymbol{t}$ of document i given the values $\boldsymbol{c}^{(k)}$ of feature k:

$$\mathbb{E}[I(\boldsymbol{t}; \boldsymbol{c}^{(k)})] = H(\boldsymbol{t}) - \sum_{i=1}^{N} \sum_{l=1}^{L} p(c_i^{(k)} = l) H(t_i | c_i^{(k)} = l), \tag{3.29}$$

where L is the number of values that $c_i^{(k)}$ can take and H is the Shannon entropy. Although not optimal, a simple method for selecting features is to greedily choose those with the highest expected information gain. The features for the web attack dataset are plotted in Figure 3.15, showing that a small number of the 1328 features are significantly more informative than the others. Selecting the highest 250 features leads to an AUC of 0.89,

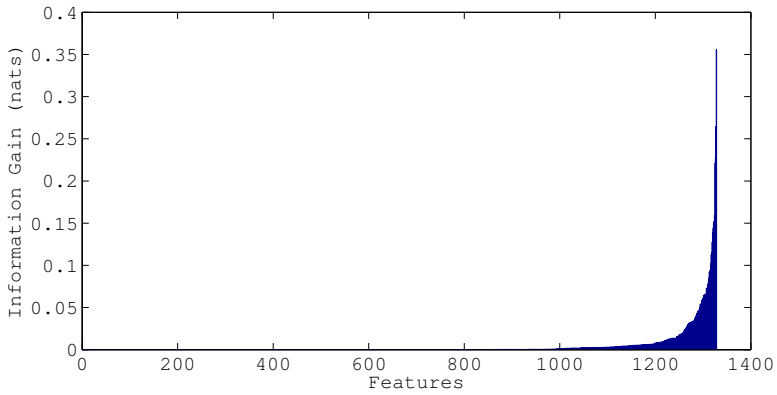


Figure 3.15: Features ranked by expected information gain over object class priors.

confirming that much information is retained, even with this very simple feature selection procedure. Reducing the number of features substantially decreases the computational cost of IBCC-VB. Chapter 5 considers this idea in more detail, looking at the case where obtaining feature values or agents' decisions is also costly.

These experiments demonstrate the flexibility of IBCC, showing it can be readily applied to text features as well as decisions from Machine Learning classifiers or human agents. While we establish that it performs well, it is possible that performance could be increased further by altering the the feature model, for instance by using the multinomial feature model similar to MNB for some features. The advantages of IBCC may also stem partly from its application of the fully Bayesian approach, learning distributions over parameters in a semi-supervised manner, rather than estimating point values.

3.7 Analysing Communities of Agents

This chapter has so far focused on the challenge of inferring target labels by combining responses from multiple agents or features. Learning about the behaviour of those agents or the information contained in the features is also very valuable when managing a system with multiple information sources. The previous section, 3.6, touched on this briefly, showing how IBCC can be used to perform feature selection. This section proposes a novel application of recent community detection methodology to analyse groupings of agents with

similar behaviour. This application exploits the confusion matrices, $\mathbf{\Pi}$, learnt by IBCC to provide an insight into behavioural patterns within a pool of agents. Such information enables system designers to identify problems facing particular subsets of users, so that they may be rectified to obtain more informative decisions. For example, the designer can modify user interfaces or alter the questions asked. Understanding the agents also allows the system to automatically optimise the multi-agent system, by deploying individuals to specific tasks, as will be shown in Chapter 5. The work here uses Galaxy Zoo Supernovae (GZSN) as a case study. The first subsection uncovers distinct types of user by grouping similar confusion matrices, while the second subsection investigates the effect of completing particular tasks on the agents' behaviour.

3.7.1 $\mathbf{\Pi}$ Communities

Community detection is the process of clustering a "similarity" or "interaction" network, so that agents within a given group are more strongly connected to each other than the rest of the graph. Identifying overlapping communities in networks is a challenging task. In recent work [Psorakis et al., 2011] presented a novel approach to community detection that infers such latent groups in the network by treating communities as explanatory latent variables for the observed connections between nodes, so that the stronger the similarity between two decision makers, the more likely it is that they belong to the same community. Such latent grouping is extracted by an appropriate factorisation of the connectivity matrix, where the effective inner rank (number of communities) is inferred by placing shrinkage priors [Tan and Févotte, 2009] on the elements of the factor matrices. The scheme has the advantage of soft-partitioning solutions, assignment of node participation scores to communities, an intuitive foundation and computational efficiency.

We apply the approach described in [Psorakis et al., 2011] to a similarity matrix calculated over all the citizen scientists in our study, based upon the expected values of each users' confusion matrix. To characterise the behaviour of the agents, expectations are taken over the distributions of the confusion matrices inferred using IBCC-VB over the imbal-

anced dataset in Section 3.4.2. Denoting $\mathbb{E}[\pi^{(k)}]$ as the (3×2) confusion matrix inferred for agent k, we may define a simple similarity measure between agents m and n as

$$V_{m,n} = \exp\left(-\sum_{j=1}^{J} \mathcal{HD}^2\left(\mathbb{E}[\boldsymbol{\pi}_j^{(m)}], \mathbb{E}[\boldsymbol{\pi}_j^{(n)}]\right)\right), \tag{3.30}$$

where $\mathcal{HD}^2()$ is the *squared Hellinger distance* between two distributions, meaning that two agents who have very similar confusion matrices will have high similarity. Since the confusion matrices are multinomial distributions, squared Hellinger distance is calculated as:

$$\mathcal{HD}^2\left(\mathbb{E}[\boldsymbol{\pi}_j^{(m)}], \mathbb{E}[\boldsymbol{\pi}_j^{(n)}]\right) = 1 - \sum_{l=1}^{L} \sqrt{\mathbb{E}[\pi_{j,l}^{(m)}]\mathbb{E}[\pi_{j,l}^{(n)}]} \tag{3.31}$$

As confusion matrices represent probability distributions, so Hellinger distance is chosen as an established, symmetrical measure of similarity between two probability distributions [Bishop, 2006]. Taking the exponential of the negative squared Hellinger distance converts the distance measure to a similarity measure with a maximum of 1, emphasising cases of high similarity.

Application of Bayesian community detection to the matrix $\boldsymbol{V}$ robustly gave rise to *five* distinct groupings of users. In Figure 3.16 we show the centroid confusion matrices associated with each of these groups of citizen scientists. The centroids are the expected confusion matrices of the individuals with the highest node participation scores for each community. The labels indicate the target label (0 for "not supernova" or 1 for "supernova") and the preference for the three scores offered to each user by the Zooniverse questions (-1, 1 & 3). Group 1, for example, indicates users who are clear in their categorisation of "not supernova" (a score of -1) but who are less certain regarding the "possible supernova" and "likely supernova" categories (scores 1 & 3). Group 2 are "extremists" who use little of the middle score, but who confidently and correctly use scores of -1 and 3. By contrast group 3 are users who almost always use score -1 ("not supernova") whatever objects they are presented with. Group 4 incorrectly avoid scores of -1 and, finally, group 5 consists of "non-committal" users who rarely assign a score of 3 to supernova objects, preferring

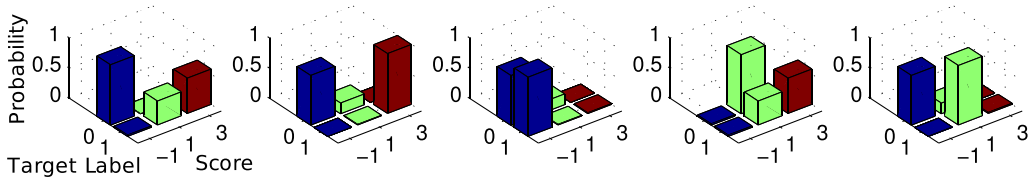


Figure 3.16: Prototypical confusion matrices for each of the five communities inferred using Bayesian social network analysis. Each graph corresponds to the most central individual in a community, with bar height indicating probability of producing a particular score for a candidate of the given target label.

the middle score ("possible supernova"). It is interesting to note that all five groups have similar numbers of members (several hundred) but clearly each group indicates a very different approach to decision making. It is possible that different groups relate to difficulty with certain questions presented by the system as part of the classification process, for example, always answering "no" to "is the candidate centered in a circular host galaxy" will prevent the score of 3 from being assigned, possibly leading to the behavioural pattern of group 5. Membership of a particular community may therefore inform what kind of help the system gives to individuals to help them perform better.

3.7.2 Common Task Communities

This section examines groups of agents that have completed similar sets of classification tasks, referred to here as *common task communities*. Such communities may indicate agents that choose to complete similar tasks, rather than skipping them, or volunteers that participate over similar periods. Analysing the confusion matrices associated with these communities also indicates whether there any particular behavioural characteristics, such as a bias toward giving a particular answer, which relate to membership of a community with shared experiences.

First, a larger set of Galaxy Zoo Supernovae data is obtained, which includes unlabelled objects, since these may be part of a shared experience that affects relationships between agents. The dataset contains all data from the imbalanced dataset used in Section 3.4.2, along with any unlabelled observations over the same period, resulting in a larger dataset

of $493,048$ classifications.

The overlapping community detection method [Psorakis et al., 2011] is now applied to a co-occurrence network defined as follows. Each edge connect citizen scientists that have completed a common task, where edge weights $w_{m,n}$ reflect the proportion of tasks common to both individuals, such that

$$w_{m,n} = \frac{\text{number\_of\_common\_tasks}(m,n)}{N^{(m)} + N^{(n)}} \,, \tag{3.32}$$

where $N^{(m)}$ and $N^{(n)}$ are the total numbers of objects seen by agents m and n respectively. The normalisation term $N^{(m)} + N^{(n)}$ reduces the weight of edges from agents that have completed large numbers of tasks, as these would otherwise have strong links to a large number of other agents who share proportionally few common classifications. The edge weights capture the proportion of common tasks that individuals have completed and estimate in a simple manner the probability that an object i that has been classified by either agent m or n will have been classified by both agents. For agents that have made few classifications, the low values of $N^{(m)}$ and $N^{(n)}$ can cause the strength of weights to be over-exaggerated. A possible remedy is to treat the problem in a Bayesian manner, placing priors over $w_{m,n}$, but in practice there is insufficient information to make a confident clustering of these agents, so their inclusion does not inform our analysis of the communities and may add noise. Therefore, these experiments filter out agents with fewer than 10 classifications, so the dataset size of $493,048$ classifications excludes observations from such agents.

The algorithm found 32 communities for 2,131 citizen scientists and produced a strong community structure with modularity of 0.75. Modularity is a measure between -1 and 1 that assumes a strong community structure has more intra-community edges (edges that connect nodes in the same community) than inter-community edges. It is defined as the fraction of intra-community edges minus the expected fraction of intra-community edges for a random graph with the same node degree distribution [Girvan and Newman, 2002]. In Galaxy Zoo Supernovae, this very modular community structure may arise through users

with similar preferences or times of availability being assigned to the same objects. At a given point in time, Galaxy Zoo Supernovae prioritises the oldest objects that currently lack a confident classification and assigns these to the next available citizen scientists. It also allows participants to reject tasks if desired, for example if it is too difficult or repetitive. Common task communities may therefore form where decision makers have similar abilities, preferences for particular tasks (e.g. due to interesting features in an image) or are available to work at similar times. When considering the choice of decision makers for a task, these communities could therefore inform who is likely to be available and who will complete a particular task.

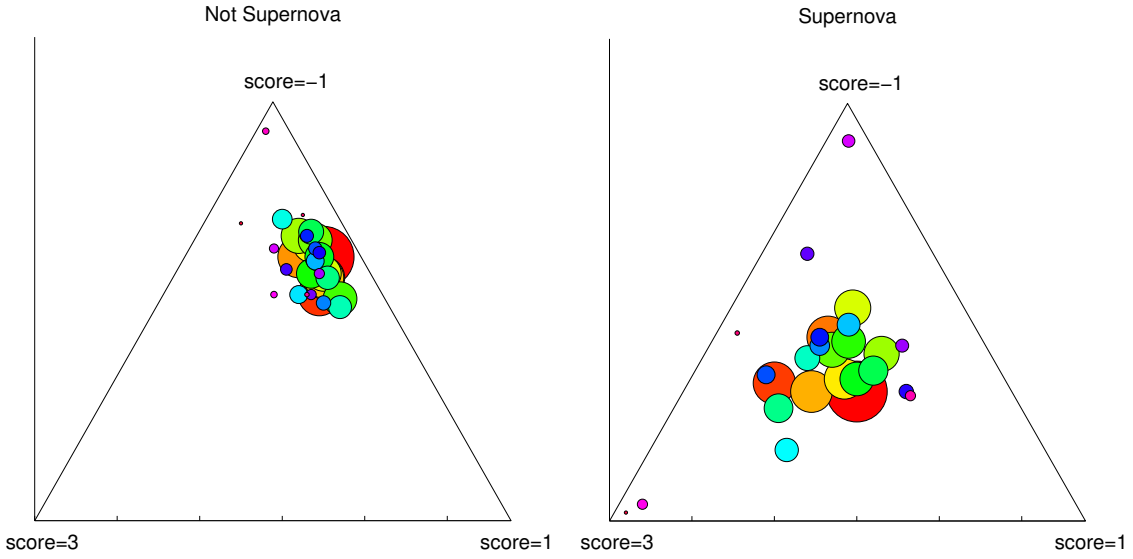


Figure 3.17: Mean expected confusion matrices of common task communities. Proximity to a vertex indicates likelihood of a score given the target class shown above each plot. Size of circles indicates number of community members.

Figure 3.17 plots the distribution of the means of the community members' confusion matrices for each of the target values. The images show ternary plots where each circle represents one community g, and its position indicates the mean of the expectations of its agents' response likelihoods for target value j:

$$\bar{\mathbb{E}}[\boldsymbol{\pi}_j^{(k)}]_g = \frac{1}{N_\mathrm{m}} \sum_{k=1}^{N_\mathrm{g}} \mathbb{E}[\boldsymbol{\pi}_j^{(k)}], \tag{3.33}$$

where N_g is the number of members in community g. Proximity to a vertex indicates the probability of a score given an object with the stated target label, e.g. in the graph labelled "Supernova", a point near the vertex labelled "score = -1" indicates a very high probability of giving a decision of -1 when presented with images of a genuine supernova. The left-hand plot shows the mean confusion matrices for the target label "not a supernova"; the right-hand plot shows the confusion matrices for the target label "supernova". The size of the nodes indicates the number of members of the cluster. Differences between communities for the label "not supernova" are less pronounced than for the label "supernova". For the latter class we have only 330 objects as opposed to 1,091 for the former, so the agents see fewer tasks with target label "supernova". This means that individual tasks with features that are more likely to elicit a certain agent response can have a greater effect on the confusion matrices learned. For instance, some tasks may be easier to classify than others or help a decision maker learn through the experience of completing the task, thus affecting the confusion matrices we infer. As we would expect, some smaller communities have more unusual means as they are more easily influenced by a single community member. The effect of this is demonstrated by the difference between community means in Figure 3.17 for groups of decision makers that have completed common sets of tasks. In summary, while common task communities are highly modular, a strong effect on confusion matrices is not evident from the current analysis of the communities.

3.8 Conclusions

This chapter first proposed a scalable, computationally efficient, Bayesian approach to decision combination, IBCC-VB. Over a series of simulated and real-world applications, IBCC-VB was shown to produce equivalent results at a far lower computational cost, compared to learning the exact posterior using a Gibbs' sampling algorithm. The methodology advocated in this chapter learns the reliability of individual agents, which naturally enables feature selection and the analysis of agent behaviour. A proposed method using community detection to extract groupings of agents showed that there are distinct types of user within

the case study application, Galaxy Zoo Supernovae. The information learnt by IBCC-VB suggests the opportunity to optimise the deployment of agents to suitable tasks or training. However, the current approach is limited in that it cannot detect when agents improve or alter their behaviour. The next chapter therefore extends IBCC to dynamic confusion matrices, and the following chapter shows how this extension facilitates intelligent task assignment.

Chapter 4

Modelling the Dynamics of Agents

In real-world applications such as Galaxy Zoo Supernovae, the behaviour of agents is not necessarily consistent over time. Many agents, especially humans, adapt as they learn, or their goals or mood change over time. The environment may also vary, as may the interface through which systems ask agents for information, or indeed the target concepts themselves. In a multi-agent system, a central decision-making node can influence these dynamics to optimise the workforce through intelligent choice of agents, tasks, questions and training. However, this kind of weak control presupposes the ability to track changing agent behaviour and monitor the effects of attempts to influence it. The IBCC model that was shown to perform well in previous chapters models agent behaviour through their confusion matrices, $\mathbf{\Pi}$, but these are assumed to be constant over time. This chapter therefore proposes a dynamic variant of IBCC, *DynIBCC*, that can track an agent's changing behaviour through dynamic confusion matrices.

Section 4.1 presents an overview of the DynIBCC model. The following section, 4.2 gives further background on how the dynamics are modelled, then Section 4.4 details an efficient inference algorithm using variational Bayes. Using simulated data, Section 4.5 assesses the performance of this method and its ability to detect changes. Section 4.6 tests DynIBCC on the Galaxy Zoo Supernovae (GZSN) application, while Section 4.7 examines some of the agent behaviour extracted from the GZSN dataset. The final sections, 4.8 and 4.9, explore the dynamics of communities of agents.

4.1 Dynamic Independent Bayesian Classifier Combination

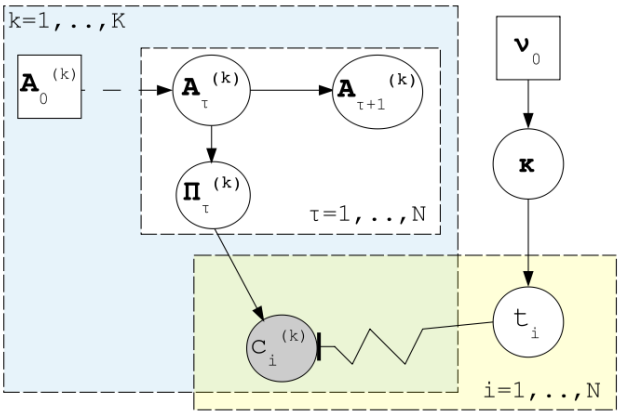


Figure 4.1: Graphical model for DynIBCC. Dashed arrows indicate dependencies on nodes at other time-steps. The zig-zag line means t_i is a switch that selects parameters from $\Pi_\tau^{(k)}$. The shaded node represents observed values. Circular, hollow nodes are variables with a distribution. Square nodes are variables instantiated with point values.

Dynamic Independent Bayesian Classifier Combination (DynIBCC) uses a similar model to standard IBCC, where the behaviour of each agent k is described by a confusion matrix $\boldsymbol{\pi}^{(k)}$. With DynIBCC, however, the confusion matrices may take different values at each discrete *time-step*. Theoretically, a time-step corresponds to a period in which we assume the agent's behaviour does not change. This chapter assumes that each response from an agent corresponds to a different time-step, so that we make exactly one observation of the agent's behaviour at each time-step. Figure 4.1 shows the resulting graphical model for DynIBCC, which will be explained below, starting with a recap of parts carried over from static IBCC.

The right-hand side of the model relates to the target labels and is the same as the standard, static variant of IBCC. For object $i \in \{1, .., N\}$ we assume its target label $t_i \in \{1, .., J\}$ is generated by a categorical distribution with parameters $\boldsymbol{\kappa}$. The parameters $\boldsymbol{\kappa}$ represent the proportions of each target value and have a Dirichlet prior distribution with hyperparameters $\boldsymbol{\nu}_0$.

As with the static variant of IBCC, an agent $k \in \{1, .., K\}$ produces a discrete, cate-

gorical response $c_i^{(k)} \in \{1, .., L\}$ for object i. This is shown on the left-hand side of the graphical model. As before, we assume that $c_i^{(k)}$ is generated by a categorical distribution. The categorical distribution is selected by the value of t_i from J distributions, where J is the number of target values. With DynIBCC, however, the distribution over $c_i^{(k)}$ has a parameter vector $\boldsymbol{\pi}_{\tau,t_i}^{(k)}$, which applies only to the time-step τ at which k produced response $c_i^{(k)}$. The parameter vectors at one time-step are rows in a confusion matrix, $\boldsymbol{\Pi}_\tau^{(k)}$:

$$\boldsymbol{\Pi}_\tau^{(k)} = \begin{bmatrix} \boldsymbol{\pi}_{\tau,j}^{(k)} \\ \vdots \\ \boldsymbol{\pi}_{\tau,j}^{(k)} \end{bmatrix}. \tag{4.1}$$

This confusion matrix characterises the behaviour of k in the same manner as the time-independent confusion matrices in the original variant of IBCC. The parameters $\boldsymbol{\pi}_{\tau,t_i}^{(k)}$ are drawn from a Dirichlet distribution with hyperparameters $\boldsymbol{\alpha}_{\tau,t_i}^{(k)}$. The hyperparameters evolve over time according to a *Markov process*, so that the hyperparameters $\boldsymbol{\alpha}_{\tau,t_i}^{(k)}$ at time-step τ depend only on the hyperparameters $\boldsymbol{\alpha}_{\tau-1,t_i}^{(k)}$ at the previous time-step $\tau - 1$. These dynamic changes are described in detail in the following section, and pseudo-code for a corresponding inference algorithm is given in Appendix B.

Agents may analyse objects in different orders, so for each agent $k \in 1, .., K$ we use a function s to map objects to the time-steps at which the agents responded to them. Thus, agent k produces a response for object i at time-step $\tau = s(i, k)$.

The joint distribution for the DynIBCC model is:

$$\begin{aligned} p(\boldsymbol{t}, \boldsymbol{c}, \boldsymbol{\kappa}, \underline{\boldsymbol{\Pi}} | \boldsymbol{A}_0, \boldsymbol{\nu}_0) = & \prod_{i=1}^{N} \left\{ \kappa_{t_i} \prod_{k=1}^{K} \pi_{s(i,k),t_i,c_i^{(k)}}^{(k)} \right\} p(\boldsymbol{\kappa}|\boldsymbol{\nu}_0) \\ & \prod_{k=1}^{K} \prod_{\tau=1}^{N} \prod_{j=1}^{J} \left\{ p\left(\boldsymbol{\pi}_{\tau,j}^{(k)} | \boldsymbol{\alpha}_{\tau,j}^{(k)}\right) p\left(\boldsymbol{\alpha}_{\tau,j}^{(k)} | \boldsymbol{\alpha}_{\tau-1,j}^{(k)}\right) \right\}, \end{aligned} \tag{4.2}$$

where $\underline{\boldsymbol{\Pi}} = \left\{ \boldsymbol{\Pi}_\tau^{(k)} | k = 1, .., K, \tau = 1, .., N^{(k)} \right\}$ is the set of all sequences of confusion matrices for all agents k, over the total number of time-steps $N^{(k)}$, and $\boldsymbol{A}_0 =$

$\left\{ \boldsymbol{A}_0^{(k)} | k = 1, .., K \right\}$ is the set of prior hyperparameters for all confusion matrices, where $\boldsymbol{A}_0^{(k)}$ is defined by:

$$\boldsymbol{A}_0^{(k)} = \begin{bmatrix} \boldsymbol{\alpha}_{0,1}^{(k)} \\ \vdots \\ \boldsymbol{\alpha}_{0,J}^{(k)} \end{bmatrix}. \tag{4.3}$$

In practice, the agent may not have responded to all objects. In cases where the agent's response is unavailable, the corresponding terms involving $\pi_{\tau,j,l}^{(k)}$ should simply be omitted.

The following two sections provide further information about DynIBCC. Section 4.2 describes the dynamic process that relates the hyperparameters $\boldsymbol{\alpha}_{\tau,j}^{(k)}$ to those of the previous time-step, $\boldsymbol{\alpha}_{\tau-1,j}^{(k)}$. Then, section 4.4 details a variational Bayes inference method for inferring the unknown parameters of DynIBCC. This algorithm is also described using pseudo-code in Appendix B.

4.2 Choosing a Dynamic Model for Confusion Matrices

DynIBCC requires a dynamic model for the confusion matrices $\boldsymbol{\Pi}_\tau^{(k)}$ that allows us to infer their distribution at each discrete time-step. Since we are working in a Bayesian framework, the dynamic model must describe how the hyperparameters $\boldsymbol{A}_\tau^{(k)}$ relate to those of the previous time-step, $\boldsymbol{A}_{\tau-1}^{(k)}$. The confusion matrices and their hyperparameters therefore relate to the latent state of a system that evolves according to a Markov Process, since the values of hyperparameters depend only on those of the previous time-step. This idea forms the underlying model of a class of algorithms for estimating probability distributions known as *Bayes filters* or *recursive Bayesian estimation*. Bayes filters assume that the distribution of an observed variable depends on the system's state. Different Bayes filters assume different observation and state models, in which the state may evolve according to either linear or non-linear operators.

A widely-used Bayes filter is the *Kalman filter*, in which observations and state transitions are generated by applying a linear function to the true state, possibly incorporating

additional input data. The distributions over all variables are assumed to be Gaussian, including white noise added at each state transition and noise added to observations of the true state. The Kalman Filter provides a computationally efficient algorithm that produces optimal estimates when the model is correct. However, its assumptions limit the range of scenarios where it can be applied. The *Extended Kalman Filter (EKF)* was developed to allow for non-linear dynamics, where the observation model and state transition models can be non-linear functions of the state [Jazwinski, 1970]. Unfortunately, a non-linear model leads to non-analytic updates of the state and observation distributions, so the EKF approximates the non-linear functions using the first two terms of their Taylor series expansion. It is possible to use the EKF to predict binary variables given a set of input data [Lowne et al., 2010], so it could be applicable to the issue of predicting discrete responses from agents.

An alternative to the EKF is the unscented Kalman filter (UKF). The UKF approximates the distribution by applying the non-linear transformation to a deterministic sample of points in the distribution [Julier and Uhlmann, 1997; Wan and Van Der Merwe, 2000]. The UKF has been shown to outperform the EKF in many cases [Julier and Uhlmann, 1997]. It is also possible to use a UKF to predict discrete variables [Lee and Roberts, 2010].

However, while the EKF and UKF are established methods for non-linear dynamics, the use of approximate non-linear transformations can introduce significant errors and invalidates the theoretical optimality of the Kalman filter [Haupt et al., 1996]. When using the models to perform inference, both methods require that we work with an approximate Gaussian distribution rather than the natural conjugate priors for our observed variables. Thus for DynIBCC, we could no longer update posteriors over the confusion matrices simply by updating the hyperparameters $\boldsymbol{A}_{\tau}^{(k)}$ of the conjugate Dirichlet distributions. As mentioned at the start of this chapter, the distributions over confusion matrices provide important information for weak control of a multi-agent system, including intelligent task assignment. Therefore, DynIBCC employs a different but related Bayes filter, the *Dynamic Generalised Linear Model (DGLM)*, which can directly model the evolution of observed

variables with any exponential-family distribution, which includes the categorical distribution over agents responses. The trade-off is that the DGLM specifies linear state transitions. The next section describes the DGLM in more depth, while the following sections present the algorithm for estimating posterior distributions in a DGLM.

4.3 Dynamic Generalised Linear Model for DynIBCC

In the case of DynIBCC, an agent's decision when presented with an object is an *observation variable* $c_i^{(k)}$. The probability of each response conditioned on the target value is given by the confusion matrix $\boldsymbol{\Pi}_\tau^{(k)}$, which we wish to infer. The core idea of the DGLM is that we can apply a non-linear *link function* to $\boldsymbol{\Pi}_\tau^{(k)}$ to obtain a *generalised linear model*. The changing distribution over $\boldsymbol{\Pi}_\tau^{(k)}$ can then be tracked through linear state transitions, which are applied to the first two moments of the distribution over $\boldsymbol{\Pi}_\tau^{(k)}$.

The use of a DGLM for tracking a binomial variable was developed in [Lee and Roberts, 2010] for use in a dynamic binary classification problem where classifications are usually estimated online by a filtering algorithm. The DGLM is applied in a novel way here, since the aim is to reliably estimate the probability of an agent's response given only a distribution over the target label, rather than the standard scenario in which a binary class label is predicted from observations of a number of input variables. The algorithm is extended to include a smoothing procedure, which updates the posterior distributions over $\boldsymbol{\Pi}^{(k)}$ at earlier time-steps after receiving subsequent decisions.

4.3.1 Linear Models

Linear models are used for regression to relate an observation variable y to the state $\boldsymbol{w}$ and an input variable $\boldsymbol{h}$. A linear model has the form

$$y = \boldsymbol{h}^{\mathrm{T}}\boldsymbol{w} + n, \tag{4.4}$$

where n is Gaussian noise with zero mean. In this standard case, the observation variable y has a multivariate Gaussian distribution with its mean given by

$$\mathbb{E}[y] = \boldsymbol{h}^{\mathrm{T}}\boldsymbol{w}. \tag{4.5}$$

4.3.2 Generalised Linear Models

The *generalised* linear model allows the observation variable y to take any exponential-family distribution, which includes the multinomial and Gaussian distributions. Beside the more common expressions used to write the probability distribution functions, exponential-family distributions can all be expressed in a canonical form with two parameters, θ and ϕ:

$$p(y|\theta, \phi) = \exp\left(\frac{d(y)\theta - b(\theta)}{a(\phi)} + c(y, \theta)\right). \tag{4.6}$$

The parameter θ is called the *canonical parameter*. The functions a, b, c and d are specified by the particular type of distribution.

In a generalised linear model, the mean $\mathbb{E}[y]$ is related to $\boldsymbol{h}$ and $\boldsymbol{w}$ by a link function g:

$$g(\mathbb{E}[y]) = \eta = \boldsymbol{h}^{\mathrm{T}}\boldsymbol{w}. \tag{4.7}$$

For convenience, we refer to the term $\eta = \boldsymbol{h}^{\mathrm{T}}\boldsymbol{w}$ as the *linear predictor*. Typically, the link function is chosen to be the *canonical link*, which means that the linear predictor η is equal to the canonical parameter θ of the distribution over y:

$$g(\mathbb{E}[y]) = \eta = \theta. \tag{4.8}$$

The state $\boldsymbol{w}$ is an unknown parameter, which is inferred from observations of y. Choosing the canonical link means that given a set of observations $\boldsymbol{Y}$, and inputs $\boldsymbol{H}$, $\boldsymbol{HY}$ is a minimal sufficient statistic for $\boldsymbol{w}$. This means that $\boldsymbol{HY}$ provides all the information about the distribution over $\boldsymbol{w}$ that is held in the data $\boldsymbol{Y}$ and $\boldsymbol{H}$. This makes it easier to update

the distribution over $\boldsymbol{w}$ on receiving new observations because we only need to perform calculations over a single value, $\boldsymbol{HY}$, rather than the original dataset.

4.3.3 Generalised Linear Model of Agent Responses

Let us derive the generalised linear model for the responses from agent k, first considering a static model. First, consider the variables $\boldsymbol{h}$, y and $\boldsymbol{w}$ in Equation 4.7 and how they relate to the DynIBCC model. The target label t_i can be converted to the input vector $\boldsymbol{h}$. Since t_i may be uncertain, each element is set so that $h_j = p(t_i = j)$. The observation variable y corresponds to the agent's response $c_i^{(k)} = l$, which takes value $l \in \{1, .., L\}$. We represent this as a vector $\boldsymbol{y}$ in which one element $y_l = 1$ and all other elements are zero. The generalised linear model predicts the expected value of $\boldsymbol{y}$ given the input vector $\boldsymbol{h}$. We write this prediction as $\tilde{\boldsymbol{\pi}}^{(k)} = \mathbb{E}[\boldsymbol{y}]$. Updating Equation 4.7 to use the vector form for $\boldsymbol{y}$, we obtain the generalised linear model:

$$g(\mathbb{E}[\boldsymbol{y}]) = g(\tilde{\boldsymbol{\pi}}^{(k)}) = \boldsymbol{\eta} = \boldsymbol{h}^{\mathrm{T}}\boldsymbol{W} \tag{4.9}$$

where $\boldsymbol{W}^{(k)}$ is a state matrix. We wish to choose the link function $g(.)$ to be the canonical link so that $\boldsymbol{\theta} = g(\tilde{\boldsymbol{\pi}}^{(k)})$. Since $\boldsymbol{y}$ has a categorical distribution, the canonical parameter $\boldsymbol{\theta}$ is a vector

$$g(\tilde{\boldsymbol{\pi}}^{(k)}) = \boldsymbol{\theta}^{(k)} = [\ln \tilde{\pi}_1^{(k)}; ...; \ln \tilde{\pi}_L^{(k)}]. \tag{4.10}$$

Therefore,

$$\tilde{\boldsymbol{\pi}}^{(k)} = \exp(\boldsymbol{\eta}) = \exp(\boldsymbol{h}^{\mathrm{T}}\boldsymbol{W}). \tag{4.11}$$

Now we relate this back to the DynIBCC model. If the value of t_i is known, then $\tilde{\boldsymbol{\pi}}_\tau$ is equal to the corresponding row of the confusion matrix $\boldsymbol{\pi}_{\tau,t_i}$. However, if t_i is uncertain, the prediction $\tilde{\boldsymbol{\pi}}_\tau$ integrates over this uncertainty, producing a mixture of rows of the confusion matrix, where each row $\boldsymbol{\pi}_{\tau,j}$ is weighted by the probability $t_i = j$.

The generalised linear model for IBCC can also be seen as L independent models, where each element of $\boldsymbol{\theta}^{(k)}$ has a corresponding state vector $\boldsymbol{w}_l^{(k)}$. The state vector $\boldsymbol{w}_l^{(k)}$

refers to a column of the matrix $\boldsymbol{W}^{(k)}$:

$$\boldsymbol{W}^{(k)} = [\boldsymbol{w}_1^{(k)}, ..., \boldsymbol{w}_L^{(k)}]. \tag{4.12}$$

In some calculations the notation requires that we refer to the state vectors separately. The remainder of this section omits the superscripts $^{(k)}$ for clarity, assuming we are working with the confusion matrices of one agent only.

4.3.4 Introducing Dynamics to the Generalised Linear Model

Now let us introduce state transitions to produce a dynamic generalised linear model. Each state vector $\boldsymbol{w}_l$ corresponding to an agent's response l evolves according to a random walk. At time-step τ, the state vector $\boldsymbol{w}_l$ transitions according to

$$\boldsymbol{w}_{\tau,l} = \boldsymbol{w}_{\tau-1,l} + \boldsymbol{v}_{\tau,l}, \tag{4.13}$$

where $\boldsymbol{v}_{\tau,l}$ is the state noise vector. Since we are working in a Bayesian setting, where there is a distribution over the confusion matrices $\boldsymbol{\Pi}_\tau$, there must also be a distribution over the state $\boldsymbol{w}$ and the state noise $\boldsymbol{v}_\tau$. Here, we introduce an approximation since we do not assume a particular distribution for these variables; instead we specify only the first two moments:

$$\boldsymbol{w}_{\tau,l} \sim \hat{\boldsymbol{w}}_{\tau,l}, \boldsymbol{P}_{\tau,l} \tag{4.14}$$

$$\boldsymbol{v}_{\tau,l} \sim \boldsymbol{0}, q_{\tau,l}\boldsymbol{I}, \tag{4.15}$$

where $\hat{\boldsymbol{w}}_{\tau,l}$ is the state mean for decision l at time τ, $\boldsymbol{P}_{\tau,l}$ is the corresponding state covariance, $q_{\tau,l}$ is the state noise variance, and $\boldsymbol{I}$ is the identity matrix. We do not specify the complete distributional form for $\boldsymbol{w}_{\tau,l}$ because this allows estimates of state transitions in Equation 4.13 to remain tractable.

The generalised linear model allows us to relate the state moments to moments of the

distribution over the full confusion matrix $\boldsymbol{\Pi}_\tau$. In the dynamic generalised linear model, we introduce time indexes τ to all variables in the generalised linear model. Equation 4.11 now produces a prediction $\tilde{\boldsymbol{\pi}}_\tau$ given a specific input vector $\boldsymbol{h}_\tau$. In the DynIBCC model, confusion matrices $\boldsymbol{\Pi}_\tau$ have a row for each possible target value. If we remove the effect of the input vector $\boldsymbol{h}_\tau$ that represents the target label, the moments of the confusion matrices $\boldsymbol{\Pi}_\tau$ map directly to the state moments:

$$\mathbb{E}[g(\boldsymbol{\Pi}_\tau)] = \mathbb{E}[\ln(\boldsymbol{\Pi}_\tau)] = \hat{\boldsymbol{w}} \tag{4.16}$$

$$\mathbb{V}[g(\boldsymbol{\Pi}_\tau)] = \mathbb{V}[\ln(\boldsymbol{\Pi}_\tau)] = \begin{bmatrix} P_{11} \\ P_{22} \\ \vdots \\ P_{JJ} \end{bmatrix}. \tag{4.17}$$

This completes the underlying model of the DGLM. The aim is to use this model to learn a full posterior distribution $p(\boldsymbol{\Pi}_1, .., \boldsymbol{\Pi}_N | c_1, .., c_N)$ over a sequence of confusion matrices $\boldsymbol{\Pi}_1, ..., \boldsymbol{\Pi}_N$ given a set of observations $c_1, .., c_N$. The method for estimating the distribution consists of two sequential passes. The first is a forward pass in which we scroll through the data starting from $\tau = 1$, estimating the distribution at each time-step τ given all observations up to time τ. The forward pass is known as *filtering* and gives us $p(\boldsymbol{\Pi}_\tau | \boldsymbol{c}_\tau)$, where $\boldsymbol{c}_\tau$ is the set of responses up to time τ, for all values of $\tau \in 1, ..., N$. However, when data has been observed up to time N, we need to use a backward pass to obtain the full posterior, $p(\boldsymbol{\Pi}_\tau | \boldsymbol{c}_N)$, where $\boldsymbol{c}_N$ is the set of all responses from this agent. This backward pass is referred to as *smoothing*. The next two subsections detail the filtering and smoothing calculations.

4.3.5 Filtering

This subsection derives the calculations for each step of the filtering algorithm, which iteratively calculates $p(\boldsymbol{\Pi}_\tau | \boldsymbol{c}_\tau)$ at each time-step $\tau = 1, .., N$, given a set of responses $\boldsymbol{c}_\tau$ up to time τ.

Step One: Calculate State Priors

In each iteration of the forward pass, we first estimate the mean and covariance of $\boldsymbol{w}_{\tau,l}$, for each response l, given previous observations up to time-step $\tau - 1$. We refer to these moments as the prior state mean, $\hat{\boldsymbol{w}}_{\tau|\tau-1,l}$ and prior state covariance, $\boldsymbol{P}_{\tau|\tau-1,l}$.

In the first iteration $\tau = 1$, there are no previous observations, but the hyperparameters $\boldsymbol{A}_0$ specify a Dirichlet prior over $\boldsymbol{\Pi}_1$. Using the link function in Equation 4.10, we can approximate the prior state mean using moments of the Dirichlet prior:

$$\hat{w}_{1|0,j,l} = \Psi(\alpha_{0,j,l}) - \Psi(\sum_{\lambda=1}^{L} \alpha_{0,j,\lambda}) \qquad \approx \ln\left(\frac{\alpha_{0,j,l}}{\sum_{\lambda=1}^{L} \alpha_{0,j,\lambda}}\right) \tag{4.18}$$

$$P_{1|0,j,j,l} = \Psi'(\alpha_{0,j,l}) + \Psi'(\sum_{\lambda=1}^{L} \alpha_{0,j,\lambda}) \qquad \approx \frac{1}{\alpha_{0,j,l}} + \frac{1}{\sum_{\lambda=1}^{L} \alpha_{0,j,\lambda}}, \tag{4.19}$$

where $\Psi()$ is the standard digamma function and $\Psi'()$ is its first derivative [Davis, 1965]. The off-diagonal elements of $\boldsymbol{P}_{1|0}$ are set to zero.

At time-steps where $\tau > 1$, we estimate the prior state moments by adding noise to the moments from the previous time-step:

$$\hat{\boldsymbol{w}}_{\tau|\tau-1,l} = \hat{\boldsymbol{w}}_{\tau-1|\tau-1,l} \tag{4.20}$$

$$\boldsymbol{P}_{\tau|\tau-1,l} = \boldsymbol{P}_{\tau-1|\tau-1,l} + q_{\tau-1,l}\boldsymbol{I}, \tag{4.21}$$

where $\hat{\boldsymbol{w}}_{\tau-1|\tau-1,l}$ and $\boldsymbol{P}_{\tau-1|\tau-1,l}$ are the posterior state mean and covariance from the previous time-step, and $q_{\tau-1,l}$ is the state noise variance. The noise variance plays an important role in the speed of transition allowed between states. A high value will effectively dilute the information from previous iterations. Step Five shows how to estimate $q_{\tau-1,l}$.

Step Two: Translate Current State Distribution to Dirichlet Prior

We refer to the prediction as

$$\tilde{\boldsymbol{\pi}}_\tau = p(c_i|\boldsymbol{h}_\tau, \boldsymbol{W}_\tau). \tag{4.22}$$

Since the state $\boldsymbol{W}_\tau$ is uncertain, we have a distribution over the prediction $\tilde{\boldsymbol{\pi}}_\tau$. Prediction $\tilde{\boldsymbol{\pi}}_\tau$ takes a Dirichlet distribution since it is the conjugate prior to the categorical distribution. When we observe c_i at time τ, we wish to update the conjugate prior directly, so that the observation can be incorporated in a fully Bayesian manner. To do this, we must convert the prior state mean and covariance to Dirichlet hyperparameters $\tilde{\boldsymbol{\alpha}}_\tau$.

The distribution over $\tilde{\boldsymbol{\pi}}_\tau$ given the prior state moments is

$$\begin{aligned} p(\tilde{\boldsymbol{\pi}}_\tau|\hat{\boldsymbol{W}}_{\tau|\tau-1}, \boldsymbol{P}_{\tau|\tau-1,1}, .., \boldsymbol{P}_{\tau|\tau-1,L}, \boldsymbol{h}_\tau) &= \mathrm{Dir}(\tilde{\boldsymbol{\pi}}_\tau|\tilde{\boldsymbol{\alpha}}_{\tau|\tau-1}) \\ &= \mathrm{Dir}(\exp \boldsymbol{\eta}_\tau|\tilde{\boldsymbol{\alpha}}_{\tau|\tau-1}) \\ &= \frac{1}{\mathrm{B}(\tilde{\boldsymbol{\alpha}}_{\tau|\tau-1})} \prod_{l=1}^{L} \exp(\eta_{\tau,l})^{\tilde{\boldsymbol{\alpha}}_{\tau|\tau-1,l}} \end{aligned} \tag{4.23}$$

where $\mathrm{Dir}()$ symbolises a Dirichlet distribution and $\mathrm{B}()$ is the multinomial beta function [Davis, 1965]. Here, we have re-introduced the linear predictor, $\boldsymbol{\eta_\tau} = \boldsymbol{h}^{\mathrm{T}}\boldsymbol{W}_\tau$, which we use to translate the state mean and covariance into the hyperparameters $\tilde{\boldsymbol{\alpha}}_{\tau|\tau-1}$ as follows. First, we calculate the elements of the prior mean of the linear predictor η_τ:

$$\begin{aligned} \mathbb{E}[\ln \tilde{\boldsymbol{\pi}}_\tau] = \hat{\eta}_{\tau|\tau-1,l} &= \mathbb{E}[\eta_{\tau,l}|\boldsymbol{c}_{\tau-1}, p(\boldsymbol{t}_\tau)] \\ &= \boldsymbol{h}_\tau^{\mathrm{T}} \hat{\boldsymbol{w}}_{\tau|\tau-1,l} \\ &= \Psi(\tilde{\alpha}_{\tau|\tau-1,l}) - \Psi(\sum_{l=1}^{L} \tilde{\alpha}_{\tau|\tau-1,l}) \\ &\approx \ln\left(\frac{\tilde{\alpha}_{\tau|\tau-1,l}}{\sum_{l=1}^{L} \tilde{\alpha}_{\tau|\tau-1,l}}\right), \end{aligned} \tag{4.24}$$

where $\boldsymbol{c}_{\tau-1}$ is the set of observed agent responses up to time $\tau - 1$, and $\boldsymbol{t}_\tau$ are target labels up to time τ. The approximation here can be used to save computational cost. Next, the

prior variance of the linear predictor is given by:

$$\begin{aligned}\mathbb{V}[\ln \tilde{\boldsymbol{\pi}}_\tau] = r_{\tau|\tau-1,l} &= \mathbb{V}[\eta_{\tau,l}|\boldsymbol{c}_{\tau-1}, p(\boldsymbol{t}_\tau)] \\ &= \boldsymbol{h}_\tau^{\mathrm{T}} \boldsymbol{P}_{\tau|\tau-1,l} \boldsymbol{h}_\tau \\ &= \Psi'(\tilde{\alpha}_{\tau|\tau-1,l}) + \Psi'(\sum_{l=1}^{L} \tilde{\alpha}_{\tau|\tau-1,l}) \\ &\approx \frac{1}{\tilde{\alpha}_{\tau|\tau-1,l}} + \frac{1}{\sum_{l=1}^{L} \tilde{\alpha}_{\tau|\tau-1,l}} \end{aligned} \tag{4.25}$$

Rearranging these equations, we obtain the prior hyperparameters $\tilde{\boldsymbol{\alpha}}_{\tau|\tau-1}$ in terms of the prior mean and variance of the linear predictor:

$$\tilde{\alpha}_{\tau|\tau-1,l} \approx \frac{1 + \exp(\hat{\eta}_{\tau|\tau-1,l})}{r_{\tau|\tau-1,l}}. \tag{4.26}$$

Step Three: Update Given Current Observation

This step updates the conjugate prior over $\tilde{\boldsymbol{\pi}}_\tau$ at time τ to the posterior given the current observation, c_i, which is the response to the object i at time τ. The observation c_i at time τ, can be represented as a binary vector $\boldsymbol{y}_\tau$, in which all elements are zero except $y_l = 1$, where $c_i = l$. Using this representation, we calculate posterior hyperparameters $\tilde{\boldsymbol{\alpha}}_{\tau|\tau,l}$ by adding to the prior hyperparameters:

$$\tilde{\boldsymbol{\alpha}}_{\tau|\tau} = \tilde{\boldsymbol{\alpha}}_{\tau|\tau-1} + \boldsymbol{y}_\tau. \tag{4.27}$$

In this way we update the pseudo-counts of agent responses as we did in the static IBCC model in Equation (3.19).

Step Four: Update the State Mean and Covariance

Step Four converts the hyperparameters $\tilde{\boldsymbol{\alpha}}_{\tau|\tau}$ back to the posterior mean and covariance of $\boldsymbol{W}_\tau$. Given the updated hyperparameters, the posterior moments of the linear predictor are

approximated by

$$\hat{\eta}_{\tau|\tau,l} \approx \ln\left(\frac{\tilde{\alpha}_{\tau|\tau,l}}{\sum_{l=1}^{L}\tilde{\alpha}_{\tau|\tau,l}}\right) \tag{4.28}$$

$$r_{\tau|\tau,l} \approx \frac{1}{\tilde{\alpha}_{\tau|\tau,l}} + \frac{1}{\sum_{l=1}^{L}\tilde{\alpha}_{\tau|\tau,l} - \tilde{\alpha}_{\tau|\tau,l}}. \tag{4.29}$$

Then, we can apply an update to the mean and covariance of the state variable using linear Bayesian estimation, described in [West and Harrison, 1997]:

$$\hat{\boldsymbol{w}}_{\tau|\tau,l} = \hat{\boldsymbol{w}}_{\tau|\tau-1,l} + \boldsymbol{K}_{\tau,l}\left(\hat{\eta}_{\tau|\tau,l} - \hat{\eta}_{\tau|\tau-1,l}\right) \tag{4.30}$$

$$\boldsymbol{P}_{\tau|\tau,l} = \left(\boldsymbol{I} - \boldsymbol{K}_{\tau,l}\boldsymbol{h}_{\tau}^{\mathrm{T}}\left(1 - \frac{r_{\tau|\tau,l}}{r_{\tau|\tau-1,l}}\right)\right)\boldsymbol{P}_{\tau|\tau-1,l}, \tag{4.31}$$

where $\boldsymbol{K}_{\tau,l}$, the equivalent of the optimal Kalman gain, is

$$\boldsymbol{K}_{\tau,l} = \frac{\boldsymbol{P}_{\tau|\tau-1,l}^{\mathrm{T}}\boldsymbol{h}_{\tau}}{r_{\tau|\tau-1,l}}. \tag{4.32}$$

The term $\frac{r_{\tau|\tau,l}}{r_{\tau|\tau-1,l}}$ in the covariance update corresponds to our uncertainty over $\eta_{\tau,l}$, which we do not observe directly. The posterior state moments derived here for time-step τ are used in Step One to calculate the prior moments for the subsequent time-step $\tau + 1$.

Linear Bayes estimation gives an optimal estimate when the full distribution over the state variable $\boldsymbol{w}_{\tau,l}$ is unknown, and therefore differs theoretically from a Kalman filter in not specifying a Gaussian distribution over $v_{\tau,l}$ in Equation (4.13). However, these update equations are otherwise very similar to those of the Kalman Filter.

Step Five: Estimate State Noise Variance

In Bayes filters, the state noise variance $q_{\tau,l}$ can be estimated according to several different principles: choosing the value that maximises the evidence of observations [Jazwinski, 1969]; choosing $q_{\tau,l}$ to maximise the model evidence [Penny and Roberts, 1999]; setting $q_{\tau,l}$ so that the prior and posterior distributions over the observation variable $\boldsymbol{y}$ have the same variance [Lowne et al., 2010]. The first method sets $q_{\tau,l}$ so that the current observation

is maximally probable, therefore moving the state toward the current observation. In the case where we observe only one discrete label, y_τ, at each time-step, τ, this may be less appropriate, since a single decision contains only a small amount of information about the probability of different decision at future time-steps. Instead, [Penny and Roberts, 1999] propose retrospectively maximising the prior evidence of the posterior prediction $\boldsymbol{\eta}_{\tau|\tau}$. A value of $q_{\tau,l}$ is chosen so that when added to the prior covariance, $P_{\tau|\tau-1,l}$, as in Equation 4.21, the value of $p(\boldsymbol{\eta}_{\tau|\tau}|\hat{\boldsymbol{w}}_{\tau|\tau-1}, P_{\tau|\tau-1,l} + q_{\tau,l})$ is maximised. This is equivalent to variance matching, where we set $q_{\tau,l}$ so that the variance over $\boldsymbol{y}$ given prior state mean $\hat{\boldsymbol{w}}_{\tau|\tau-1}$ and an adjusted prior covariance $P_{\tau|\tau-1,l} + q_{\tau,l}\boldsymbol{I}$ is equal to the variance over $\boldsymbol{y}$ given posterior state mean $\hat{\boldsymbol{w}}_{\tau|\tau}$ and posterior state covariance $P_{\tau|\tau,l}$. The model evidence approach of [Penny and Roberts, 1999] requires an expensive line search step, while taking the variance matching viewpoint results in the simpler method proposed by [Lowne et al., 2010], which is applied here to DynIBCC. Here, we estimate the state noise variance $q_{\tau,l}$ as

$$q_{\tau,l} = \max[u_{\tau|\tau,l} - u_{\tau|\tau-1,l}, 0] + z_{\tau,l} \tag{4.33}$$

where $u_{\tau|\tau,l}$ and are $u_{\tau|\tau-1,l}$ the variances in the distribution over the agent's decision c_i after observing data up to time τ and $\tau - 1$ respectively. The optional term $z_{\tau,l}$ is the uncertainty in the agents' decisions, which is zero when we have observed them directly. The uncertainty $u_{\tau|\upsilon,l}$ is defined for observations up to an arbitrary time-step υ:

$$u_{\tau|\upsilon,l} = \mathbb{V}[c_i|\boldsymbol{c}_\upsilon, \boldsymbol{t}, \boldsymbol{\alpha}_0] = \hat{\pi}_{\tau|\upsilon,l}\left(1 - \hat{\pi}_{\tau|\upsilon,l}\right), \tag{4.34}$$

where

$$\hat{\pi}_{\tau|\upsilon,l} = \mathbb{E}[\tilde{\pi}_{\tau,l}|\boldsymbol{c}_\upsilon, \boldsymbol{t}, \boldsymbol{\alpha}_0] \approx \exp\left(\boldsymbol{h}_\tau^{\mathrm{T}}\hat{\boldsymbol{w}}_{\tau|\upsilon,l}\right). \tag{4.35}$$

When the agent's decisions are not observed, we use $\hat{\pi}_{\tau|\tau-1,l}$ as an estimate of the missing output, so $z_{\tau,l} = u_{\tau|\tau-1,l}$. Equation 4.33 means that the prediction variance increases if and only if the prediction variance $u_{\tau|\tau,l}$ increased after observing $\boldsymbol{y}_\tau$. Thus, we increase the state noise variance only when we receive improbable observations that suggest a state

transition. This method of estimating the state noise variance therefore does not automatically include a forgetting factor based solely on the age of observations; rather the previous observations continue to inform the later predictions until a shift in behaviour is observed.

In Equation 4.21, the state noise variance is applied isotropically to all diagonal elements of the matrix $P_{\tau|\tau-1,l}$ through the term $q_{\tau,l}\boldsymbol{I}$, which is an approximation, since the terms corresponding to each target value may not change equally.

4.3.6 Smoothing

The backward pass smooths the posterior state moments at each time-step by incorporating observations from later time-steps. The algorithm proposed for the DynIBCC backward pass is the Modified Bryson-Frazier smoother [Bierman, 1973], which is chosen because each iteration is computationally inexpensive, avoiding the need to invert the covariance matrix, as required by alternatives such as the Rauch-Tung-Striebel smoother.

Having calculated the posterior state moments $\hat{\boldsymbol{w}}_{\tau|\tau,l}$ and $\boldsymbol{P}_{\tau|\tau,l}$ given data up to time τ, the backward pass finds the approximate posterior state moments given all subsequent data points, $\hat{\boldsymbol{w}}_{\tau|N,l}$ and $\boldsymbol{P}_{\tau|N,l}$. From these we can obtain the posterior hyperparameters given all data, $\tilde{\boldsymbol{\alpha}}_{\tau|N}$, and therefore the distribution over the confusion matrices.

The Modified Bryson-Frazier smoother [Bierman, 1973] updates the state mean and covariance by applying an adjoint mean vector $\hat{\boldsymbol{\lambda}}_{\tau,l}$ and adjoint covariance matrix $\hat{\boldsymbol{\Lambda}}_{\tau,l}$ as follows.

$$\hat{\boldsymbol{w}}_{\tau|N,l} = \hat{\boldsymbol{w}}_{\tau|\tau,l} - \boldsymbol{P}_{\tau|\tau,l}\hat{\boldsymbol{\lambda}}_{\tau,l} \tag{4.36}$$

$$\boldsymbol{P}_{\tau|N,l} = \boldsymbol{P}_{\tau|\tau,l} - \boldsymbol{P}_{\tau|\tau,l}\hat{\boldsymbol{\Lambda}}_{\tau,l}\boldsymbol{P}_{\tau|\tau,l}. \tag{4.37}$$

The adjoint mean and covariance represent the state change to the subsequent time-step $\tau+1$, which is moderated by the uncertainty $\boldsymbol{P}_{\tau|\tau,l}$ at the current time-step τ. $\hat{\boldsymbol{\lambda}}_{\tau,l}$ and $\hat{\boldsymbol{\Lambda}}_{\tau,l}$ are defined recursively as the posterior updates from the subsequent step $\tau+1$ given data

from $\tau + 1$ to N:

$$\hat{\boldsymbol{\lambda}}_{\tau,l} = \tilde{\boldsymbol{\lambda}}_{\tau+1,l} \qquad \hat{\boldsymbol{\lambda}}_N = \mathbf{0} \tag{4.38}$$

$$\hat{\boldsymbol{\Lambda}}_{\tau,l} = \tilde{\boldsymbol{\Lambda}}_{\tau+1,l} \qquad \hat{\boldsymbol{\Lambda}}_N = \mathbf{0}, \tag{4.39}$$

where we define

$$\tilde{\boldsymbol{\lambda}}_{\tau,l} = -\frac{\boldsymbol{h}_\tau}{r_{\tau|\tau-1,l}} \left(\hat{\eta}_{\tau|\tau,l} - \hat{\eta}_{\tau|\tau-1,l}\right) + \left(\boldsymbol{I} - \boldsymbol{K}_{\tau,l}\boldsymbol{h}_\tau^{\mathrm{T}}\right)^{\mathrm{T}} \hat{\boldsymbol{\lambda}}_{\tau,l} \tag{4.40}$$

$$\tilde{\boldsymbol{\Lambda}}_{\tau,l} = \frac{\boldsymbol{h}_\tau \boldsymbol{h}_\tau^{\mathrm{T}}}{r_{\tau|\tau-1,l}} \left(1 - \frac{r_{\tau|\tau,l}}{r_{\tau|\tau-1,l}}\right) + \left(\boldsymbol{I} - \boldsymbol{K}_{\tau,l}\boldsymbol{h}_\tau^{\mathrm{T}}\right)^{\mathrm{T}} \hat{\boldsymbol{\Lambda}}_{\tau,l} \left(\boldsymbol{I} - \boldsymbol{K}_{\tau,l}\boldsymbol{h}_\tau^{\mathrm{T}}\right). \tag{4.41}$$

The posterior updates $\tilde{\boldsymbol{\lambda}}_{\tau,l}$ and $\tilde{\boldsymbol{\Lambda}}_{\tau,l}$ are both sums of two terms: the first term is the update from the current time-step τ; the second term is the change passed back from subsequent time-steps. Using the smoothed state mean and covariance, the estimates for final posterior hyper-parameters are given by

$$\begin{aligned} \hat{\eta}_{\tau|N,l} &= \boldsymbol{h}_\tau^{\mathrm{T}} \hat{\boldsymbol{w}}_{\tau|N,l} \\ r_{\tau|N,l} &= \boldsymbol{h}_\tau^{\mathrm{T}} \boldsymbol{P}_{\tau|N,l} \boldsymbol{h}_\tau \\ \tilde{\alpha}_{\tau|N,l} &= \frac{1 + \exp(\hat{\eta}_{\tau|N,l})}{r_{\tau|N,l}}. \end{aligned} \tag{4.42}$$

This section explained a dynamic model for estimating state transitions over the agents' confusion matrices. An algorithm was presented for learning the posterior mean and covariance of an evolving latent state variable, $\boldsymbol{W}_\tau$. This approach enables us to work with conjugate prior distributions over $\tilde{\boldsymbol{\pi}}_\tau$, which is the probability of an agent's responses, given a particular distribution over the target labels $\boldsymbol{t}$. Computational tractability is retained by using approximations to update the hyperparameters at each step, and by using a scalable smoothing algorithm. The next section shows how to use the DGLM to infer the distribution over the set of complete confusion matrices $\underline{\boldsymbol{\Pi}}$ when the target labels $\boldsymbol{t}$ are unknown.

4.4 Variational Inference for DynIBCC

Usually, the target labels $\boldsymbol{t}$ are unknown, along with the confusion matrices $\underline{\boldsymbol{\Pi}}$ and target value proportions $\boldsymbol{\kappa}$. As with static IBCC algorithm, we propose a variational Bayes algorithm, *DynIBCC-VB*, to infer these unknown variables given a set of observed agent decisions, $\boldsymbol{c}$, and prior hyperparameters $\boldsymbol{A}_0$ and $\boldsymbol{\nu}_0$. The algorithm follows the same steps as the static variant described in Chapter 3, Section 3.2, but with the update equations referring to confusion matrices at particular time-steps.

In the static variant, the variational distribution for the confusion matrices, $q^*(\boldsymbol{\Pi})$ is given by Equation (3.16), and the corresponding hyperparameter updates in Equation 3.19. For DynIBCC-VB, $q^*(\boldsymbol{\Pi})$ becomes $q^*(\underline{\boldsymbol{\Pi}})$ for the set of confusion matrices at all time-steps:

$$q^*(\underline{\boldsymbol{\Pi}}) = \prod_{k=1}^{K}\prod_{j=1}^{J}\prod_{\tau=1}^{N} q^*\left(\boldsymbol{\pi}_{\tau,j}^{(k)}\right), \tag{4.43}$$

where $\underline{\boldsymbol{\Pi}}$ refers to the set of all confusion matrices for all agents, over all time-steps. The factor for each row is given by

$$\begin{aligned} q^*(\boldsymbol{\pi}_{\tau,j}^{(k)}) &= \frac{1}{B\left(\boldsymbol{\alpha}_{\tau|N,j}^{(k)}\right)} \prod_{l=1}^{L} \left(\pi_{\tau,j,l}^{(k)}\right)^{\alpha_{\tau|N,j,l}^{(k)}-1} \\ &= \mathrm{Dir}\left(\boldsymbol{\pi}_{\tau,j}^{(k)} | \alpha_{\tau|N,j,1}^{(k)}, .., \alpha_{\tau|N,j,l}^{(k)}\right) \end{aligned} \tag{4.44}$$

where $\mathrm{Dir}()$ is the Dirichlet distribution with parameters $\boldsymbol{\alpha}_{\tau|N,j}$ calculated according to

$$\alpha_{\tau|N,j,l} = \frac{1+\exp(\hat{w}_{\tau|N,j,l})}{P_{\tau|N,j,j,l}}, \tag{4.45}$$

which replaces the simpler static Equation 3.19. This equation is equivalent to Equation 4.42 with $h_{\tau,j} = 1$. The state mean $\hat{\boldsymbol{w}}_{\tau|N}$ and covariance $\boldsymbol{P}_{\tau|N}$ are calculated using the filtering and smoothing algorithms given the current distribution over target labels $\boldsymbol{t}$, which

is estimated with minor changes to the Equations 3.6:

$$\begin{aligned}\ln \rho_{i,j} &= \mathbb{E}_{\kappa_j, \underline{\mathbf{\Pi}}} \left[\ln p(\kappa_j, t_i, \underline{\mathbf{\Pi}}, \boldsymbol{c}) \right] \\ &= \mathbb{E}_{\boldsymbol{\kappa}}[\ln \kappa_j] + \sum_{k=1}^{K} \mathbb{E}_{\underline{\mathbf{\Pi}}} \left[\ln \pi^{(k)}_{\tau|N,j,c_i^{(k)}} | \tau = s(k,i) \right] + \text{const}, \end{aligned} \tag{4.46}$$

where $\tau = s(k,i)$ records the time-step at which agent k responded to object i. The distribution over a single target label is then identical to Equation 3.7:

$$q^*(t_i = j) = \mathbb{E}_{\boldsymbol{t}}[t_i = j] = \frac{\rho_{i,j}}{\sum_{\iota=1}^{J} \rho_{i,\iota}}. \tag{4.47}$$

In each iteration of the VB algorithm we must calculate Equation 4.45. This involves running the filtering and smoothing algorithms, using the current expectation over a target label as an estimate of the input data, so that $\tilde{h}_{\tau,j} = \mathbb{E}_{\boldsymbol{t}}[t_i = j]$, where $\tau = s(k,i)$.

The expectation over rows of the confusion matrices given by the static Equation (3.20) becomes

$$\mathbb{E}_{\underline{\mathbf{\Pi}}} \left[\ln \pi^{(k)}_{\tau|N,j,l} \right] = \Psi \left(\alpha^{(k)}_{\tau|N,j,l} \right) - \Psi \left(\sum_{m=1}^{L} \alpha^{(k)}_{\tau|N,j,m} \right). \tag{4.48}$$

This can then be used in the variational distribution over t_i in Equation 4.46. The complete algorithm follow the pseudo-code given in Listing B.1, except for the method "updateAlpha()", which is illustrated in pseudo-code in Listing B.2. This function runs the filtering and smoothing procedure at each iteration of the VB algorithm.

4.4.1 Variational Lower Bound

As explained in Chapter 3, the variational lower bound can be used to check for convergence of the VB algorithm. For DynIBCC this is given by

$$\begin{aligned}
L(q) &= \iiint q(\boldsymbol{t}, \underline{\boldsymbol{\Pi}}, \boldsymbol{\kappa}) \ln \frac{p(\boldsymbol{c}, \boldsymbol{t}, \underline{\boldsymbol{\Pi}}, \boldsymbol{\kappa} | \boldsymbol{A}_0, \boldsymbol{\nu}_0)}{q(\boldsymbol{t}, \underline{\boldsymbol{\Pi}}, \boldsymbol{\kappa})} \mathrm{d}\boldsymbol{t} \mathrm{d}\underline{\boldsymbol{\Pi}} \mathrm{d}\boldsymbol{\kappa} \\
&= \mathbb{E}_{\boldsymbol{t}, \underline{\boldsymbol{\Pi}}, \boldsymbol{\kappa}}[\ln p(\boldsymbol{c}, \boldsymbol{t}, \underline{\boldsymbol{\Pi}}, \boldsymbol{\kappa} | \boldsymbol{A}_0, \boldsymbol{\nu}_0)] - \mathbb{E}_{\boldsymbol{t}, \underline{\boldsymbol{\Pi}}, \boldsymbol{\kappa}}[\ln q(\boldsymbol{t}, \underline{\boldsymbol{\Pi}}, \boldsymbol{\kappa})] \\
&= \mathbb{E}_{\boldsymbol{t}, \underline{\boldsymbol{\Pi}}}[\ln p(\boldsymbol{c} | \boldsymbol{t}, \underline{\boldsymbol{\Pi}})] + \mathbb{E}_{\boldsymbol{t}, \underline{\boldsymbol{\Pi}}}[\ln p(\underline{\boldsymbol{\Pi}} | \boldsymbol{t}, \boldsymbol{A}_0)] + \mathbb{E}_{\boldsymbol{t}, \boldsymbol{\kappa}}[\ln p(\boldsymbol{t} | \boldsymbol{\kappa})] + \mathbb{E}_{\boldsymbol{t}_{\boldsymbol{\kappa}}}[\ln p(\boldsymbol{\kappa} | \boldsymbol{\nu}_0)] \\
&\quad - \mathbb{E}_{\boldsymbol{t}}[\ln q(\boldsymbol{t})] - \mathbb{E}_{\boldsymbol{\Pi}}[\ln q(\underline{\boldsymbol{\Pi}})] - \mathbb{E}_{\boldsymbol{\kappa}}[\ln q(\boldsymbol{\kappa})] \qquad (4.49)
\end{aligned}$$

The expectation terms are the same as for the static model in Subsection 3.2.4, with the following exceptions. We redefine the following positive terms relating to the joint probability of the latent variables, $\boldsymbol{t}$, the observed variables, $\boldsymbol{c}$, and the parameters $\underline{\boldsymbol{\Pi}}$ and $\boldsymbol{\kappa}$:

$$\mathbb{E}_{\boldsymbol{t}, \underline{\boldsymbol{\Pi}}}[\ln p(\boldsymbol{c} | \boldsymbol{t}, \underline{\boldsymbol{\Pi}})] = \sum_{i=1}^{N} \sum_{k=1}^{K} \sum_{j=1}^{J} \mathbb{E}_{t_i}[t_i = j] \mathbb{E}_{\underline{\boldsymbol{\Pi}}} \left[\ln \pi^{(k)}_{\tau, j, c_i^{(k)}} | \tau = s(k, i) \right] \qquad (4.50)$$

$$\mathbb{E}_{\underline{\boldsymbol{\Pi}}}[\ln p(\underline{\boldsymbol{\Pi}} | \boldsymbol{A}_0)] = \sum_{k=1}^{K} \sum_{j=1}^{J} \sum_{\tau=1}^{N} \left\{ - \ln \mathrm{B}\left(\boldsymbol{\alpha}^{(k)}_{0,j} \right) + \sum_{l=1}^{L} \left(\alpha^{(k)}_{0,j,l} - 1 \right) \mathbb{E}_{\underline{\boldsymbol{\Pi}}} \left[\ln \pi^{(k)}_{\tau | N, j, l} \right] \right\} \qquad (4.51)$$

In DynIBCC-VB, the expectation over the variational distribution $q^*(\underline{\boldsymbol{\Pi}})$ also differs from static IBCC:

$$\mathbb{E}_{\underline{\boldsymbol{\Pi}}}[\ln q(\underline{\boldsymbol{\Pi}})] = \sum_{k=1}^{K} \sum_{j=1}^{J} \sum_{\tau=1}^{N} \left\{ - \ln \mathrm{B}\left(\boldsymbol{\alpha}^{(k)}_{\tau | N, j} \right) + \sum_{l=1}^{L} \left(\alpha^{(k)}_{\tau | N, j, l} - 1 \right) \mathbb{E}_{\underline{\boldsymbol{\Pi}}} \left[\ln \pi^{(k)}_{\tau | N, j, l} \right] \right\}, \qquad (4.52)$$

where $\boldsymbol{\alpha}^{(k)}_{\tau | N, j}$ is given by Equation 4.45.

4.4.2 Duplicate and Missing Responses

The original static model did not allow for duplicate responses for the same object by the same agent. We assumed that even if an agent alters their decision when they see an object a second time, the two decisions are likely to be highly correlated and so cannot be treated as independent and only one decision should be accepted. However, the dynamic model reflects the possibility that the agent may change its own underlying model. Therefore, responses may be uncorrelated if they are separated by a sufficient change to the agents. A model that handles dependencies between duplicate classifications, first at time τ_{orig}, then at time τ_{dup}, may adjust the confusion matrices to compensate for correlation, for example by applying a moderating power:

$$\begin{aligned}\tilde{\boldsymbol{\pi}}^{(k)}_{\tau_{\text{orig}}} &= \left(\boldsymbol{\pi}^{(k)}_{\tau_{\text{orig}}}\right)^{1/\text{corr}(k,\tau_{\text{orig}},\tau_{\text{dup}})} \\ \tilde{\boldsymbol{\pi}}^{(k)}_{\tau_{\text{dup}}} &= \left(\boldsymbol{\pi}^{(k)}_{\tau_{\text{dup}}}\right)^{1/\text{corr}(k,\tau_{\text{orig}},\tau_{\text{dup}})},\end{aligned} \tag{4.53}$$

where $\text{corr}()$ is a function that measures correlation. However, the assumption of independence between agents is already inherent in DynIBCC and IBCC, yet has been shown to work in both synthetic cases (Chapter 2) and real-world situations (Chapter 3), where this assumption is unlikely to hold. Therefore, it may be possible to find a heuristic to determine when a duplicate decision can be counted as an independent observation, for example, when the two decisions are separated by a large number of time-steps. In cases where duplicates are allowed it is more convenient to index decisions by their time-step as $c^{(k)}_{\tau}$, rather than by the object index i. For model variants that permit duplicates the joint distribution is hence:

$$\begin{aligned}p(\boldsymbol{t}, \boldsymbol{c}, \boldsymbol{\kappa}, \underline{\boldsymbol{\Pi}}|\boldsymbol{A}_0, \boldsymbol{\nu}_0) = p(\boldsymbol{\kappa}|\boldsymbol{\nu}_0) \prod_{i=1}^{N} \kappa_{t_i} \\ \prod_{k=1}^{K} \prod_{\tau=1}^{N^{(k)}} \left\{ \pi^{(k)}_{\tau,T(k,\tau),c^{(k)}_{\tau}} \prod_{j=1}^{J} p\left(\boldsymbol{\pi}^{(k)}_{\tau,j}|\boldsymbol{\alpha}^{(k)}_{\tau,j}\right) p\left(\boldsymbol{\alpha}^{(k)}_{\tau,j}|\boldsymbol{\alpha}^{(k)}_{\tau-1,j}\right) \right\}\end{aligned} \tag{4.54}$$

where $T(k, \tau)$ maps the agent index k and the time-step τ back to the target label t_i for the corresponding object, and $N^{(k)}$ is the total number of decisions produced by agent k. We must also update Equation (4.47) to allow duplicates:

$$\ln q^*(t_i = j) = \mathbb{E}[\ln \kappa_j] + \sum_{k=1}^{K} \mathbb{E}\left[\ln \pi^{(k)}_{\tau,j,c^{(k)}_\tau} | \tau = s(k,i)\right] + \text{const}, \tag{4.55}$$

where $\tau = s(k, i)$ maps the object index i to the time-step τ at which agent k responded to i. Further research is required to handle duplicates in a principled manner, and would likely coincide with research into explicit handling of correlations between agents in general.

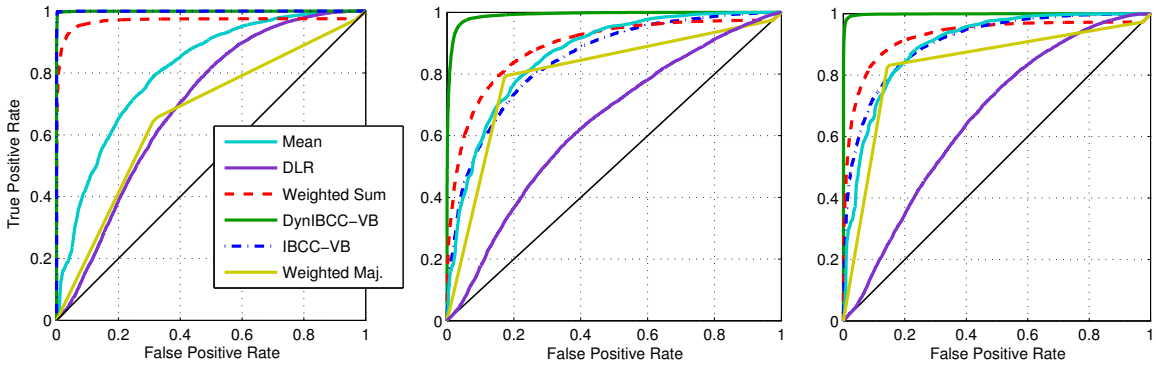
As with the static IBCC model, for any sample i_{unseen} not analysed by k, the terms relating to k are simply omitted from Equations 4.47 and 4.55. If we wish to predict the distribution over a decision $c^{(k)}_{i_{unseen}}$, we must also determine a time-step τ when $c^{(k)}_{i_{unseen}}$ occurs. To predict the decision at the next time-step, we simply use Equations 4.20 and 4.21 to find the priors over $\tilde{\boldsymbol{\pi}}^{(k)}_{N^{(k)}+1}$.

4.5 Synthetic Data Experiments

To test DynIBCC in a controlled environment, it is first run over sets of simulated agent decisions, where changes to agent reliability occur at known points. The basic experimental set-up is similar to that used for simulated experiments in Chapter 2, with the same hyperparameters provided for both static IBCC-VB and DynIBCC-VB. We have a binary classification problem with 1000 data points, of which 25 are training data. Predictions are supplied by 15 agents, simulated by logistic regressors, each one trained on different data and receiving different noisy sensor inputs. The task is to infer target labels for the test data. Each agent can be in one of two states: the informative state, where they predict with a mean accuracy of 0.846; or the uninformative state where accuracy is 0.5. Initially, three agents are in the informative state and twelve are uninformative. Three experiments are run: in the first there are no state changes to agents; the second introduces 15 sudden state changes at random points; the third includes 15 changes where agents transition grad-

ually between the informative and uninformative states, or vice-versa. Each experiment is repeated over 25 different datasets.

A number of alternative decision combination methods are included. The Weighted Sum and Weighted Majority methods used here update weights sequentially, so can adapt to changes in agent state. DLR is also a dynamic method that smoothly alters weights at each data point, evolving according to a hidden Markov process from their initial values, which are set to zero here. The DLR uses an extended Kalman Filter (EKF) to update the weights as target labels and agents' responses become available.



(a) No changes (b) Sudden changes. (c) Changes over 100 samples.

Figure 4.2: Receiver operating characteristics for dynamics experiments with simulated agents.

The AUCs and Brier scores for the experiments are listed in Table 4.1, and the ROC curves are shown in Figure 4.2. With no changes, DynIBCC-VB produces the same high level of accuracy as IBCC-VB, showing that it can handle the static case effectively. When sudden changes are introduced, the performance of IBCC-VB drops, while DynIBCC-VB remains above 0.99 AUC. Weighted Sum and Mean also outperform the static IBCC-VB in this case, and Weighted Majority sees an improvement, likely due to changes occurring before reaching a hard selection of a single agent. With gradual changes to agents, more agents are weakly informative through most of the data points. This results in a higher Mean, and contributes to improvements in all methods. DynIBCC-VB still significantly outperforms the static variant, suggesting that it can still distinguish changes of state when they are gradual.

Method	AUC		Brier Score	
	Mean	S.D.	Mean	S.D.
Mean	0.843	0.073	0.222	0.037
Weighted Sum	0.966	0.008	0.257	0.020
Weighted Maj.	0.663	0.065	0.488	0.077
DLR	0.698	0.026	0.451	0.014
IBCC-VB	1.000	0.000	0.002	0.002
DynIBCC-VB	1.000	0.000	0.002	0.003

(a) *No changes* to agents.

	AUC		Brier Score	
Mean	S.D.	Mean	S.D.
0.887	0.048	0.221	0.037
0.889	0.021	0.308	0.015
0.802	0.046	0.363	0.048
0.643	0.047	0.450	0.014
0.853	0.009	0.076	0.011
0.991	0.000	0.007	0.002

(b) *Sudden* changes to agents.

Method	AUC		Brier Score	
	Mean	S.D.	Mean	S.D.
Mean	0.918	0.025	0.195	0.033
Weighted Sum	0.928	0.018	0.301	0.017
Weighted Maj.	0.834	0.037	0.314	0.040
DLR	0.659	0.053	0.453	0.015
IBCC-VB	0.911	0.029	0.131	0.029
DynIBCC-VB	0.999	0.001	0.011	0.005

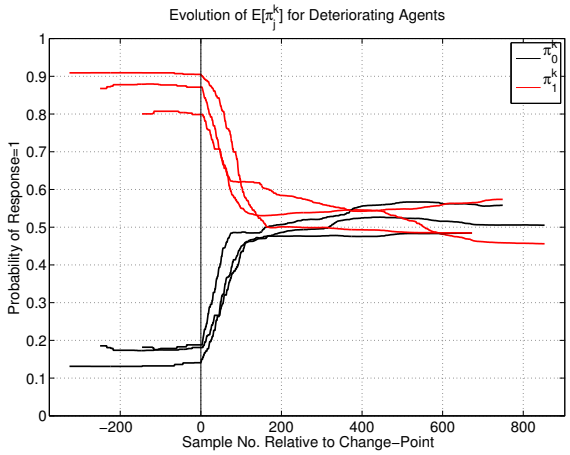
(c) *Gradual* changes to agents over 100 data points.

Table 4.1: Comparison of DynIBCC-VB with other decision combination methods over simulated data.

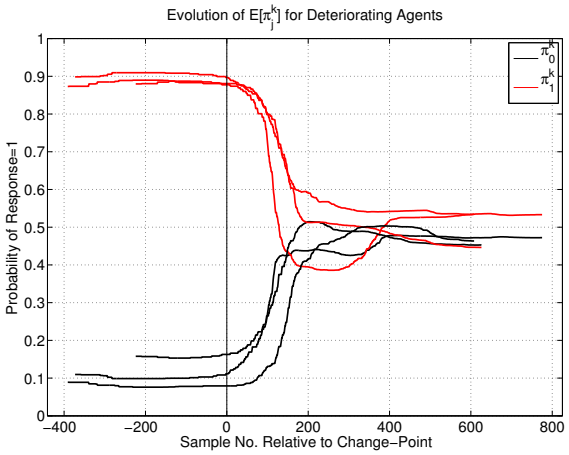
The confusion matrices learnt by DynIBCC-VB are now inspected to assess whether the algorithm has modelled the agent's behaviour adequately. The "true" confusion matrix, i.e. the generating distribution for the synthetic data, has the same form for any agents at data points where they are in the informative state: $\boldsymbol{\pi}_{inf}^{(k)} = \begin{pmatrix} 0.85 & 0.15 \\ 0.15 & 0.85 \end{pmatrix}$ At any point an agent is in the uninformative state, the "true" confusion matrix is: $\boldsymbol{\pi}_{un}^{(k)} = \begin{pmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{pmatrix}$. Figure 4.3 plots the confusion matrices learnt for the agents in the one test run of the experiments with sudden and gradual changes. The plots for each agent are aligned so that the x-axis is marked 0 where the change-points occur. For gradual changes, this marks the start of the transition between two states over the next 100 data points. Separate plots show different kinds of agent: *deteriorating* agents that change once from the informative to uninformative state; *improving* agents that change from uninformative to informative state; and *fluctuating* agents that improve then deteriorate. The plots also separate the

results for the experiments with sudden changes and those with gradual changes. Each line corresponds to one entry of the second column of the confusion matrix, showing its expected value at each data point relative to where a change occurred. In most cases, all agents within a single plot have a very similar pattern, with the algorithm producing a smooth transition between two states. However, there are notable differences in the changes we can see in each plot.

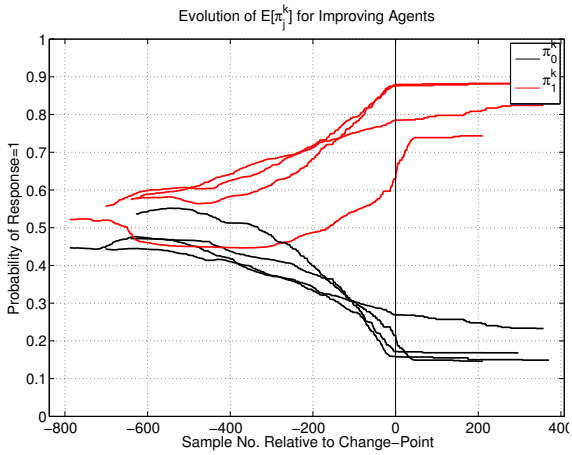
First, consider the difference between the deteriorating agents and the improving agents from both experiments. Deterioration is registered very quickly, while with the improving agents, the uninformative state is merged with the informative state over a large number of data points. This appears to result from the estimation of the state noise variance, q_τ, which acts as a forgetting factor on data from previous time-steps. When the agent's decisions become more random, q_τ increases, so the state transition is allowed to be quicker. However, when the agent becomes more informative, q_τ is zero, so there is no forgetting factor between the uninformative and informative states. This allows the model to accumulate evidence by increasing the values of $\boldsymbol{\alpha}^{(k)}$ during filtering over a number of time-steps. The low state noise in the informative states also means that information is passed back during smoothing from the informative to uninformative states. The evolution of q_τ is more clearly seen in Figure 4.4, where high values of q_τ occur for only the deteriorating agent and when both agents have very weak priors at the start of the learning process. To understand the effect of estimating q_τ for each time-step, the proposed approach to using a fixed forgetting factor, where q_τ is fixed to a pre-determined value at each step. This means that earlier observations have the same effect, regardless of whether a change in behaviour has occurred or not. Figure 4.5 shows the effect of fixing $q_\tau = 0.005$. The results show greater fluctuations, but also less crisp changes of state when an agent deteriorates or improves suddenly. This is due to less data being shared between time-steps within a stable period, and a lower forgetting rate when an agent becomes more random. For the gradually improving agents, the state change is more rapid as the forgetting factor is non-zero, but the change point is registered too early and no longer appears gradual. Figure 4.5 therefore



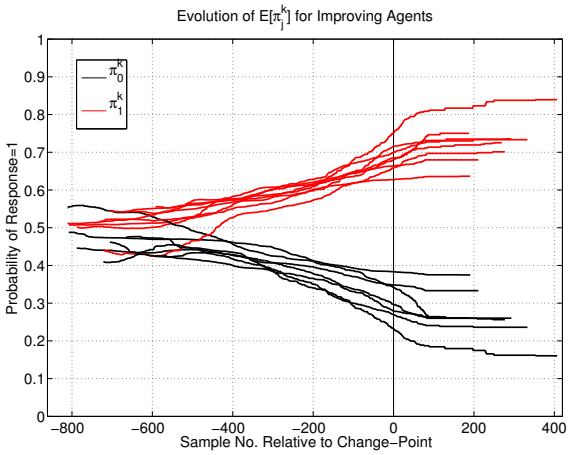
(a) Suddenly deteriorating agents



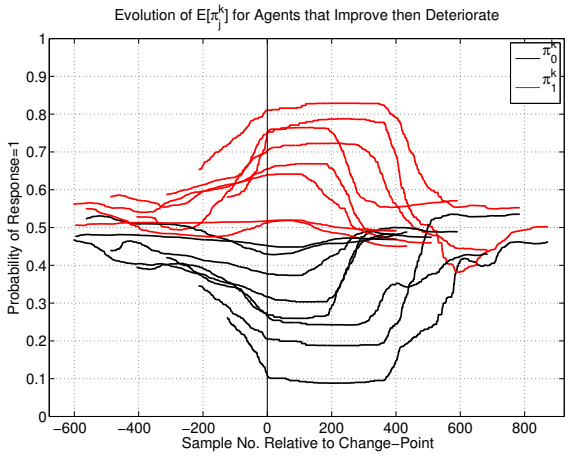
(b) Gradually deteriorating agents



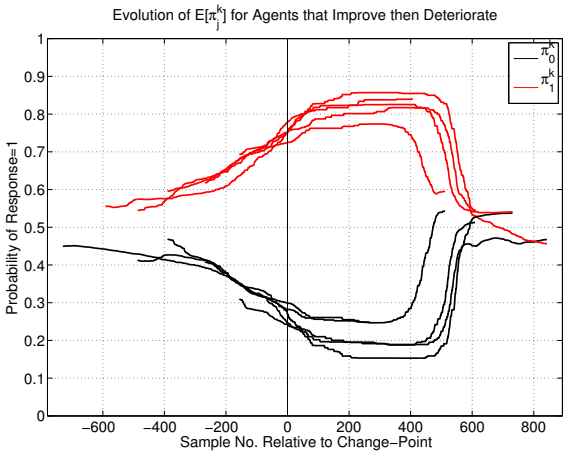
(c) Suddenly improving agents



(d) Gradually improving agents



(e) Suddenly fluctuating agents that improve then deteriorate



(f) Gradually fluctuating agents that improve then deteriorate

Figure 4.3: Dynamics of the $\boldsymbol{\pi}^{(k)}$-matrices inferred by DynIBCC-VB for individual agents for one dataset.

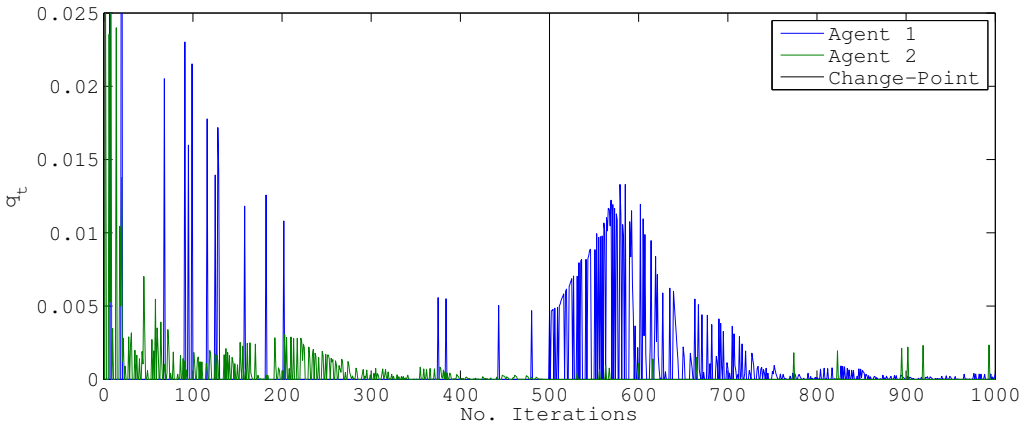


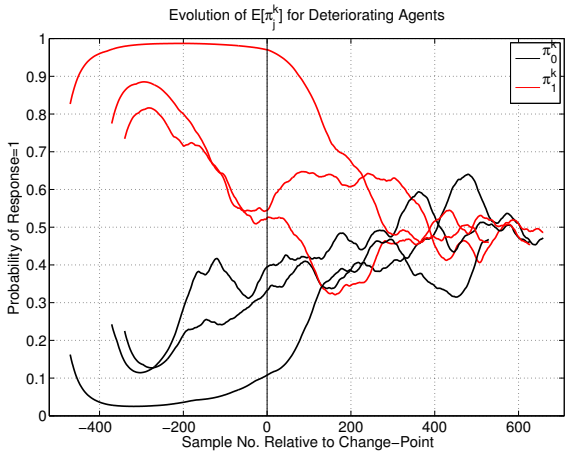
Figure 4.4: An example plot for two agents showing the state noise covariance, q_τ. Agent 1 deteriorates after 500 iterations, showing high values of q_τ, while and agent 2 improves.

highlights the benefit of adapting q_τ to the current observations.

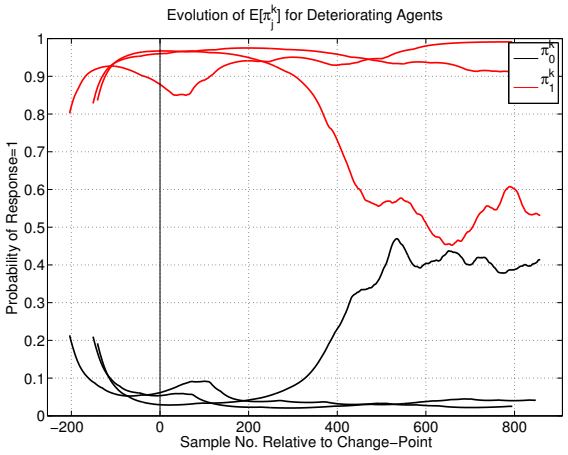
To address the issue of asymmetry in q_τ, future work may explore alternative criteria for estimating q_τ. However, the results in this section show that the present method learns a better model of dynamic agents than using a fixed forgetting factor and significantly improves on static IBCC. Most importantly, the proposed method for DynIBCC infers the latest state of an agent accurately, which is of greatest interest when deciding how to allocate tasks to workers.

The second point of interest is that sudden changes are not registered immediately, even for deterioriating agents, since the algorithm has insufficient information to pinpoint the exact sample at which a change occurs. However, the gradual changes are correctly inferred as taking approximately 100 data points more than the sudden changes.

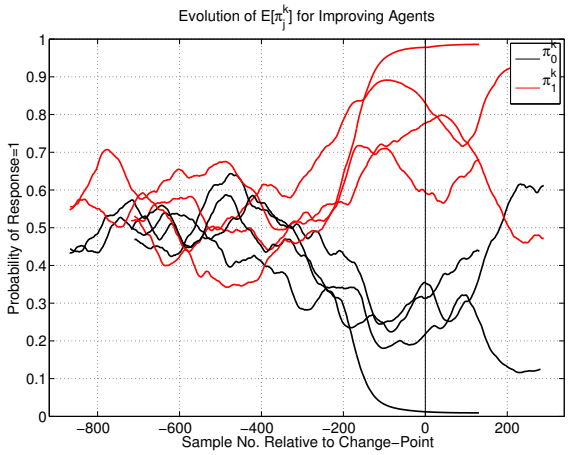
Thirdly, there are some differences between the $\boldsymbol{\pi}^{(k)}$ values for different agents in the informative state. Those agents that spend less time in the informative state have a lower value of $\pi_{11}^{(k)}$, appearing less informative. This occurs because there is insufficient data available from the short time in the informative state to learn the correct model. This occurs with agents in Figures 4.3c and 4.3d when the test finishes shortly after a change to the informative state. A more extreme occurrence are fluctuating agents in Figure 4.3e, where agents return to the uninformative state after approximately 200 data points. These later change-points are not explicitly labelled since they occur at different times for different



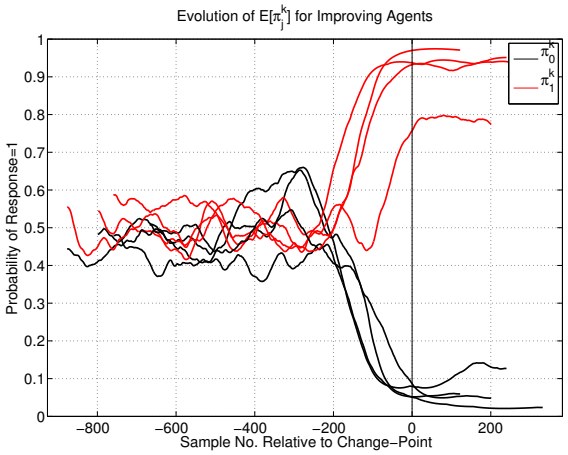
(a) Suddenly deteriorating agents



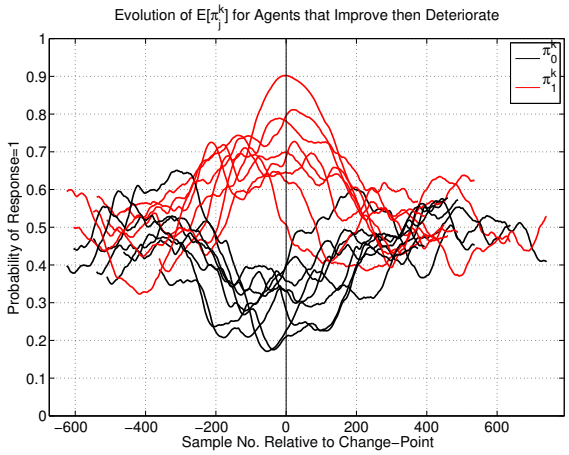
(b) Gradually deteriorating agents



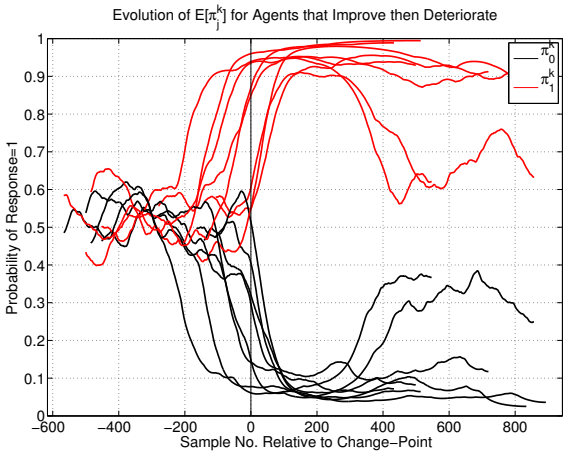
(c) Suddenly improving agents



(d) Gradually improving agents



(e) Suddenly fluctuating agents that improve then deteriorate



(f) Gradually fluctuating agents that improve then deteriorate

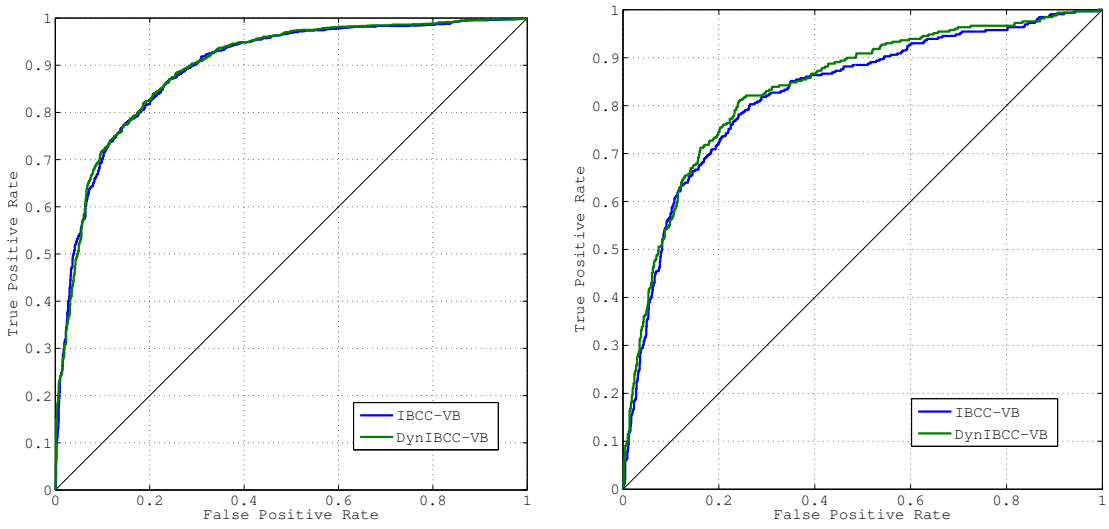
Figure 4.5: Using a fixed forgetting factor by setting $q_\tau = 0.005$ for all time-steps τ. Plots show the dynamics of the $\boldsymbol{\pi}^{(k)}$-matrices inferred by DynIBCC-VB for individual agents for one dataset.

agents, but are visible from the pattern of the graph.

The behaviours seen in these simulations reflect the uncertainty that occurs when only small amounts of data are available to DynIBCC-VB. The results show that DynIBCC-VB produces good estimates of the confusion matrices $\boldsymbol{\Pi}$ given these challenges, representing uncertainty in change-points in the manner intended.

4.6 Labelling Performance of DynIBCC with GZSN

The primary purpose of introducing dynamic confusion matrices is to track agents over time, providing an up-to-date model of their current state that will inform decisions such as task allocation. However, to show that DynIBCC is also capable of producing reliable combined decisions in a real-world scenario, the Galaxy Zoo Supernovae (GZSN) experiments from Chapter 3, Section 3.4 were repeated with DynIBCC-VB. The results show a small improvement over the static variant of IBCC, with AUC and Brier scores given in Table 4.2 and the ROC curves shown in Figure 4.6. This shows that the additional complexity of the DynIBCC model does not reduce its accuracy.



(a) Balanced datasets

(b) Imbalanced datasets

Figure 4.6: Receiver Operating Characteristic of DynIBCC-VB with GZSN datasets.

Method	Mean AUC	S.D. of AUC	Mean Brier Score	S.D. of Brier Score
IBCC-VB	0.897	0.040	0.146	0.039
DynIBCC-VB	0.900	0.034	0.136	0.030

(a) Balanced Datasets

Method	Mean AUC	S.D. of AUC	Mean Brier Score	S.D. of Brier Score
IBCC-VB	0.824	0.027	0.166	0.016
DynIBCC-VB	0.837	0.027	0.160	0.083

(b) Imbalanced Datasets

Table 4.2: Performance of DynIBCC-VB on GZSN Data.

4.7 Dynamics of Galaxy Zoo Supernovae Contributors

This section examines the dynamics of individual agents in the GZSN dataset, using the results from one run of the imbalanced dataset described in Chapter 3, Section 3.4. The dynamics inferred by DynIBCC-VB vary significantly between volunteers, in many cases showing sustained drifts in a particular direction, suggesting real changes in the behaviour of the human agents. Examples of the types of changes found are presented below in Figures 4.7 to 4.10. Each figure consists of two plots, which show different views of a 3-D ternary plot of the expected confusion matrices of a particular agent at each time-step, where time-steps correspond to responses from the agent. Each line corresponds to the confusion vector $\boldsymbol{\pi}_j^{(k)}$ for target value j. In this ternary plot, the proximity of a point on the line to a corner of the plot indicates the probability of the agent's response. So, for the line marked "supernova", points close to the vertex $score = 3$ indicate that there is a high probability that the agent responds with score 3 when presented with an object of target label "supernova". Two views are shown: the lower plots show an end-on view of the 3-D upper plots to help the reader to determine the correct location of the lines in the ternary plot.

The first example in Figure 4.7 shows a volunteer whose behaviour is fairly stable over time, with small drifts in both target values toward assigning as score of 1. Given the similar response distributions for both target values, the agent is not highly informative. In Figure 4.8 there is a more sustained drift for both target label values, but in different directions,

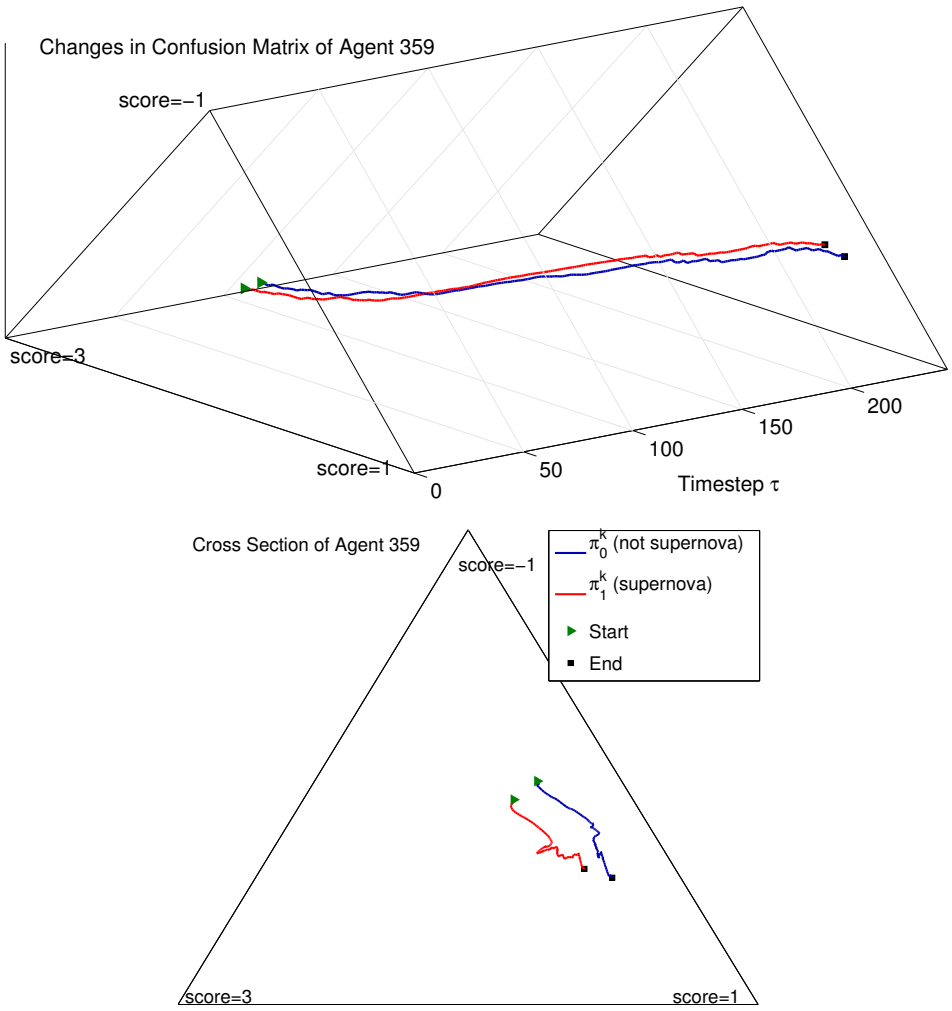


Figure 4.7: An agent with a small drift toward responses of 1.

with a clear improvement in correctly assigning a score of 3 to genuine supernovae. As the gap widens over time, the agent's responses become more informative. Note from the top plot that changes are more rapid early on. Figure 4.9 shows an agent with continuing changes. First, the agent improves rapidly as the distribution of responses to "supernova" objects moves towards a score of 3, but the agent gradually becomes less informative, with an overall shift towards giving responses of 1, indicating uncertainty. Finally, Figure 4.10 contains examples of more sudden changes. After the responses for both target values initially drift toward uncertain responses (score=1), a distinct change-point occurs with the agent becoming far more informative. After this change, there is a steady drift toward giving responses of 3 for both target values.

To illustrate the diversity in the behavioural changes, DynIBCC was run over the same

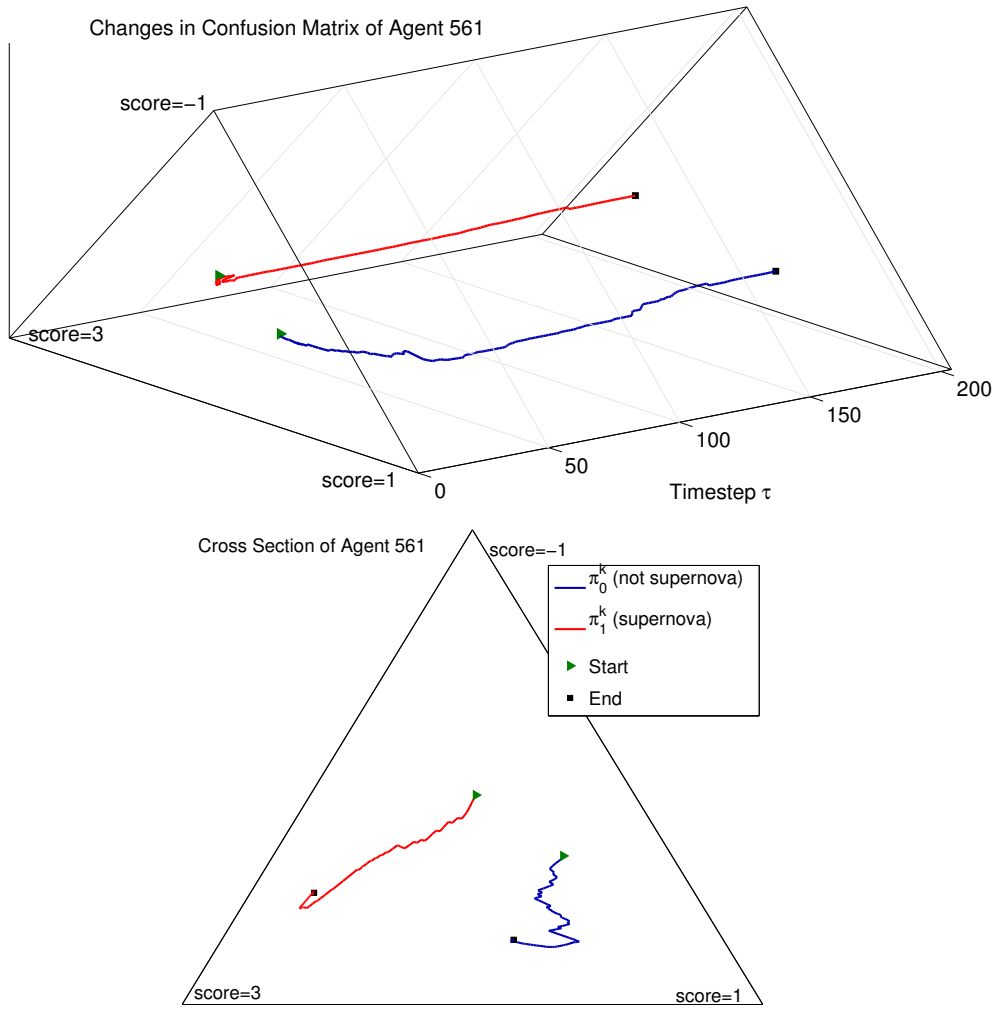


Figure 4.8: An improving agent with some overall drift away from score=-1.

dataset, with all ground truth target labels supplied to the algorithm. This means that the confusion matrices are learned from verified target labels, rather than from those that we infer using DynIBCC. To remove any effect of priors $\boldsymbol{A}_0$, we set all values to 1, i.e. uninformative priors. This means that the smoothing step can move the values of the confusion matrix away from their priors so that the confusion matrices are learned entirely from verified data points. Figure 4.11 plots the confusion matrices over the first 50 time-steps of all agents with more than 50 verified responses (data points for which the ground truth is available). Here, we see that there is no discernible pattern of behavioural change common to all agents. However, a notable number of agents vary their behaviour after the first few responses, and many agents have a high tendency to answer "1" to objects that are not supernovae. The ability to correctly recognise supernovae is clearly extremely mixed.

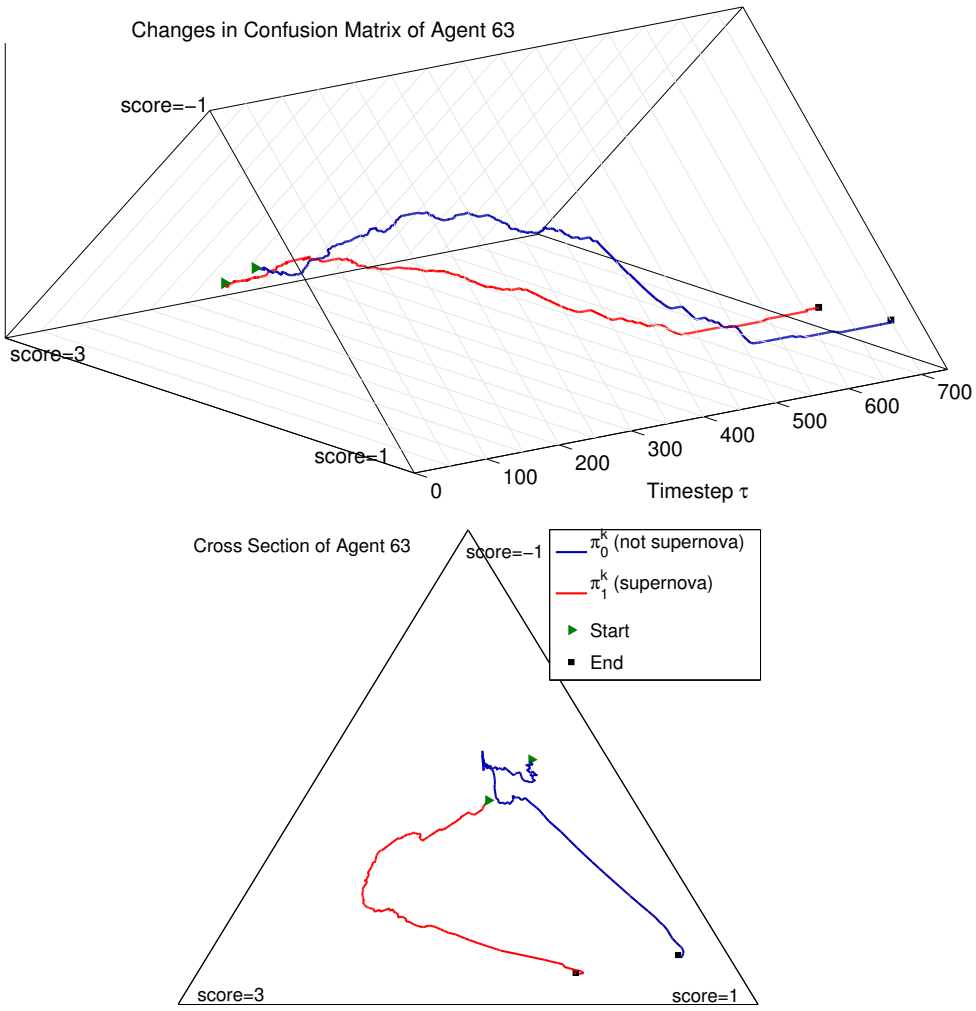


Figure 4.9: An agent with sustained drift and rapid changes that differ between target values.

The examples in this section show that changes can be sudden, sustained over a long period, and may affect responses to both target values, or just to one. Such variation between agents suggests that agents may benefit from bespoke interaction with the Galaxy Zoo Supernovae system, which may, for example, assist agents to improve when they present certain behaviour. For example, the agents that drift toward uncertain responses of 1 may benefit from training. The information in the dataset is insufficient to explain the reasons for changes in the volunteers' behaviour. Possible causes for agents improving, particularly after the first few time-steps, are that they are learning or adapting their behaviour as they become accustomed to the tasks. The user interface may also have undergone some changes during the time the data was collected, as may the method for capturing and pre-

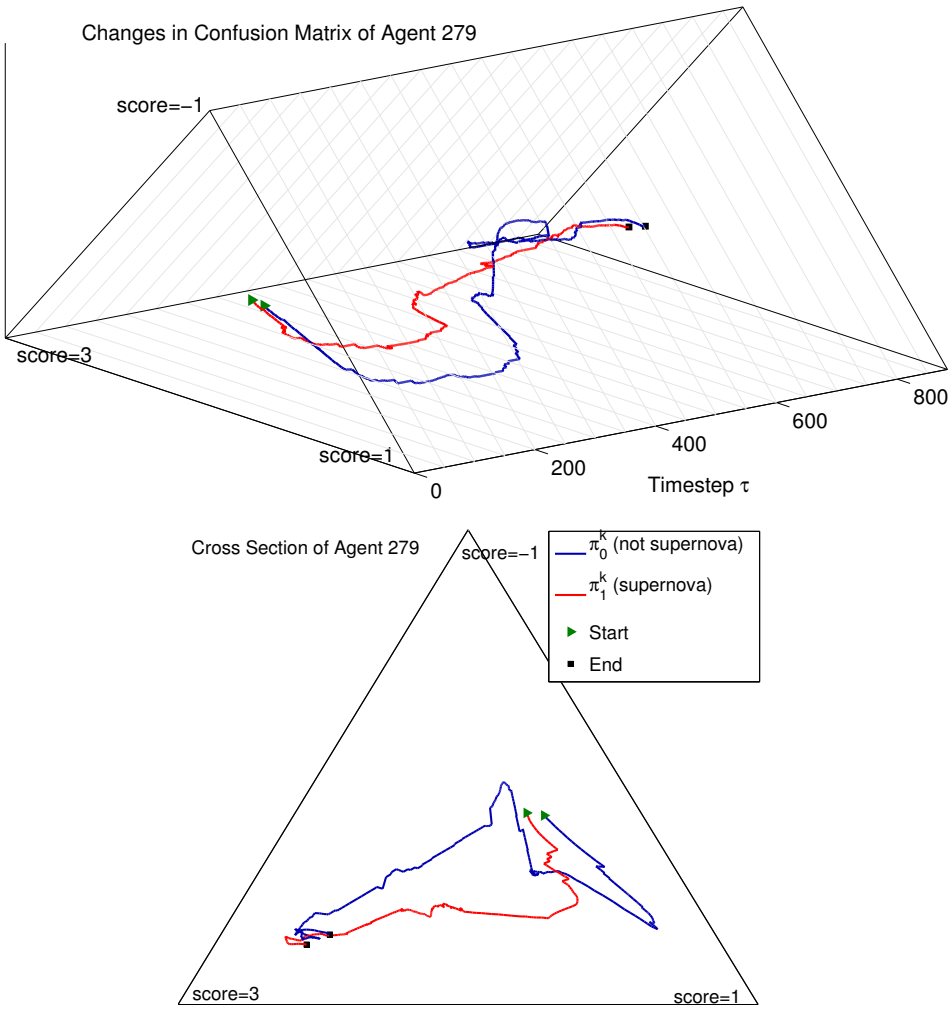


Figure 4.10: An agent with many fluctuations with an overall tendency toward always giving score=3.

processing the images that were presented to volunteers. This information is not recorded in the present dataset, so future analysis may be improved by capturing any such changes by recording the provenance of information in a standardised manner [Huynh et al., 2013]. The dynamic confusion matrices could provide a valuable resource for analysing the effect of changes to the system, as well as measuring the results of any training carried out. The shifts in agent behaviour could also allow better planning of when to hire new agents or intervene to assist uninformative agents.

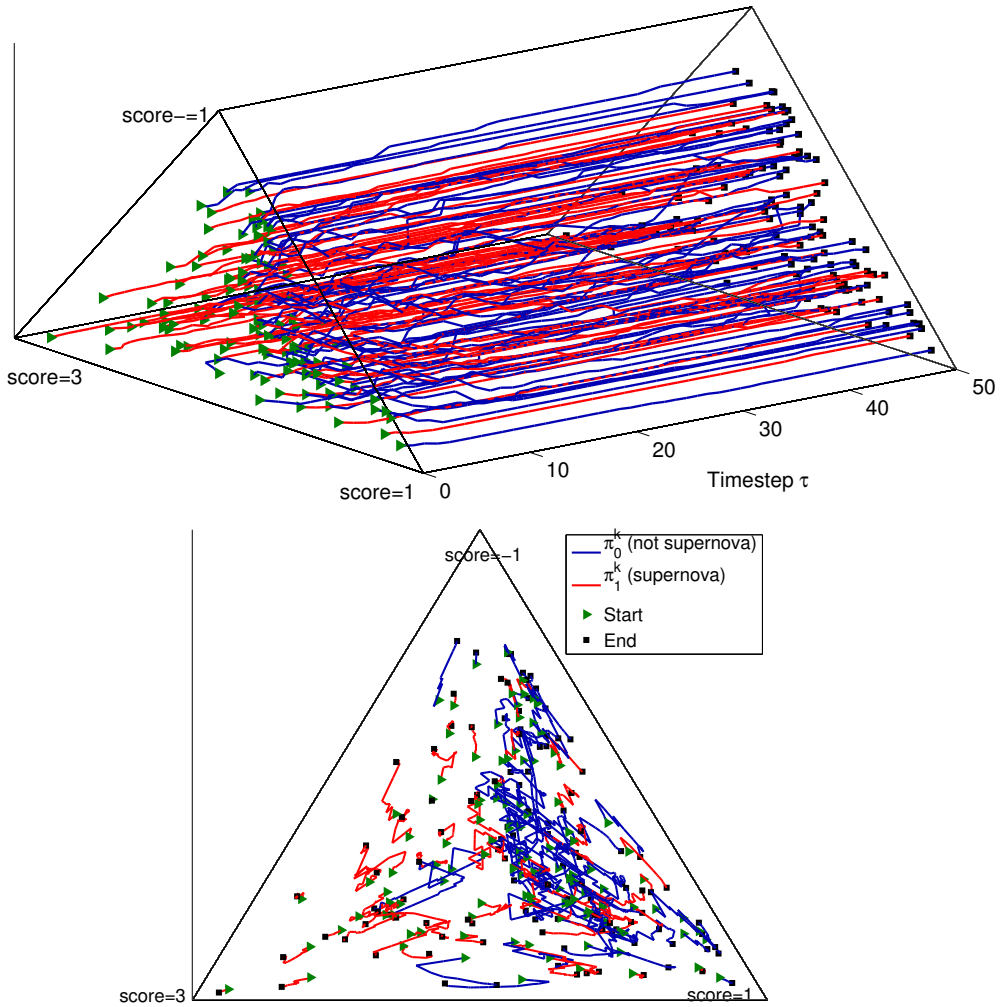
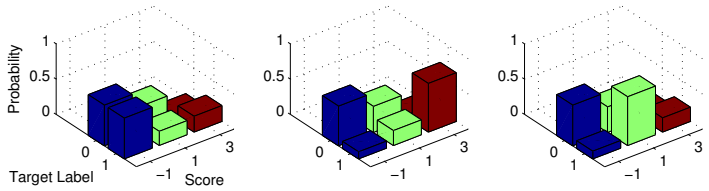


Figure 4.11: Behavioural changes of all agents with more than 50 verified responses, showing the diversity in behavioural changes.

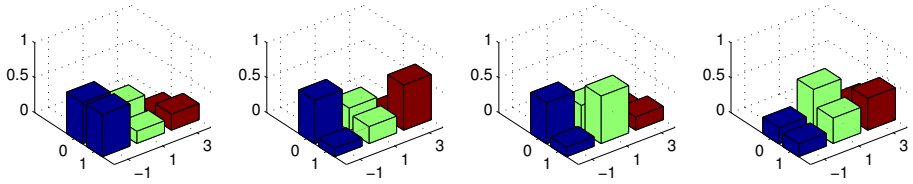
4.8 Dynamics of Π Communities

The community analysis method used in Section 3.7.1 is applied to the dynamic confusion matrices inferred from an imbalanced GZSN dataset to examine the development of the community structure over time. The dataset contains all data points used in Section 3.4.2, plus any unlabelled data points for the same agents that might show changing behaviour, and all available training data is used when learning DynIBCC-VB. The community detection method of [Psorakis et al., 2011] is run over an adjacency matrix, using Equation 3.30 with the most recent confusion matrices for all agents after s classification tasks have been completed.

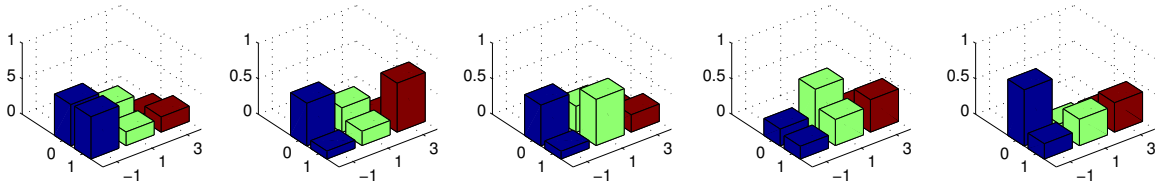
Figure 4.12 shows how the same communities we saw in Section 3.7.1, Figure 3.16



(a) 3000 tasks completed.



(b) 12000 tasks completed.



(c) 26558 tasks completed.

Figure 4.12: $\mathbf{\Pi}$ communities: means over the expected confusion matrices of community members after different numbers of tasks. At each time point we see a new community appear while previous communities persist with similar means.

emerge over time. Initially, only three communities are present, with those corresponding to groups 4 ("optimists") and 1 ("reasonable") in Figure 3.16 only appearing after 12,000 and 26,558 responses have been submitted by the citizen scientists. The "reasonable" group is the last to emerge and most closely reflects the way the designers of the system intend good decision makers to behave. It may therefore appear as a result of new volunteers joining with different behaviour, or participants learning, or of modifications to the user interface or instructions as the Galaxy Zoo Supernovae application was being developed to encourage this behaviour.

Figure 4.13 shows the node participation scores at each number of tasks s for selected individuals who completed a large number of tasks. The community detection method of [Psorakis et al., 2011] allows for overlapping or uncertain community membership, so at some points in time the agents may be in an intermediate state between communities.

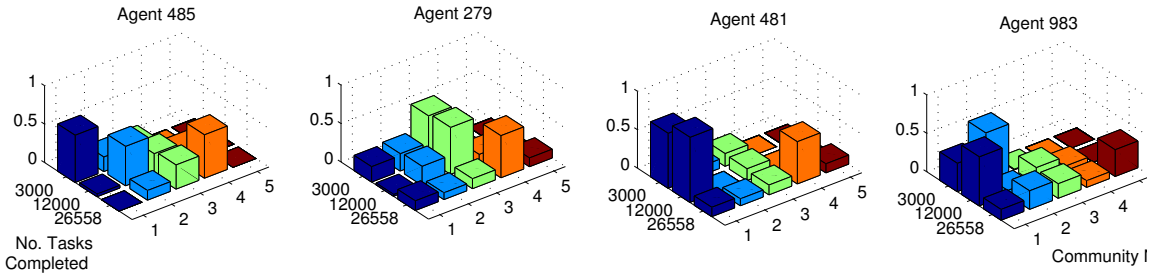


Figure 4.13: Node participation scores for the π communities for selected individuals after different numbers of tasks. Each bar shows the individual's participation in a particular community after running the community analysis over confusion matrices produced from observations up to that point. Participation scores close to 1 indicate very strong membership of a community. Node participation scores at each number of tasks completed sum to one over the communities 1 to 5.

Note that these agents switch between communities, as significant changes to the individual's confusion matrix occur. For example, Agent 279 is shown in Figure 4.10 to have a significant change toward more commonly responding with a score of 3. In Figure 4.13 we can see a change of community membership that reflects this between 12,000 and 26,558 tasks. If we examine the prototypical confusion matrices of communities in Figure 4.12, we can see that the change in agent 279 from community 3 to community 4 corresponds to an decreased likelihood of score -1 and increased likelihood of score 3. Agents that do not produce classifications after 3000 tasks could be frozen in one community, since the clustering algorithm uses the latest dynamic confusion matrix for each agent. The original communities appear to be persistent groupings of agents' behaviour, despite the addition of new agents at each time slice and the changes in existing agents' behaviour, with the example agents in 4.13 changing membership of these communities.

4.9 Dynamics of Common Task Communities

The final part of this chapter considers the evolution of common task communities to observe the effect of recent tasks on the community structure and confusion matrices. Common task communities were introduced in Chapter 3, Section 3.7.2, as communities of agents who have classified similar sets of objects. The aim of this analysis is to show

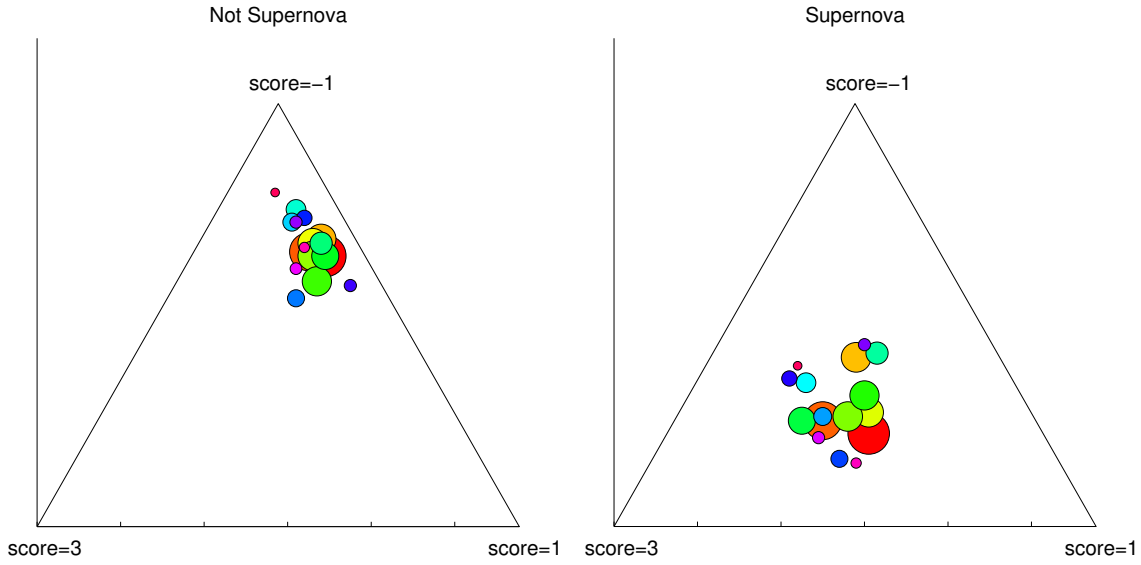


Figure 4.14: Mean expected confusion matrices of common task communities after 50,000 tasks. Proximity to a vertex indicates likelihood of a score given the target class shown above each plot. Size of circles indicates number of community members.

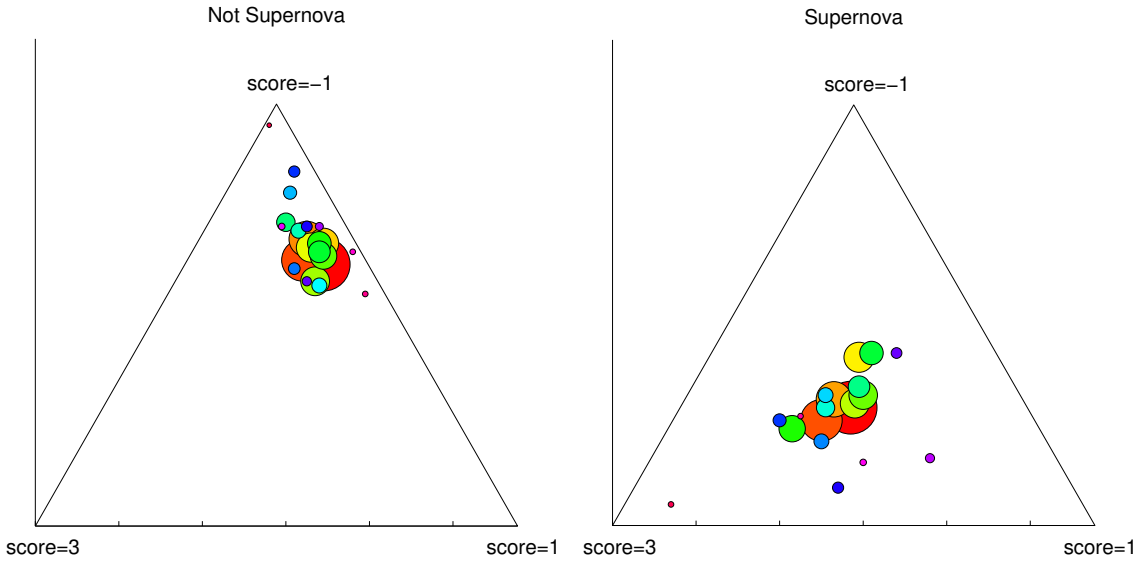


Figure 4.15: Mean expected confusion matrices of common task communities after 200,000 tasks. Proximity to a vertex indicates likelihood of a score given the target class shown above each plot. Size of circles indicates number of community members.

whether the mean confusion matrices of the communities change as more tasks are completed, indicating a relationship between the objects seen and the confusion matrices we infer. The experiment described in Section 3.7.2 is repeated over three batches of tasks: the first 50,000 tasks; the first 200,000 tasks; the complete set of 493,048 tasks, includ-

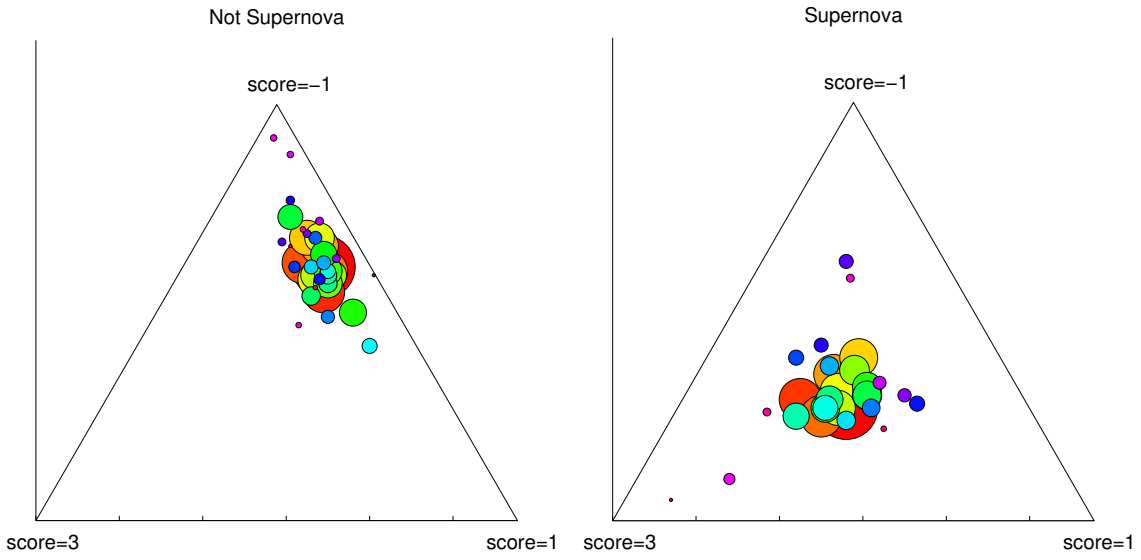


Figure 4.16: Mean expected confusion matrices of common task communities after 493,048 tasks. Proximity to a vertex indicates likelihood of a score given the target class shown above each plot. Size of circles indicates number of community members.

ing unlabelled data points that indicate shared tasks between agents. The algorithm produced community structures with modularities of 0.67, 0.69 and 0.75 respectively, showing that good community structure is present for smaller periods of observations, but emerges more distinctly over time. Figures 4.14, 4.15 and 4.16 show for each batch of classifications the means of the most recent confusion matrices of community members inferred using DynIBCC-VB. Since DynIBCC models the dynamics of agent confusion matrices as a random walk, the observations closest to the current time have the strongest effect on the distribution over the confusion matrices. Therefore, the mean of expected confusion matrices characterise a community at a given point in time. Some changes to the means may be the result of observing more data about the agents. However, individual behaviours may also evolve as a result of learning new types of tasks or changes to individual circumstances, such as the times a volunteer is available to carry out tasks. Thus, changes in an individual's community membership may have implications on their availability or willingness to complete certain types of task.

In all three networks there is a persistent core for both target values, where the means for the large communities remain similar. Some communities within this group move a

small amount, for example, the large red community in the "supernova" class. In contrast, we see more scattered small communities appear after 200,000 tasks and at 493,048 tasks. It is possible that the increase in number of agents as we see more data means that previous individual outliers are now able to form communities with similar outliers. Therefore outlying communities could be hard to detect with smaller data-sets. However, many outliers appear in the same place in only one of the figures, suggesting that they may contain new agents that have made few classifications up to that point. Some are less transient however: the top-most community in the "not supernova" class in Figures 4.15 and 4.16 moves only a small amount. Similar sets of tasks may produce more extreme confusion matrices such as these for different agents at different times, implying that these tasks induce a particular bias in the confusion matrices.

The changes we observe in Figures 4.14, 4.15 and 4.16 demonstrate how the tasks completed affect the confusion matrices we infer and alter the communities of related agents. Future work may seek to quantify which tasks have a particular effect on a community of agents, as these may suggest suitable training examples. Future investigations may also need to consider modifying the co-occurrence network to discount older associations between agents, if the dynamics of common task communities are to be tracked over a longer period.

4.10 Discussion

This chapter proposed a novel method for decision combination that enables the tracking of agents' behaviour over time. This model, DynIBCC, is based on Independent Bayesian Classifier Combination (IBCC), which assumes static decision-making behaviour. As with IBCC, this chapter developed an efficient variational Bayesian inference algorithm, DynIBCC-VB, in this case introducing additional filtering and smoothing steps to handle the dynamics. The approach was tested first on synthetic data, showing that the algorithm can detect both sudden and gradual changes in agent behaviour and produce superior performance over static IBCC. Using real data from Galaxy Zoo Supernovae, the algorithm

performs favourably compared to static IBCC, continuing to out-perform alternative methods despite the additional complexity of tracking agents. Behavioural changes in real citizen scientists were charted, indicating shifts in the way human volunteers make decisions over time. Social network analysis methods can also be used to investigate how different groups of agents emerge over time, as agents change and new agents join.

The rich information learned using DynIBCC has huge potential to improve the efficiency of decision making with multi-agent systems. For example, we can use the changes to the confusion matrices, $\underline{\boldsymbol{\Pi}}$, and the community structure to gauge the effect of actions such as training, changing the interaction between agents in the system, or altering user interfaces to human agents. Common task communities and $\boldsymbol{\pi}$ communities may assist in estimating the effects of task assignments and training on related individuals. They could also be exploited to reduce the size of the task assignment problem to one of choosing classifiers from a small number of groups rather than evaluating each classifier individually. The next chapter shows how the expressive model of changing agent behaviour provided by DynIBCC can be used to intelligently select agents for particular tasks.

Chapter 5

Intelligent Tasking

Previous chapters focused on the task of aggregating responses from multiple agents and inferring their reliability. This chapter considers the complete system for obtaining those responses and influencing the agents through weak control decisions, such as suggesting optimal tasks to agents. This situation is depicted in Figure 5.1, showing how an agent may exert weak control to influence the connections between agents and objects, which represent their current analysis tasks. An information-theoretic approach labelled *intelligent tasking* is proposed to determine the optimal tasks for aggregating information from multiple agents. The results demonstrate clear advantages over more simplistic approaches, but also indicate opportunities for future work to automate other facets of weak control, such as agent training and improvement.

Within a multi-agent system (MAS), the individual agents share information as they

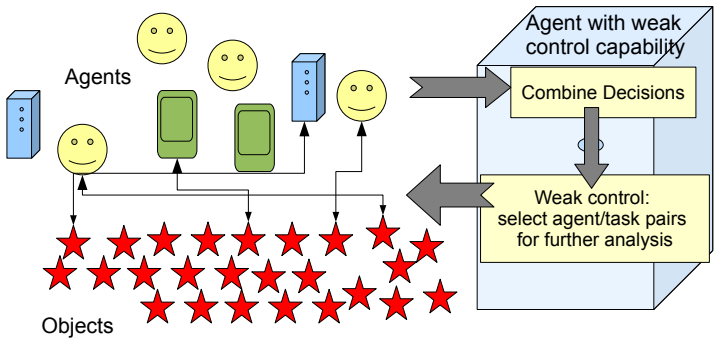


Figure 5.1: Overview of the intelligent tasking problem: how to assign tasks to agents given current combined decisions.

observe and analyse objects or the environment. Often there are many objects to analyse, locations from which to observe, and agents from which information could be gathered. Certain tasks may provide more information than others, but this also depends on how the agents' skills suit those tasks. Choosing suitable pairs of agents and tasks is therefore important when aggregating decisions as efficiently as possible. To facilitate this choice and maximise the amount of data we can analyse, we could exploit additional information obtained through computational analysis of the objects or environment of interest.

Depending on the application, either a central control node can direct the agents, or individual agents can determine tasks that optimise the collaborative output. In Citizen Science projects, a central node must allow volunteers to accept or reject tasks, but can still provide *weak control* through a series of nudges and suggestions toward tasks that meet both the system's goals, and enhance the experience of citizen scientists, helping them find suitable training exercises and diverse tasks.

The chapter begins by looking at related work on information aggregation systems and whether they account for these issues. A case study is then introduced for a crowdsourcing system in which it is important to select and deploy agents efficiently. An extension to DynIBCC is proposed to enable discrete agent decisions to be augmented by continuous features in the range $[0, 1]$. This extension is demonstrated with the crowdsourcing case study, attaining strong performance with limited data, followed by introducing an intelligent tasking framework for optimising the deployment of agents, which naturally negotiates the need to explore and exploit the agents' skills. The approach is used to develop the *Hiring and Firing* algorithm, which addresses the need to select tasks and agents in a unified manner. While early results are promising, there remains a wealth of opportunities for extensions to intelligent tasking, some of which are are discussed in the final chapter.

5.1 Related Work

In many existing systems, there is no attempt to select agents to perform particular tasks based on ability or diversity of skills. In Citizen Science applications, such as Galaxy

Zoo [Smith and Lintott, 2010], the aim is to assign more agents to a task until a clear combined decision has been made. For example, Galaxy Zoo Supernovae [Smith et al., 2010], prioritises objects that have no classifications, and does not request labels for those that already have a sufficient number of answers that agree. The remaining objects are prioritised according to the probability of a positive example. Thus, the system uses a heuristic method to label uncertain objects. The choice of whether to hire more agents to classify a Galaxy Zoo object is addressed by [Kamar et al., 2012] using a partially-observable Markov Decision process (POMDP), but this choice is not tailored to the individual agents, which are not modelled in their approach.

Related work on crowdsourcing has considered the problem of selecting trustworthy workers. Web-based crowdsourcing platforms such as *Amazon Mechanical Turk (AMT)*[1] allow workers to receive payments for tasks presented through its web interface, but have been shown to suffer from unreliable workers, including spammers who guess random answers to complete tasks more rapidly for money [Bloodgood and Callison-Burch, 2010; Ipeirotis et al., 2010]. Some systems focus on rejecting unreliable workers, but assume constant reliability [Raykar and Yu, 2012; Ipeirotis et al., 2010; Liu et al., 2012]. For example, [Raykar and Yu, 2012] provides a mechanism for blocking spammers on the fly. In [Liu et al., 2012], various algorithms are presented for inferring a single reliability metric, where priors can be set to identify workers as spammers or hammers, i.e. trustworthy workers. A method proposed by [Ipeirotis et al., 2010] infers a single error and bias measure per agent for blocking unreliable workers. Since these methods do not model the changing worker dynamics, they in effect treat agents' distant past responses as a significant indication of current reliability. Thus they are unable to account for learning, boredom, or the movement of agents who are mobile observers. Worker dynamics are addressed by [Donmez et al., 2010], who demonstrate how to reject poor performers from a pool of workers by thresholding a changing reliability value. These methods are restricted to iteratively filtering workers, and do not consider the choice of task, e.g. which object to

[1] See https://www.mturk.com

label, which affects the information gain over the target variables and can influence future behaviour of agents. Most of these methods assign scalar reliability scores [Donmez et al., 2010; Liu et al., 2012; Ipeirotis et al., 2010], so are unable to consider how a worker's reliability varies between types of task, which may be the result of their expertise or boredom with a particular task type. For example, a bored worker may be reinvigorated by completing a different kind of task. Therefore, there are advantages to using confusion matrices as an underlying representation, as in IBCC, DynIBCC and in [Raykar and Yu, 2012; Ipeirotis et al., 2010; Liu et al., 2012; Dawid and Skene, 1979], as the behaviour with each type of task is characterised by a row in the matrix.

In other work on crowdsourcing by [Quinn et al., 2010], tasks are allocated to either humans or artificial agents according to speed, cost and quality constraints. However, the system makes decisions according to prior knowledge about agent types rather than observing the abilities of individuals.

Several pieces of related work have considered *active learning* with crowdsourcing. Active learning in this context refers to the iterative process of deciding which objects to label, and choosing a labeller, who may be unreliable. In [Yan et al., 2011], a strategy is developed for binary classification where agents are selected based on how confident they are likely to be for a particular task. However, reliable confidence measures are often unavailable, especially for human agents. The work of [Chen et al., 2013] implements a learning strategy for ranking problems, which seeks to maximise expected information gain over both the model and the target variables, introducing a heuristic parameter to balance the exploitation and exploration of the reliability model of the workers. In summary, the related work has not yet produced a unified approach for adaptively assigning labelling tasks to agents in a bespoke manner. The next section presents a case study for which the following sections develop a methodology to account for these factors.

5.2 Case Study: TREC Crowdsourcing Challenge

The work in this chapter relates to an information aggregation problem that requires the efficient use of unreliable workers. The 2012 TREC Crowdsourcing challenge[2] was a competition to determine whether documents in a given dataset were relevant to a set of 10 search queries. The complete dataset contains 15,424 documents and 18,260 document/query pairs that must be confirmed as true or false. Each search query has a detailed description of a very specific information need, so that it is not possible to confidently judge relevance by searching for a short text string. Examples of topic titles include "definition of creativity" and "recovery of treasure from sunken ships", with the descriptions that follow specifying the query more precisely. The documents were compiled into the TREC 8 corpus, originally sourced from the *Financial Times*, *Los Angeles Times* and *Federal Register*. No training examples were provided for the given queries, the aim of the challenge being to use crowdsourcing to obtain accurate document relevance classifications. However, with a large number of document/query pairs, it is desirable to reduce the number of relevance judgements we need to obtain from the crowd to limit the time and cost taken to classify the complete dataset. Given a subset of crowdsourced training examples, textual features extracted from the documents allow for the prediction of labels for documents that have not been processed by the crowd. These predictions could potentially be used to prioritise documents for further crowdsourcing, for example, where their classification is most uncertain. This chapter therefore presents an approach that employs more expensive human agents only when necessary, using cheaper automated techniques when possible, aggregating both types of information.

In contrast to the categorical responses provided by the crowd, textual features may be unbounded discrete variables, such as word counts, or continuous variables, such as probabilities. However, the decision combination methods considered so far, including IBCC, have only been used to aggregate nominal decisions such as class labels. The results

[2]The Text REtrieval Conference, or TREC, consists of several competitions. For the crowdsourcing challenge, see https://sites.google.com/site/treccrowd/.

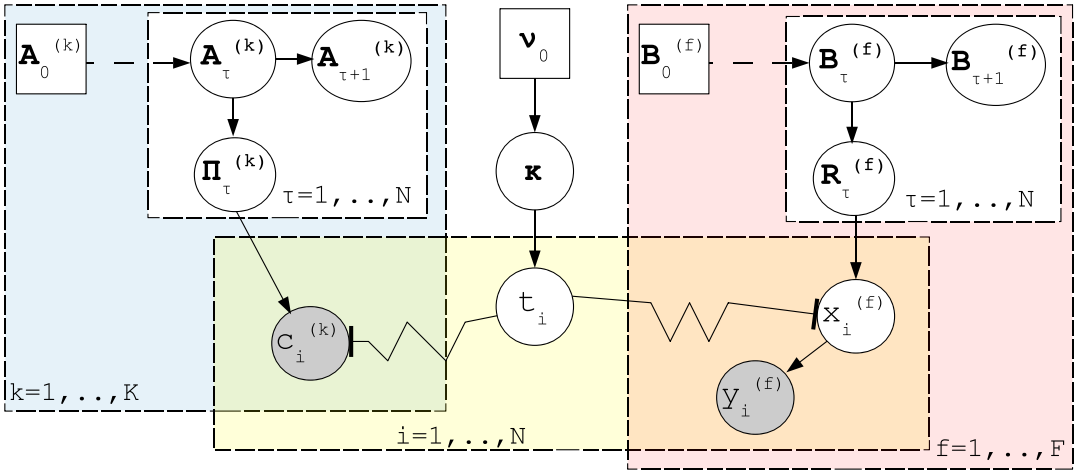


Figure 5.2: Graphical model for DynIBCC extended to accommodate continuous features. Dashed arrows indicate dependencies on nodes at other time-steps. The zig-zag line means t_i is a switch that selects parameters from $\Pi_\tau^{(k)}$. The shaded node represents observed values. Circular, hollow nodes are variables with a distribution. Square nodes are variables instantiated with point values.

in Chapter 3, Section 3.6 show that aggregating binary textual features using IBCC can produce accurate classifications by treating features in the same way as responses from agents. However, it may not be possible to compress all kinds of feature to a small number of categories. For example, it may not be possible to simply threshold a continuous variable about 0.5 if that feature is not a prediction of the target decision. The next section details how the IBCC model can be extended to handle such continuous variables, enabling its application to the TREC Crowdsourcing challenge. The following section then presents the results of the competition, and discusses some of the crowdsourcing issues arising in this scenario that motivate a unified intelligent tasking approach.

5.3 DynIBCC for Combining Probabilities

A diverse set of information sources often requires the aggregator to combine both discrete and continuous measurements, such as probability estimates. The extension to DynIBCC proposed here naturally accommodates mixed discrete variables and continuous variables in the range $[0, 1]$. The graphical model for the modified DynIBCC approach is shown

in Figure 5.2. The model is extended from that of Chapter 4 through the addition of the right-hand plate, shaded pink. The variables relating directly to target objects, $\boldsymbol{\nu}_0$, $\boldsymbol{\kappa}$ and t_i are as for DynIBCC, as are the variables concerned with agents $k = 1, .., K$, where the discrete responses are observed (blue plate on left). Features for which we observe only probabilities, $f = 1, .., F$, are treated as agents for whom we observe distributions over latent response variables $x_i^{(f)}$ for objects $i = 1, .., N$. In effect, we could view the probabilities as spreading a single response between multiple discrete values. Each feature, f, has time-dependent confusion matrix, $\boldsymbol{R}_\tau^{(f)} = \{\boldsymbol{\rho}_{\tau,j}^{(f)} | j = 1, .., J\}$, where each row $\boldsymbol{\rho}_{\tau,j}^{(f)}$ is a parameter vector for a categorical distribution with elements

$$\rho_{\tau,j,l}^{(f)} = p(x_i^{(f)} = l | t_i = j, \boldsymbol{R}_\tau^{(f)}, \tau = r(i,f)), \tag{5.1}$$

where $r(i, f)$ maps the object index i to time τ at which the feature f was recorded for i. For many features, it may be appropriate to assume a static confusion matrix. However, dynamics may be important if the target labels undergo concept drift, or the feature is a sensor reading for a moving target object. The rows in the confusion matrix each have a Dirichlet prior with hyperparameters $\boldsymbol{\beta}_{\tau,j}^{(f)}$. The matrix of hyperparameters for all target values j is referred to as $\boldsymbol{B}^{(f)} = \{\boldsymbol{\beta}_{\tau,j}^{(f)} | j = 1, .., J\}$. Features are therefore modelled using confusion matrices in a similar manner to agents, but are separated here for clarity since the discrete value $x_i^{(f)}$ is unobserved. The observed vector $\boldsymbol{y}_i^{(f)}$ describes a categorical distribution over the feature value, such that $y_{i,l}^{(f)} = p(x_i^{(f)} = l)$. The complete model for the extended DynIBCC is represented by the joint distribution, which is an extension of Equation 4.2:

$$p(\boldsymbol{t}, \boldsymbol{y}, \boldsymbol{c}, \boldsymbol{\kappa}, \underline{\boldsymbol{R}}, \underline{\boldsymbol{\Pi}} | \boldsymbol{B}_0, \boldsymbol{A}_0, \boldsymbol{\nu}_0) = \prod_{i=1}^{N} \left\{ \kappa_{t_i} \prod_{k=1}^{K} \pi_{s(i,k),t_i,c_i^{(k)}}^{(k)} \cdot \prod_{f=1}^{F} \sum_{l=1}^{L} y_i^{(f)} \rho_{r(i,f),t_i,l}^{(f)} \right\} p(\boldsymbol{\kappa} | \boldsymbol{\nu}_0)$$
$$\prod_{\tau=1}^{N} \prod_{j=1}^{J} \left\{ \prod_{k=1}^{K} p\left(\boldsymbol{\pi}_{\tau,j}^{(k)} | \boldsymbol{\alpha}_{\tau,j}^{(k)}\right) p\left(\boldsymbol{\alpha}_{\tau,j}^{(k)} | \boldsymbol{\alpha}_{\tau-1,j}^{(k)}\right) \cdot \prod_{f=1}^{F} p\left(\rho_{\tau,j}^{(f)} | \boldsymbol{\beta}_{\tau,j}^{(f)}\right) p\left(\boldsymbol{\beta}_{\tau,j}^{(f)} | \boldsymbol{\beta}_{\tau-1,j}^{(f)}\right) \right\}, \tag{5.2}$$

where $\underline{\boldsymbol{R}} = \{\boldsymbol{R}_\tau^{(f)} | \tau = 1, .., T^{(f)}, f = 1, ..., F\}$ and $\boldsymbol{B}_0 = \{\boldsymbol{B}_0^{(f)} | f = 1, .., F\}$ is the set of all prior hyperparameters for $\underline{\boldsymbol{R}}$. This model assumes conditional independence of features given the target labels $\boldsymbol{t}$. Distributions over the variables $\underline{\boldsymbol{\Pi}}$, $\underline{\boldsymbol{R}}$, $\boldsymbol{t}$ and $\boldsymbol{\kappa}$ can be inferred using the VB approach of Chapter 4. In situations such as the TREC challenge, we have continuous features for all objects, but agent responses for only a subset. Inferring distributions over the feature confusion matrices $\underline{\boldsymbol{R}}$ allows us to predict target labels for all objects, giving a measure of confidence that accounts for uncertainty over the feature confusion matrices. As the next sections will show, this allows us to evaluate the utility of obtaining additional labels from agents to reduce this uncertainty.

5.3.1 TREC Results

The TREC crowdsourcing challenge was addressed through the novel application of the extended IBCC approach. This method allowed the entire corpus of $15,424$ documents to be classified with no *a priori* training labels, by combining $2,500$ crowdsourced labels (16% of the corpus) with 2000 textual features for each document. The crowdsourced labels were collected from Amazon Mechanical Turk (AMT). In this application, human agents completed tasks, each of which consisted of a reading a document, then determining a label, which was either one of ten search queries or the option "none of the above". The details of the crowdsourcing system are described in [Simpson et al., 2013].

This system also provided textual features from the documents using Latent Dirichlet Allocation (LDA) [Blei et al., 2003]. LDA infers a distribution over topics for each document according to the words it contains, so that in this implementation, each document is associated with a vector of 2000 probability values. These probability values are treated as observations $\boldsymbol{y}$ of feature values, which are combined with the crowdsourced responses, $\boldsymbol{c}$, using the IBCC variant described in the previous section.

The system was evaluated by examining 18,260 document/query pairs, which were verified by a committee as true or false matches [Smucker et al., 2012a]. Using the same set of crowdsourced labels, IBCC was compared to a two-stage naïve Bayes method [Simpson

Method	No. Labels Collected	Mean AUC	Described in
OrcVB1	2500	0.806	The current section
Orc2Stage	2500	0.774	The current section
SSEC3inclML	30312	0.914	[Nellapati et al., 2012]
UIowaS02r	3520 from crowd + 129 sets of past results	0.881	[Harris and Srinivasan, 2012]
NEUNugget12	N/A	0.748	[Bashir et al., 2012]
BUPTPRISZHS	54780	0.597	[Zhang et al., 2012]
INFLB2012	N/A	0.517	N/A
yorku12cs03	N/A	0.479	[Hu et al., 2012]

Table 5.1: AUCs for competitors in the TREC Crowdsourcing challenge. The IBCC method described in this chapter is referred to as OrcVB1, while the simpler two-stage classifier using the same crowdsourced data is labelled Orc2Stage.

et al., 2013]. The two-stage method used a training phase to learn likelihood distributions for binary features given each target class, treating the crowdsourced labels as reliable classifications. Unlabelled documents were ignored during the training step and priors were not placed over the model parameters. In the prediction phase, the two-stage method uses the feature likelihood distributions to predict the correct search query for the entire corpus. The results of IBCC and the two-stage method were also compared to the systems used by other competitors, which obtained different sets of crowdsourced responses using a variety of approaches. The results are given in [Smucker et al., 2012b] and summarised in Table 5.1. The system detailed above using IBCC is labelled *OrcVB1*, while the variant using the two-stage classifier is called *Orc2Stage*. The original publication of results [Smucker et al., 2012b] did not evaluate the AUCs for runs labelled *UIowaS02r*, *BUTPRISZHS*, *INFLB2012*, and *yorku12cs03*, as these methods produced only binary classifications.

In comparison with the two-stage aggregator, the results show the superior performance of IBCC extended to LDA features. A likely cause of this increased performance is that IBCC accounts for unreliability in the crowdsourced labels and the confusion matrices. In contrast, the two-stage classifier trains the model by assuming these labels are correct and makes predictions assuming that all confusion matrices have been confidently learnt.

OrcVB1 also outperformed several other approaches, both when using IBCC and when using the two-stage classifier, although various elements of the crowdsourcing system may

have contributed to the system's overall performance. None of the other competitors used a Bayesian decision combination method to account for uncertainty in model parameters relating to the crowd's responses or textual features. However, two competitors – SSEC3inclML and UIowaS02r – outperformed OrcVB1, but using a substantially larger amount of data. No limit was placed on the budget allowed for the competition, nor on the number of labels the crowd could provide.

SSEC3inclML [Nellapati et al., 2012] labelled every document at least once, obtaining a total of $30,312$ labels. Their intention was to obtain reliable labels by using an expert information analyst to train an in-house crowd. Machine Learning techniques analysed text features to flag up possible errors after all documents had been labelled once, so that those documents could be re-labelled.

UIowaS02r [Harris and Srinivasan, 2012], exploited relevance judgements submitted to a previous competition for the same documents and queries. First, the system ranked the documents in an estimated order of relevance by combining the rankings from 129 earlier submissions. Then, for each query, the 10 highest-ranked documents were marked as positive examples of those queries. The remaining documents were labelled iteratively in batches of 20 using crowdsourcing, in order of increasing rank. Once an entire batch had been marked irrelevant, no more batches were sent to the crowd for that search query. While $3,520$ labels were extracted from the crowd, which is approximately 40% more than for OrcVB1, a far larger number of relevance judgements were contained in the data used from the earlier competition (the exact number is not given).

The superior outcomes of SSEC3inclML and UIowaS02r most likely stem from the far larger numbers of relevance judgements used. However, training the crowd was also a key feature in SSEC3inclML, and both methods focused on labelling difficult or uncertain documents. The information learnt by IBCC could be used to select particular documents for crowdsourcing or automatically train unreliable workers, since IBCC computes confidence in the target labels $\boldsymbol{t}$ and feature confusion matrices $\underline{\boldsymbol{R}}$, and models the reliability of workers through the confusion matrices, $\underline{\boldsymbol{\Pi}}$. This would require IBCC to be run as new

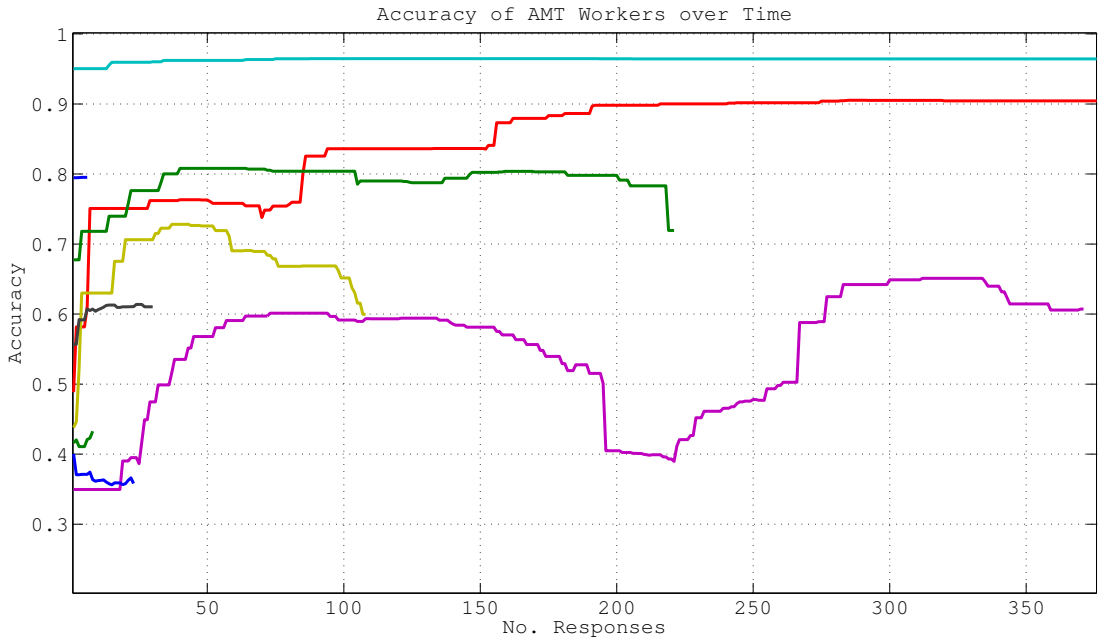


Figure 5.3: Accuracy of workers over time, as inferred by DynIBCC-VB.

labels are received from the crowd.

As discussed in Chapter 2, the error rates of the agents affects the accuracy of combined decisions. To examine the reliability of the agents, additional post-hoc analysis was performed by running DynIBCC over the final set of crowdsourced responses, given the correct decisions, to learn the dynamic confusion matrices. The confusion matrices were then summarised by a single accuracy value a_τ at each time-step, calculated as

$$a_\tau = \sum_{j=1}^{J} \left\{ \frac{\mathbb{E}[\pi_{\tau,j,j}^{(k)}]}{\sum_{l=1,l\neq j}^{L} \mathbb{E}[\pi_{\tau,j,l}^{(k)}]} \mathbb{E}[\kappa_j] \right\} \tag{5.3}$$

Figure 5.3 plots the accuracy a_τ over time, showing significant variation and changes to workers. The system used to crowdsource labels for OrcVB1 and Orc2Stage employed a simple screening step, in which workers completed ten documents, for which the correct labels were known. Workers were then employed if their accuracy on the test tasks was greater than 0.67. When IBCC was run, workers were initialised with the same values for $\boldsymbol{A}_0^{(k)}$ to give an expected accuracy of $a_0 = 0.67$. However, the post-hoc analysis inferred accuracies ranging from approximately 0.35 to 0.96. While some workers appear to have

improved over time, there are also those that deteriorate, four of which do so before they stop providing more labels. This perhaps suggests a loss of interest in the highly repetitive tasks, although a thorough investigation of AMT worker behaviour is required to determine the causes of behavioural changes.

The large variation in worker reliability shown in Figure 5.3, suggests that intelligent selection of agents is important, particularly when the budget or time is limited. The communities found within Galaxy Zoo Supernovae (see Chapter 3) previously demonstrated the large variation in agents' behaviour in a different context, while related work described in Section 5.1 also identifies problems with spammers in AMT. The varying accuracies shown in Figure 5.3 point to the need for on-going worker selection to maintain an effective pool of workers.

The remainder of the chapter therefore focuses on a theoretically-motivated intelligent tasking approach for agent selection and task assignment. Such a method should be able to make effective decisions when only a small dataset is available, as is the case at the start of the crowdsourcing process when few labels have been received from the crowd.

5.4 A Utility Function for Intelligent Tasking

Intelligent tasking is an information-theoretic approach to determining the optimal action when aggregating information in a multi-agent system. The remainder of this chapter focuses on two key problems that intelligent tasking can solve: selecting informative analysis tasks and maintaining a reliable workforce through the selection of workers. The core idea is that every action can be evaluated by a utility function that defines value in terms of information gain. Each action consists of an agent, k, performing a task, i, depending on the application. In the Galaxy Zoo Supernovae scenario, task indexes correspond to objects that must be analysed. In scenarios involving mobile agents, tasks may also include moving to observe from a particular location. Besides such information-gathering tasks, agents can also take actions that may lead to rewards in the future, such as carrying out training exercises. It is assumed that the overall goal of the information gathering exercise

is to learn the values of a set of target variables, $\boldsymbol{t}$ and that each action generates a new observation, $c_i^{(k)}$. We can define a utility function for the result of an agent k performing task i given previous responses $\boldsymbol{c}$ and object features $\boldsymbol{y}$:

$$U(k, i|\boldsymbol{c}) = \sum_{\iota=1}^{N} I(t_\iota; c_i^{(k)}|\boldsymbol{c}, \boldsymbol{y}), \tag{5.4}$$

where $I()$ refers to the Kullback-Leibler *Information Gain* [Kullback and Leibler, 1951]. Kullback-Leibler Information Gain is a suitable choice for defining our utility function because it quantifies the amount of information learned about the target decision t_ι if we can predict t_ι using $p(t_\iota|c_i^{(k)}, \boldsymbol{c}, \boldsymbol{y})$ rather than $p(t_\iota|\boldsymbol{c}, \boldsymbol{y})$. It is defined as:

$$I(t_\iota; c_i^{(k)}|\boldsymbol{c}, \boldsymbol{y}) = \sum_{j=1}^{J} p(t_\iota = j|c_i^{(k)}, \boldsymbol{c}, \boldsymbol{y}) \ln \left(\frac{p(t_\iota = j|c_i^{(k)}, \boldsymbol{c}, \boldsymbol{y})}{p(t_\iota = j|\boldsymbol{c}, \boldsymbol{y})} \right). \tag{5.5}$$

If the logarithms used in this equation are base e, the information gain is measured in nats, and if base 2 is used, the units are bits. Hence we can measure the information obtained from an agent and compare this quantity to the information provided by other agents and by other labelling tasks. Kullback-Leibler Information Gain also depends on the terms in the conditions $\boldsymbol{c}, \boldsymbol{y}$, i.e. what we already know, so that information is only valued if it is complementary to what we have already learned.

The true value of the utility function can only become known once we observe the value of $c_i^{(k)}$. *Intelligent tasking* therefore refers to any algorithm that directs agents to tasks that maximise the *expected* utility, which can be derived by considering that information gain

can also be defined in terms of entropy reduction:

$$\begin{aligned}
\hat{U}(k,i|\boldsymbol{c}) &= \sum_{\iota=1}^{N} \mathbb{E}_{c_i^{(k)}}[I(t_\iota; c_i^{(k)}|\boldsymbol{c},\boldsymbol{y})] \\
&= \sum_{\iota=1}^{N} H(t_\iota|\boldsymbol{c},\boldsymbol{y}) - \sum_{c_i^{(k)}=1}^{L} p(c_i^{(k)}=l|\boldsymbol{c},\boldsymbol{y}) H(t_\iota|\boldsymbol{c},\boldsymbol{y},c_i^{(k)}=l) \\
&= \sum_{\iota=1}^{N} \left\{ \sum_{j=1}^{J} \sum_{c_i^{(k)}=1}^{L} p(t_\iota=j, c_i^{(k)}=l|\boldsymbol{c},\boldsymbol{y}) \ln p(t_\iota=j|\boldsymbol{c},\boldsymbol{y},c_i^{(k)}=l) \right. \\
&\quad \left. - \sum_{j=1}^{J} p(t_\iota=j|\boldsymbol{c},\boldsymbol{y}) \ln p(t_\iota=j|\boldsymbol{c},\boldsymbol{y}) \right\}, \qquad (5.6)
\end{aligned}$$

where $H(x)$ is the Shannon entropy, which evaluates the uncertainty of a random variable x by giving the amount of information learned about x by observing its value. When logarithms are taken to base e, the entropy is measured in nats. The expected information gain is therefore the expectation with respect to the label $c_i^{(k)}$ of the reduction in entropy of the target decision t_ι. This expected information gain can also be referred to as the *mutual information* between a response $c_i^{(k)}$ and a target decision t_ι.

If we take the decision that maximises the expected utility $\hat{U}(k,i|\boldsymbol{c})$, we are also minimising a loss function that is the negative of the utility function. The Bayesian decision rule means we take the decision that minimises the expected loss, which is an *admissible* decision rule, i.e. there is no better decision given our loss function and current knowledge [Berger, 1985]. Thus, given that we have defined utility according to Equation (5.4), the optimal action is to choose the agent-task pair that maximises Equation (5.6). However, Equation (5.4) is a *greedy* utility function, i.e. one that considers only the immediate utility of the next action taken by an agent. If we use the greedy utility function to assign agents to tasks iteratively as each task is completed, we are operating with a greedy strategy. Such a greedy strategy is sub-optimal, meaning that it may not result in the maximum reduction in uncertainty in the target decisions over multiple iterations. This sub-optimality occurs because the utility function does not consider how the immediate response will affect later decisions, nor how future observations might affect the current choice. Therefore, using the

greedy strategy to select objects for an agent to analyse will optimise the only the utility of the current assignment, rather than future assignments. However, it leads to far more scalable algorithms and has been shown to be approximately as good as the optimal algorithm for minimising the number of labels required in an *Active Learning* scenario [Dasgupta, 2004]. In applications such as citizen science, it may be necessary to propose several tasks for an agent, since control over the agents is limited to the ability to make suggestions, which may be rejected.

The terms in Equation (5.6) can be obtained by learning the DynIBCC model, or indeed any other Bayesian decision combination model. It is important to use a model that accounts for uncertainty in the model parameters, otherwise we will underestimate the entropy in the target decisions, $H(t_\iota|\boldsymbol{c}, \boldsymbol{y})$, so that the information gain predictions are not meaningful. The term $p(t_\iota = j|\boldsymbol{c}, \boldsymbol{y})$ can be estimated by running the DynIBCC algorithm with the current set of observations. The terms $p(t_\iota = j|\boldsymbol{c}, \boldsymbol{y}, c_i^{(k)} = l)$ are calculated by re-running the algorithm for each possible value of $c_i^{(k)} = l$, which is added as a simulated observation to the current observations. To encourage rapid convergence of the estimates for $p(t_\iota = j|\boldsymbol{c}, \boldsymbol{y}, c_i^{(k)} = l)$, we can initialise all variables relating to model parameters $\underline{\boldsymbol{\Pi}}$, $\underline{\boldsymbol{R}}$ and $\boldsymbol{\kappa}$ to their final values from the earlier run of the algorithm used to estimate $p(t_\iota = j|\boldsymbol{c}, \boldsymbol{y})$. If the addition of a single crowdsourced label $c_i^{(k)}$ causes only small changes to the distribution over the target decisions $\boldsymbol{t}$, then the algorithm in Section 5.3 will require very few iterations to converge. Section 5.5 explains further how we can use Equation (5.6) to develop a practical method for selecting agents and analysis tasks in a crowdsourcing application.

5.4.1 Exploitation and Exploration

This section considers some important properties of Equation (5.6), which defines the expected utility of obtaining a label $c_i^{(k)}$ from agent k for target decision i. This utility function naturally balances the value of both *exploitation* and *exploration* of the model. Exploitation refers to using the current model to to learn target decisions $\boldsymbol{t}$ from new

crowdsourced responses, while exploration means learning the model itself. When using DynIBCC, exploration involves learning the confusion matrices $\underline{\mathbf{\Pi}}$ that describe the agents' behaviour. Chapter 6 extends the exploitation/exploration trade-off to consider improving the behaviour of agents, e.g. through training exercises. To see the balance between exploration and exploitation, consider two extreme scenarios where we take a new label $c_i^{(k)}$ for object i:

1. Intelligent tasking *exploits* the model. As uncertainty over an agent's reliability decreases, i.e. $H(\boldsymbol{\pi}_j^{(k)}) \rightarrow 0$, the amount we can learn from a new label $c_i^{(k)}$ about objects previously classified by agent k decreases. So we tend to choose an object i where the target label t_i is uncertain. A numerical example of the agent in this scenario is agent $k = 1$ in Table 5.2, which would be assigned to object $i = 1$.

2. Intelligent tasking *explores* the model. As the uncertainty over the agent's reliability increases, i.e. $H(\boldsymbol{\pi}_j^{(k)}) \rightarrow \infty$, the amount we can learn about target label t_i decreases, so we choose an object i where the target label t_i has been learned with high confidence (a *silver-labelled task*), or is known for certain (a *gold-labelled task*). Choosing a gold or silver-labelled task will increase the information gain over objects previously classified by agent k. In Table 5.2, assigning object $i = 3$ to agent $k = 2$ is an example of silver tasking. Silver tasking is an alternative to approaches that insert *gold tasks* for which the ground truth or expert label is known. Silver tasks avoid the need for expert labelling, and information gain provides an automatic method of selecting these tasks when appropriate.

The Bayesian decision rule therefore avoids the need for any explicit exploitation/exploration parameters, as the balance arises naturally from Equation 5.6. The next section develops an intelligent tasking algorithm that simultaneously tackles the problem of maintaining a reliable workforce, while selecting informative analysis tasks.

	$i=1$	$i=2$	$i=3$
$k=1$	20	10	≈ 0
$k=2$	≈ 0	≈ 0	4.7
$k=3$	≈ 0	≈ 0	≈ 0

(a) Expected utility of assigning each object to each agent.

Obj. ID	$i=1$	$i=2$	$i=3$
	Unlabelled, uncertain	Incorrectly labelled	High certainty
$H(\boldsymbol{t}_i)$	2.3	2.3	0.5448

(b) Entropy of target labels for objects

Agent ID	$k=1$	$k=2$	$k=3$
	Reliable, certain confusion matrix	Uncertain confusion matrix	Unreliable, certain confusion matrix
$H(\boldsymbol{\pi}^{(k)})$	0.6	12.8	1.2

(c) Entropy of agents' confusion matrices

Table 5.2: Numerical example of utility.

5.5 Hiring and Firing for Crowdsourcing

This section develops an intelligent tasking algorithm suitable for task and agent selection in the TREC crowdsourcing scenario. The approach lays the foundations for more comprehensive intelligent tasking algorithms that tackle additional decision-making problems in multi-agent systems, such as training and motivating people. As a basis, the aim is to select task assignments that maximise the expected utility given in Equation 5.6, in order to learn a set of target decisions with confidence using a minimal number of crowdsourced responses. This section begins by outlining a number of assumptions that allow us to develop a tractable intelligent tasking algorithm for applications such as the crowdsourcing case study considered earlier in this chapter.

The first assumption is that multiple tasks can be carried out concurrently by different agents. While it may seem preferable to use only the best agent available, in practice this agent is unknown and it is desirable to use several agents to obtain responses more quickly to meet time constraints. When few gold labels are available, observing multiple agents improves the model's ability to distinguish reliable workers, since agents that agree are less

likely to be guessing answers at random. The algorithm proposed in this section therefore assumes a fixed pool size, N_{poolsize}, which is the number of workers currently employed.

The second assumption is that after a worker completes a task, they can either be re-hired immediately or fired permanently. This arises because if there is a delay in presenting new tasks, workers on platforms such as Amazon Mechanical Turk (AMT) are likely to find an alternative application to work on, so cannot be re-hired after a period of no tasks.

The final assumption is that new workers are always available to be hired to replace fired workers or those that choose to leave, and that there is no penalty when replacing workers. This assumption is suited to large crowdsourcing platforms such as AMT, where a very large number of agents are easily accessible at any time.

Given the above assumptions, we can specify an algorithm that maintains a pool of trustworthy workers by replacing those who are uninformative with new workers. The algorithm should aim to make the optimal decision each time a new response from a worker is observed: either hire the worker to perform their optimal task, or fire the worker and hire a new worker to perform the optimal task for a new worker. This requires evaluating the utility of object-worker pairs including either the available current worker or the new worker. Thus, the above assumptions constrain the space of possible task assignments. The utility function defined by Equation 5.6 is used in the experiments below. The *Hiring and Firing* algorithm for intelligent task assignment is given below.

1. Initialise the set of hired workers, $\boldsymbol{W} = \emptyset$, and idle hired workers, $\boldsymbol{W}_{\text{idle}} = \emptyset$.
2. Run DynIBCC over current set of data, $\{\boldsymbol{c}, \boldsymbol{y}\}$ to obtain probabilities of labels for all objects. Initially, crowd responses $\boldsymbol{c}$ are empty, and we only see features $\boldsymbol{y}$.
3. Calculate $\hat{U}(k, i|\mathbf{c}, \mathbf{y})$ for all tasks, i, and all available workers, $k \in \boldsymbol{W}_{\text{idle}}$ and for an unknown new worker, u.
4. Set $N_{\text{toassign}} = N_{\text{poolsize}} - |\boldsymbol{W}| + |\boldsymbol{W}_{\text{idle}}|$, where N_{toassign} is the number of workers to assign, and N_{poolsize} is the desired worker pool size. The number to assign is therefore the shortfall in the current worker pool plus the number of idle workers.
5. While $N_{\text{toassign}} > 0$:

(a) Where $k \in \{W_{\text{idle}}, u\}$ is any worker, including the unknown new worker, u, choose the assignment (k, i) that maximises the expected utility, $(k, i) = \underset{k,i}{\operatorname{argmax}} \hat{U}(k, i|\mathbf{c}, \mathbf{y})$. The chosen worker is *hired* to do task i. Do not consider any tasks that are currently being completed by other workers, as the other responses are likely to significantly reduce the utility of any repeated assignments.

(b) Remove i from the list of possible task assignments for this iteration to avoid repeating the task.

(c) If k is not the unknown worker, remove k from W_{idle} as they are no longer idle.

(d) Set $N_{\text{toassign}} = N_{\text{toassign}} - 1$.

6. Any workers remaining in $\boldsymbol{W}_{\text{idle}}$ are *fired* and removed from $\boldsymbol{W}$ and $\boldsymbol{W}_{\text{idle}}$

7. Send the selected task/worker pairs to the crowdsourcing system for agents to complete in parallel; await responses.

   (a) Any workers who have not been fired can complete tasks assigned to u, and are then added to the pool, $\boldsymbol{W}$.

   (b) A time-out occurs if a task is not completed within a specified period. The assigned worker is then removed from $\boldsymbol{W}$, and the process is repeated from Step 2 to hire a replacement worker.

   (c) On receiving a new label from worker k, add k to $\boldsymbol{W}_{\text{idle}}$, then repeat from Step 2.

In Step 7c, the first iteration will result in $\boldsymbol{W}_{\text{idle}}$ containing only the first agent to complete their task. In subsequent iterations, multiple agents could have been added to $\boldsymbol{W}_{\text{idle}}$ while the other steps of the algorithm were being computed. A delay in Step 7c before repeating could be added to wait for more idle agents before the algorithm is repeated, but the delay should not be long enough to dissuade agents from completing more tasks. Alternatively, multiple iterations of the algorithm could run in parallel as each agent completes a task,

so that $\boldsymbol{W}_{\text{idle}}$ typically contains only one agent. Assuming that each assignment is made using all labels currently available from agents, and the best agent/task pair is chosen from those currently available, the algorithm is the locally-optimal *greedy* approach. That is, we cannot improve the expected utility of the next assignment by using any other decision rule. This process combines the screening of agents with selection of informative tasks, avoiding the need for a separate method to test and screen agents periodically.

For the unknown new worker u, we use prior distributions over each row of the confusion matrix $\boldsymbol{\pi}_j^{(u)}$ to calculate $\hat{U}(u, i|\mathbf{c}, \mathbf{y})$. The priors are set by observing the performance of agents in the same crowdsourcing system on a previous set of documents, and taking a mean of their response counts. The magnitude of the counts is reduced so that the variance of $\boldsymbol{\pi}_j^{(u)}$ matches the sample variance for the observed agents.

When dealing with large numbers of objects, the computational cost of Step 3 can be reduced by considering only a subset. For a fair comparison between agents, the same subset should be used for all agents in one iteration. In theory, intelligently choosing objects from a random subset would improve over selecting entirely at random. However, it is possible to select in a more informed way by considering similarities between objects. Assuming that similar objects produce similar utility, we can search for objects that are close to optimal by sampling the utility function at different points in feature space. Samples can be chosen by grouping similar objects, then selecting from each cluster at random. The experiments below use K-means clustering [Bishop, 2006] with $k = N/25$, then select one object at random from each cluster. This approach explores the feature space while avoiding redundant comparisons of highly similar object-worker pairs.

5.5.1 Online Screening Method

The Hiring and Firing algorithm is compared to a simpler screening method, similar to that proposed by [Donmez et al., 2010]. In this method, referred to as *online screening*, the accuracy of workers' responses is tracked dynamically, and workers are fired when their accuracy drops below a certain threshold. This can be seen as a simplification of the hiring

and firing algorithm, in which the approximate utility is replaced by accuracy, independent of task. Workers are compared against the unknown worker, whose accuracy is determined from the prior confusion matrices, so is in effect a fixed threshold. If a worker is hired, the task is chosen at random. Comparing against the online screening approach highlights the advantage of selecting informative tasks for specific workers.

5.6 Hiring and Firing Experiments

The Hiring and Firing algorithm is compared with four other methods using simulated agents on 600 documents from the TREC crowdsourcing dataset. Of the documents selected, 37 belong to topic 427 from the TREC8 dataset, while the rest are randomly selected from documents that were not marked as relevant to the above topic. This experiment combines the same LDA features used in Section 5.3.1 with the simulated agent responses. The experiment was repeated over 20 datasets, each including different irrelevant documents; each algorithm was run once over all of the datasets.

5.6.1 Simulated Agents

This experiment used simulated workers so that equivalent behaviour can be replicated for each of the algorithms tested. As with the TREC crowdsourcing scenario, agents are assigned documents by a centralised decision maker, and label them as relevant to topic 427 or not relevant. The agents' responses are drawn from a categorical distribution with a given accuracy. As new agents are hired, they are randomly assigned an initial accuracy of 0.95, 0.8, or 0.5. The initial accuracy cycles through these values as new agents are generated. Thus the hired agents have mixed reliability from very accurate to uninformative. The ideal performance of the algorithms is to fire all but the most reliable workers.

To test the ability of the algorithms to deal with deterioration in behaviour, the agents switch abruptly to an uninformative mode after between 10 and 25 iterations. In the uninformative mode, the correct and incorrect target labels are chosen at random. This shift

represents agents changing their behaviour in an attempt to game the system, becoming bored and clicking answers at random; it is also similar to the situation where a physical agent or sensor moves and can no longer observe the target object.

The pool size is set to 5 workers. For each run, 10 initial responses are drawn for each worker for randomly chosen documents, and the same set of initial responses is supplied to bootstrap all the algorithms tested. These initial responses are theoretically not required to run the Hiring and Firing algorithm or the alternative methods, but saves the computation time of running the algorithms while little information is available to make informed decisions.

5.6.2 Alternative Methods

Method Name	Worker Model	Active Selection?	Hiring and Firing?
HF	DynIBCC	Yes	Yes
HFStatic	Static IBCC	Yes	Yes
AS	DynIBCC	Yes	No
OS	DynIBCC	No, random assignment	Yes
Random	DynIBCC	No, random assignment	No

Table 5.3: Features of methods tested for selecting workers and tasks.

The Hiring and Firing algorithm (HF) was compared with online screening (OS), random task selection with no firing (Random), active selection with no firing (AS), and Hiring and Firing using a static worker model (HFStatic). The AS method assigns documents to workers intelligently using the same utility function as Hiring and Firing. However, all original workers are kept on the books and no new workers are recruited. HFStatic uses the static variant of IBCC to combine worker responses with text features. Table 5.3 is an overview of the properties of each algorithm. The controlled conditions of the experiment were intended to show the benefits of each property of the complete Hiring and Firing algorithm: the ability to track changing performance; intelligent task selection; and choosing new workers when current agents are not informative.

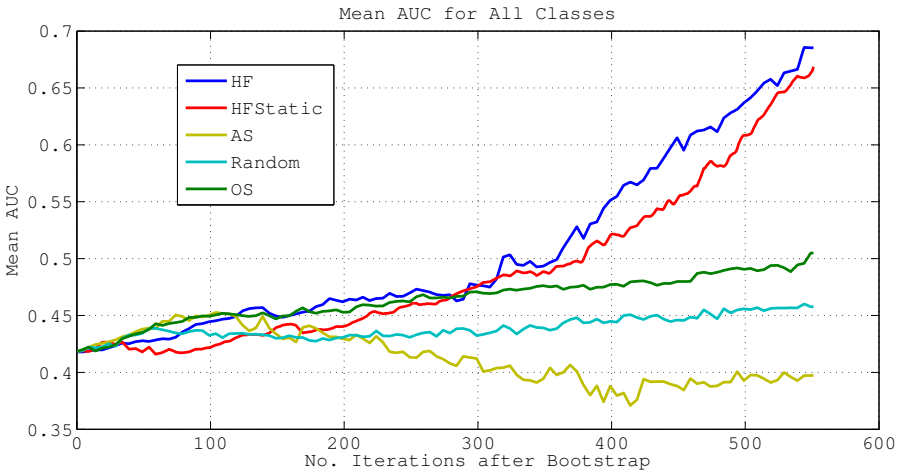
5.6.3 Results with TREC Documents

Each time new responses were obtained from the simulated crowd, DynIBCC was run to update the combined class decisions (for HFStatic, static IBCC is used instead). The performance was then measured at each iteration by calculating the AUC of the combined results.

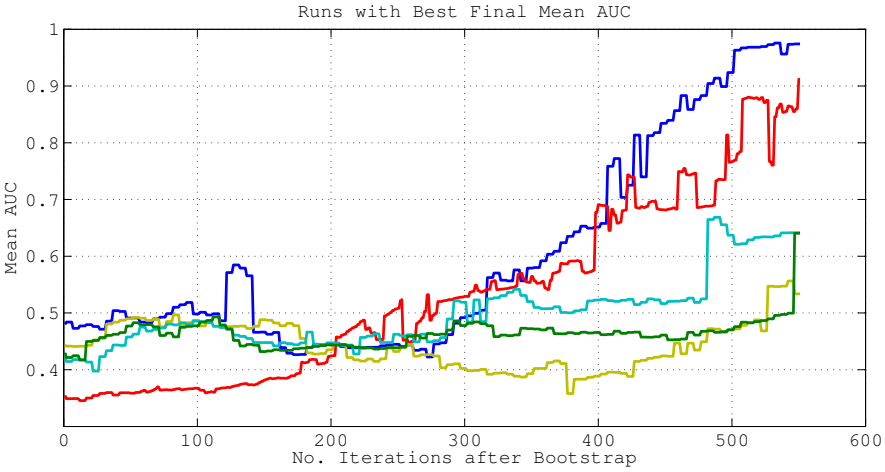
Figure 5.4a shows the mean AUC over 20 runs as a function of iterations for the methods in Table 5.3. The HF method has the best performance with a final mean AUC of 0.69, compared to its nearest competitor, the static variant of HF, with 0.67. These two are significantly better than for OS, which does not actively select tasks, with 0.51. Note that for a long period, the mean AUC of all methods is below 0.5 and a lot of time is spent recovering from this position. A difficulty in this experiment is that there were only 37 relevant documents and 600 responses from the simulated crowd, but 2000 LDA features.

After 125 iterations, none of the original set of agents is informative. Examining the mean AUCs in Figure 5.4a, the continuing improvement of HF and HFStatic after 125 iterations shows that they must have fired and hired new agents. This contrasts with AS, which does not improve after the agents become uninformative. OS also diverges from HF and HFStatic at 300 iterations, but continues to increase gradually. The Random method diverges from HF and OS around 70 iterations, when some agents start to become uninformative. The AS and Random methods stagnate after a period of time, as they are unable to fire agents and the entire pool eventually becomes uninformative. After 125 labels, all of the original agents are uninformative and AS and Random cannot attain a high AUC. Note that while Random moves closer to 0.5, i.e. expressing complete uncertainty, the AS method decreases to below 0.4 for a period.

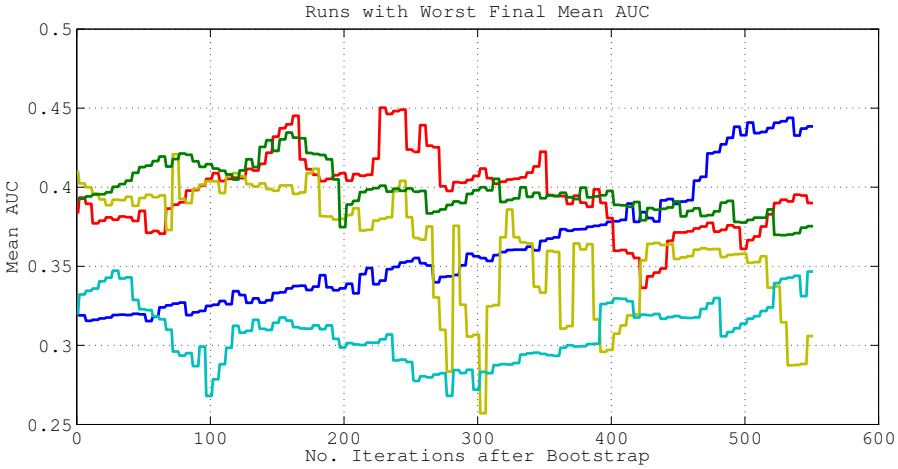
In Figure 5.4b we show the best individual run for each method, while Figure 5.4c shows the worst. This highlights the significant differences in performance between runs. In its best run, HF reaches 0.98 AUC, which follows from a starting AUC close to random around 0.5. In contrast, the worst performance starts with a much lower AUC, near to 0.3, indicating that the bootstrap labels contained a number of errors that result in the model



(a) AUC is averaged over 20 runs.



(b) Best individual runs for each method.



(c) Worst individual runs for each method.

Figure 5.4: Using LDA document features from TREC. AUC as a function of the number of labels received from workers.

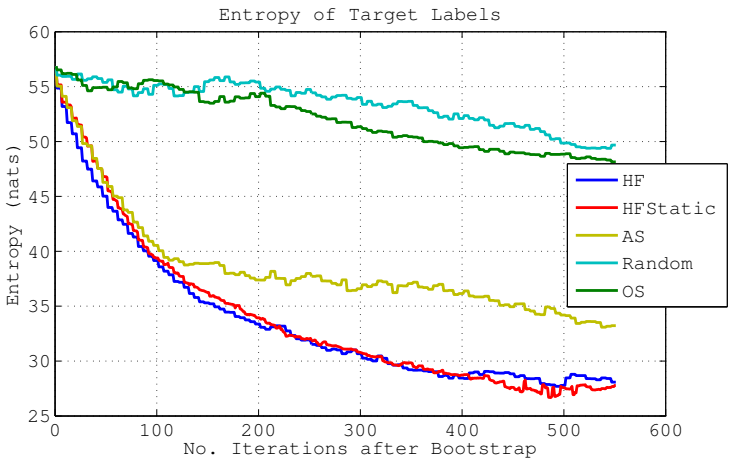


Figure 5.5: Using LDA document features from TREC. Entropy of the target labels as a function of the number of labels received. Entropy is averaged over 20 runs.

producing the reverse of the correct decisions. The worst-case AUC for HF increases steadily, in contrast to the other methods, which do not show a clear increase in the worst case within 550 iterations. Decreases in the AUC for HFStatic, OS and AS suggest that the responses are consolidating an incorrect model.

Figure 5.5 shows the mean entropy of the document labels $\boldsymbol{t}$. The entropy H is important in active learning problems such as this, as it reflects the confidence in the predictions obtained at each iteration. When workers become uninformative and issue random labels, the new unreliable data causes the entropy to rise. All methods see continued decreases in entropy, with HF and HFStatic improving most rapidly. For some runs, the AUCs for AS continued to decrease after the entire agent pool was uninformative; however, the entropy stops decreasing rapidly after 125 iterations, at the point where none of the new crowd responses obtained by AS are informative.

5.6.4 Synthetic Dataset

In a separate experiment, the methods HF, HFStatic, OS and Random were re-run over synthetic features to explore whether the LDA features themselves contributed to the variations in performance over multiple runs. It is possible that for some datasets, there were too few features that had sufficient correlation with the target labels. With many unreliable

labels and few relevant documents, it is also possible that clusters of negative documents could be identified as the positive group. Synthetic features were drawn from Beta distributions to ensure that the only latent structure in the features related to the target labels. Documents could be relevant to one of three search queries or to none. For each query, there were 15 features with a high probability of values close to one. The remaining 205 features were drawn at random, independent of the query relevance. Hence the features had weak but known correlation with the search queries.

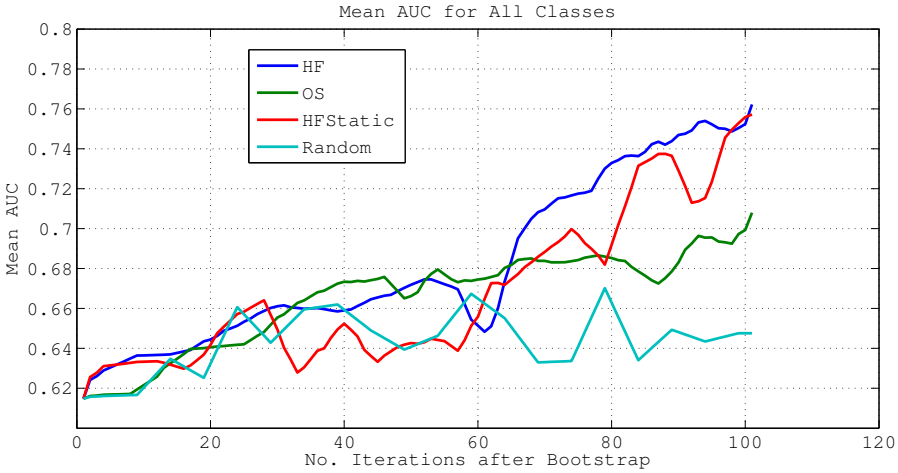
Figure 5.6 shows the mean AUCs over 10 runs. Similar patterns are observed for HF and HFStatic as with the LDA features. In the best and worst cases, HFStatic produced better results than HF, although it was worse on average. OS is less extreme but continues to be overtaken by both Hiring and Firing methods. HF therefore produced the most consistent results.

Figure 5.7 gives an example of the hiring and firing process in HF. The plot shows the approximate utility $\hat{U}(k, i)$ of the optimal task i for three example the workers. Workers $k = 1$ and $k = 2$ are reliable throughout the experiment and are hired throughout by all methods. Worker $k = 3$ appears to become gradually less informative until being fired by HF at time step 87. The gradual nature of the change is likely to be because the model requires a number of observations to become certain about the unreliability of the agent.

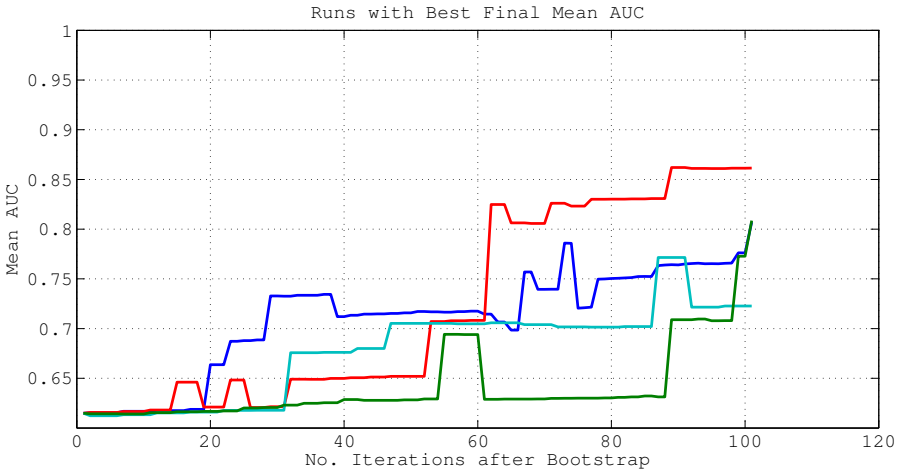
5.6.5 Summary of Results

The Hiring and Firing algorithm is the first iteration of the Intelligent Tasking approach, and these simulated experiments demonstrate its advantages over more basic alternatives. HF gains a significant improvement over alternatives through intelligent task selection, hiring new agents effectively, and responding quickly to agent dynamics.

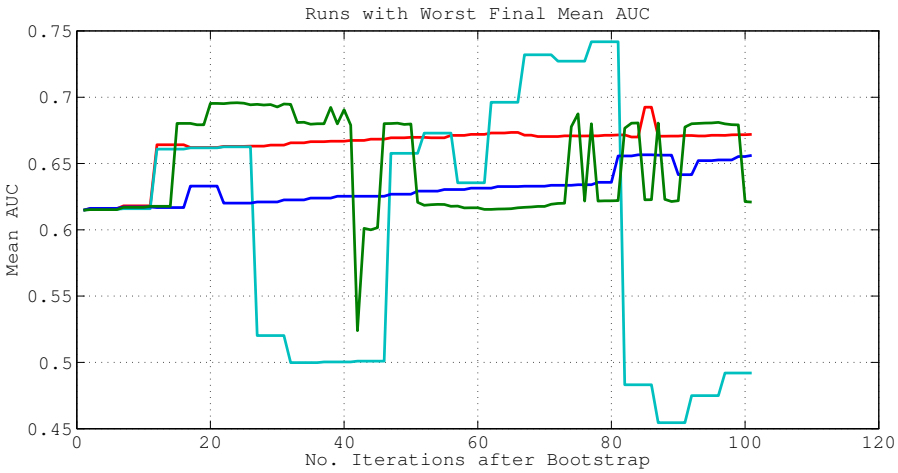
HF and HFStatic have sharper increases in the mean AUCs compared to the OS method, although the latter must replace some unreliable agents since it continues to improve gradually. The different behaviour may result from the hiring and firing algorithms selecting documents intelligently, or from better filtering of uninformative agents. Unlike OS, all



(a) Mean AUC over 10 repetitions and over the three topic labels.



(b) Using synthetic features. Best individual runs for each method, mean over three topic labels.



(c) Worst individual runs for each method, mean over three topic labels.

Figure 5.6: Synthetic features. Mean AUC as a function of the number of responses received. AUC is averaged over 10 repetitions and over the three topic labels.

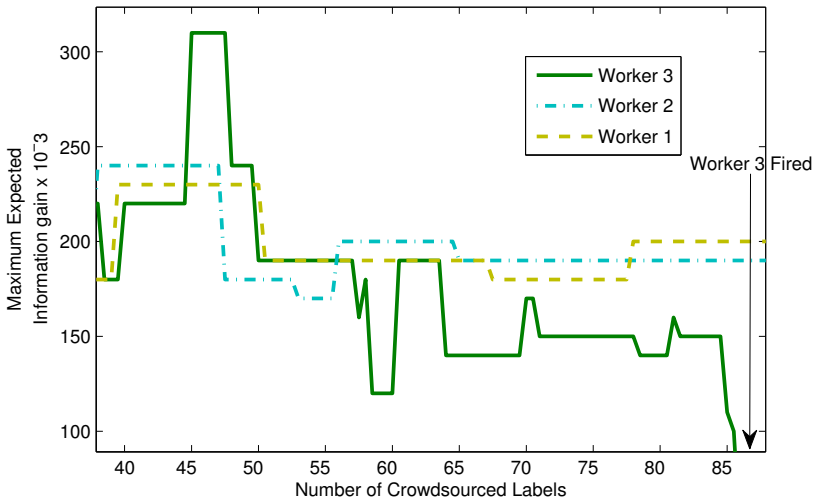


Figure 5.7: Maximum expected information gain for each of three example workers.

three methods using active document selection, i.e. HF, HFStatic and AS, have rapid decreases in entropy, indicating that we obtain a more confident model by selecting tasks intelligently using our expected utility function (see Equation 5.6). A further advantage of HF over OS may be more clearly realised in a scenario where agents have different skill levels for different types of task. The HF approach uses the full confusion matrix to evaluate agents, rather than the single reliability value used by OS. This enables the system to discriminate between agents with similar overall accuracy but different behaviours and therefore unequal utility.

In the average and best cases, HFStatic also improves throughout the experiment, but more slowly than fully dynamic HF. Since the model assumes agents are static, agents that have become uninformative will not be detected until their average confusion matrix over all submitted tasks is worse than that of the unknown agent. In contrast, DynIBCC is able to detect changes rapidly, as shown in Chapter 4. The worst case for HFStatic with TREC data (Figure 5.4a) shows the AUC declining over time, which may represent a complete failure to fire uninformative agents. The inclusion of agent dynamics in the DynIBCC model appears to produce more reliable and timely intelligent tasking decisions.

HF reduces entropy at a comparable rate to HFStatic. However, HF uses the DynIBCC

model, which has more degrees of freedom than the static IBCC model used by HFStatic, so we might expect it to be more difficult to learn with a high degree of confidence. These results suggests that the more complex DynIBCC model can be learned equally fast as the static model in practical applications.

The variation between best and worst cases suggests that the initial set of responses is critical, particularly in this case where one class occurs much more rarely. It may be possible to improve the hiring and firing algorithm by by selecting the first set of tasks intelligently. To find positive examples more quickly in real-world crowdsourcing systems, we could also introduce weak prior information about the features, for example, by looking at the relationship between features and keywords in the search query. This would allow the intelligent tasking method to select initial sets of documents for crowdsourcing that are more likely to be relevant.

When using the synthetic dataset with 250 features, the differences in performance between each run were less extreme than with 2000 LDA features. This highlights the importance of extracting useful features *a priori*, especially in the absence of training data.

Further experiments with real agents in a crowdsourcing system are needed to test the performance differences with real behavioural changes and different pool sizes. Ideally, the experiments should be expanded to larger numbers of target values (e.g. more search queries) to better compare the use of confusion matrices with single accuracy measures, such as that used in the OS method. However, to move beyond the proof-of-concept proposed here, it is important to improve scalability and address a number of other limitations, which are discussed below.

5.6.6 Discussion

Computation time is a significant obstacle that may require more drastic approximations if intelligent tasking is to be applied to larger datasets or to account for future utility. At each iteration, the number of comparisons grows with the number of possible agent-task pairs, but the cost of each comparison also grows with larger problems. First, the number of

DynIBCC runs grows linearly with the number of target values (search queries in TREC). Each DynIBCC-VB run consists of a number of iterations, the complexity of which is difficult to describe, partly because it depends on the initial values. With a single new response it is possible to restart the algorithm, adding the new response to the previous data, and since a single update is unlikely to change the variables significantly, we expect to run only a small number of iterations. A single DynIBCC-VB iteration is linear in the total number of agents' decisions plus the number of features multiplied by the number of objects. In the experiments above, documents were clustered to reduce computational cost, which fixes the number of pairs to compare, but does not address the scalability of DynIBCC iterations themselves. This may benefit from further approximating each DynIBCC update, or from ideas discussed in the next chapter, including decentralising computation.

The priors over the confusion matrices provide a threshold for hiring or firing workers, which is fixed before starting the algorithm according to our prior knowledge of similar workers. In future, this prior could be adapted as we observe more workers completing the current set of tasks, reducing the need to obtain data to set informative priors when running a new application. A pooling method for updating priors is discussed in the next chapter.

The document clustering step currently uses a fixed number of clusters, chosen to limit the computation time of each iteration of Hiring and Firing. In future the choice of number of clusters could be treated as a meta-decision, which could be optimised by weighing the expected information gain from using more clusters against the expected time cost and risk of losing workers.

This chapter focused on the efficient deployment of agents for decision-making tasks through the use of descriptive feature data and weak control. First, the potential of DynIBCC to combine continuous-valued features and human agents' responses was shown, demonstrating how this enables efficient analysis of a large dataset in the absence of training examples. Then, an information-theoretic viewpoint was taken to enable the intelligent assignment of tasks to agents in multi-agent systems, resulting in the Hiring and Firing al-

gorithm for crowdsourcing applications such as the TREC challenge. This algorithm was shown to select tasks and workers effectively, outperforming more simplistic approaches. The following chapter considers future extensions to intelligent tasking that predict the value of improvements to agents through training and incentives.

Chapter 6

Future Work and Conclusions

This thesis presents a robust, Bayesian framework for aggregating unreliable decisions from humans, software agents and sensors with changing behaviours. Consider the challenges described in Chapter 1 for combined decision making in multi-agent systems. The issues of variation among agents and uncertainty over their reliability are handled in a principled manner by the Independent Bayesian Classifier Combination (IBCC) model. The experiments in Chapters 2 and 3 demonstrate that this approach, using confusion matrices to provide an expressive model of agent behaviour, outperforms many established alternative methods, both on real-world and simulated data. The variational inference algorithm for IBCC introduced in Chapter 3 vastly improves the scalability of IBCC, allowing its application to large datasets and real-time updates. The issue of dynamics is addressed by a novel method, DynIBCC, proposed in chapter 4 for tracking changing agent behaviour. Despite the additional complexity of this model, it is able to produce accurate combined decisions as well as detecting changes in agents in both simulated and real data. DynIBCC can be used to supplement agents' responses with continuous feature data, allowing unsupervised learning of target decisions for objects not labelled by the agents, as shown in Chapter 5. The intelligent tasking approach described in Chapter 5 provides a method for weak control of agents by suggesting for informative tasks, using DynIBCC to measure the information gain of particular responses. This approach is applied by the Hiring and Firing algorithm, which uses DynIBCC to detect when agents improve or deteriorate, then

determines suitable tasks for each agent and whether to continue to hire them. Many opportunities exist to develop methods for combined decision making and intelligent tasking to address new problems and provide more scalable algorithms. This chapter now presents some of the promising new directions for this work. Finally, this thesis concludes by explaining the limits to the methods proposed here, and to decision making with multiple agents in general.

6.1 Sharing Agent Information

Knowledge about one agent's behaviour may inform us about the behaviour of similar agents. For example, if we observe that two agents previously produced similar responses, they may continue to do so when we introduce new types of task. Responses from one agent for a new task type therefore inform our priors over the behaviour of the other agent for this new task type. In a similar manner, observations of a pool of current of agents can inform priors over new agents. In this case, new agents are assumed to be drawn from a similar distribution to the current crowd. Such adaptive priors could improve the Hiring and Firing algorithm in cases where the nature of the crowd is not well known *a priori*.

As well as the relationships between agents, we can also consider the similarities between agents' behaviour when presented with different types of task or target label values. For example, when matching documents to queries (the target labels), similar queries may elicit similar responses and features. Alternatively, we may introduce an entirely new task, where the current agents must evaluate a different set of queries; agents with high accuracy in the previous task would be expected to continue to produce accurate answers.

The challenge is therefore to determine methods for sharing information relating to similar entities. In general, such sharing is useful when limited data is available for some individuals. In DynIBCC, an agent's behaviour for a particular target label $t_i = j$, within a single task, is described by a row of the confusion matrix, $\boldsymbol{\pi}_j^{(k)}$. Conceptually, we can think of adding new types of task by adding rows to the matrix for each new task/target label pair, and columns for each possible new response, where the priors of invalid combinations are

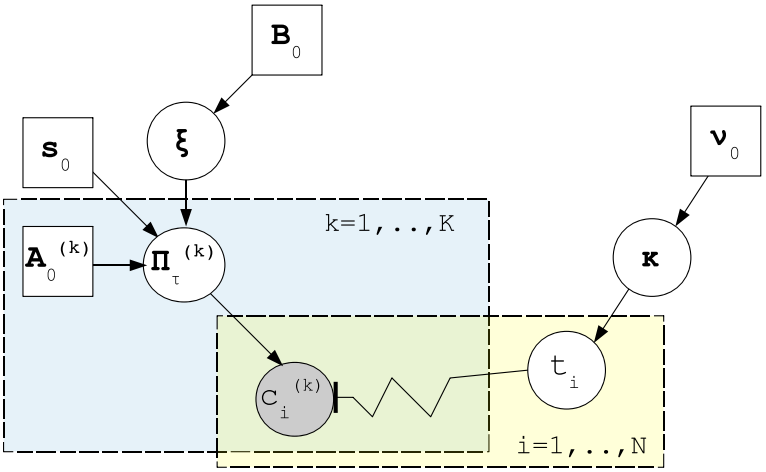


Figure 6.1: A revised graphical model for static IBCC where the agents' confusion matrices are generated from a pooled prior. The variable $A_0^{(k)}$ includes any prior knowledge of differences between the individuals as pseudo-counts.

set to zero. Currently, when an agent's response $c_i^{(k)}$ is observed, it is only used to update the distribution over a single row, $\boldsymbol{\pi}_j^{(k)}$, by adding to the hyperparameters $\boldsymbol{\alpha}_j^{(k)}$. However, these observations could also update distributions for other agents related to k or other rows in the same confusion matrix related to j through target label or task similarity.

Information can be shared through *pooled priors*, using methods such as the Dirichlet-Dirichlet model proposed by [Leonard, 1977]. This method deals with sharing data between multinomial distributions under the assumption that parameters of different distributions are exchangeable. It assumes that multinomial distributions indexed by $j = 1, .., J$ are related in a hierarchical manner. First, consider that every multinomial j has a common Dirichlet prior with parameters $[\alpha_{0,1}, .., \alpha_{0,L}] = [s\xi_1, .., s\xi_L]$, where $s_0 = \sum_{l=1}^{L} \alpha_{0,l}$ controls the precision and $\boldsymbol{\xi}$ the mean, and L is the number of components in the distribution, for example the number of possible response labels that an agent can choose from. Then, assume that parameter vector $\boldsymbol{\xi}$ is itself drawn from a Dirichlet distribution with priors $\boldsymbol{\beta}_0$. The precision s corresponds to the strength of prior belief that all multinomial distributions $j = 1, .., J$ are equal. This parameter therefore corresponds to the idea of similarity between different agents or target labels, but is uniform for all of the pooled distributions in the model suggested by [Leonard, 1977]. A graphical model for IBCC with pooled priors

between agents is shown in Figure 6.1. The differences between the multinomial distributions $j = 1, .., J$ are encoded by updating the prior over the parameter vector $\boldsymbol{\pi}_j^{(k)}$ for each j with additional pseudo-count vectors $\boldsymbol{N}_j$, which can represent both observed counts or additional prior knowledge about $\boldsymbol{\pi}_j^{(k)}$.

To obtain the posterior hyperparameters for distribution j, we first update the pooled prior by summing pseudo-counts $N_{j,l}$ from all distributions $j = 1, .., J$:

$$\beta_l = \beta_{0,l} + \sum_{j=1}^{J} N_{j,l} \tag{6.1}$$

This gives us the parameters for posterior Dirichlet distribution over $\boldsymbol{\xi}$. An approximation is then suggested by [Leonard, 1977] to enable us to calculate the posterior hyperparameters $\boldsymbol{\alpha}_j$ for each distribution j:

$$\hat{\xi}_l \approx \mathbb{E}[\xi_l] = \frac{\boldsymbol{\beta}_l}{\sum_{l=1}^{L}} \tag{6.2}$$

An approximate method is detailed in [Leonard, 1977] for updating s_0 to find a posterior mean s using Stirling's approximations:

$$\hat{s} = \frac{s_0}{s_0 + \frac{1}{L}\sum_{j=1}^{J}\sum_{l=1}^{L} N_{j,l}} \tag{6.3}$$

Alternative approximations to s may the topic of future research. Given the posterior estimate of $\boldsymbol{\xi}$, the hyperparameters $\boldsymbol{\alpha}_j$ are updated in the normal manner by adding observations to the prior:

$$\alpha_{j,l} = N_{j,l} + \hat{s}\hat{\xi}_l \tag{6.4}$$

Thus the effect of pooling is to provide more informative priors, which lose influence on the distribution as more observations from the individual distributions are received.

One way to account for varying similarity between different entities is to use this pooling mechanism between multiple overlapping pairs of entities. Rather than assuming that $\boldsymbol{\xi}_j$ is drawn from a single root Dirichlet, we may assume it is generated from a mixture of Dirichlets, where each mixture component is weighted accoring to similarity $s_{\iota,j}$ between

distributions j and ι:

$$\boldsymbol{\alpha}_j = \sum_{\iota=1}^{J} s_{\iota,j} \boldsymbol{\xi}_{\iota,j}. \tag{6.5}$$

Challenges lie in determining the pooling parameters, and doing so in a Bayesian manner. At a high level, observing similar behaviour, or similar features can provide a similarity metric between agents, task types or target labels. The communities extracted in Chapter 3 given one method for inferring similarities. When determining similarities between task types or target labels, intermediate latent features can be inferred by reducing the complete set of observations to a low dimension [Mo et al., 2013]. The use of pooling to obtain more informative priors may reduce the challenge of setting prior hyperparameters $\boldsymbol{A}_0$, leading to more accurate results. However, it introduces the new problem of how to determine similarities and pooling parameters.

6.2 Decentralised IBCC

The methods discussed so far have relied on centralised computation with a single kind of task and target variable. Individual agents could each execute the decision combination and intelligent tasking algorithms if there was perfect information sharing. However, this may not be desirable when dealing with big data problems, high costs of sharing responses or low communication bandwidth. The previous section also introduced the idea of sharing information about heterogeneous tasks, which may require information sharing between completely separate systems, so sharing the entire database would be highly impractical. Decentralising the IBCC algorithms is therefore of great value.

A natural way to split the problem is for each node to be responsible for a different set of target objects. This follows the natural division present in systems processing different datasets, potentially in remote databases, or monitoring different physical locations. Each system needs to share a summary of information about the agents they have in common, and if necessary, the different proportions of each target value. We can derive a message-passing algorithm, where after each iteration of the VB algorithm, each node sends out an

updated estimate of pseudo-counts. When the nodes are processing the same kind of task, with the same set of target values, if we use the static variant of IBCC presented in Chapter 2, the combined posterior distribution over an agent's performance can be found simply by summing the pseudo-counts in the latest messages.

With DynIBCC, the tasks relating to different nodes may form part of a sequence relating to one agent, so some dynamics may be inferred given hyperparameters sent out at the previous iteration. The effect that this has on convergence, or whether other solutions can be found remains to be investigated.

In the case of heterogeneous tasks, a function is needed to translate the pseudo-counts between systems, which may use the pooling methods described in the previous section to translate between target values of related decisions. The simplest translation is to map pseudo-counts to a simple accuracy or informativeness score per agent that can be transmitted between systems. A pooling parameter representing task similarity can then weight the scores received from other systems according to their relevance to the current system.

Another use of decentralisation is to allow scalable, parallel computation, passing only small messages between nodes. If data is divided appropriately, several iterations of VB could be completed on the same node before any messages need to be passed, reducing the overheads.

The decentralised nodes may not have perfect information sharing, especially in a multi-agent scenario where physical agents may have limited abilities to transmit data. In these cases, agents may pass on messages from other nodes as well as their own. When receiving messages from multiple agents, steps must be taken to recognise which messages are truly distinct, and which are duplicates that have been sent along multiple routes in a network. A provenance record stored in each message could act as an identifier so that agents can determine when to discard a message and whether it should be transmitted further to other nodes.

6.3 Collapsed VB

This thesis proposes the use of mean-field variational Bayes for efficient, approximate inference over IBCC models, demonstrating strong empirical performance of this approach. A promising alternative approach is *collapsed variational Bayes (CVB)* [Teh et al., 2006], which has been applied to Latent Dirichlet Allocation (LDA) [Blei et al., 2003], a technique commonly used for modelling topics in text documents that was used in experiments in Chapter 5. CVB is shown by [Teh et al., 2006] to produce more accurate results than standard VB and is computationally efficient.

Collapsed inference operates in a space where the parameters are marginalised out, rather than inferring parameters and latent target variables simultaneously. Thus an iterative VB algorithm is used to optimise the lower bound of the marginal distribution of the latent variables, rather than the joint distribution of latent variables and parameters. The collapsed approach removes the need for the strong assumption used in mean-field VB that parameters are independent of the latent variables.

As in LDA, the latent variables in the IBCC model are conditionally independent given the parameters. Marginalising out the parameters induces new, typically weak dependencies between the latent variables. In each VB iteration, updating a variable causes a change to any of its dependent variables. A mean-field VB approximation is more appropriate when dependencies are small, since changes to each variable do not cause large knock-on effects on the other variables, and the algorithm can converge quicker. However, unlike traditional VB, the collapsed variational approximation does not need to break strong dependencies between parameters and latent variables, so can offer faster convergence or more a accurate approximation when dependencies are stronger. The proposal by [Teh et al., 2006] reduces the computational expense of marginalising the parameters by accurately approximating expectations over pseudo-counts using Gaussian distributions.

Further, the CVB approach could be applied to IBCC variants that model the correlations between agents' responses. Such dependent Bayesian classifier combination (DBCC) models were proposed in [Ghahramani and Kim, 2003] but are currently seen as too com-

putationally expensive. Traditional VB is not suitable for these dependent models because it needs to impose independence between the parameters. CVB does not have such a restriction, so may offer an efficient, approximate inference method for DBCC.

6.4 Improved Decision-Making in Intelligent Tasking

Some runs of the experiments in Chapter 5 resulted in slow improvements to the AUCs as more labels were gathered, such as that shown in Figure 5.6c and the early iterations in Figure 5.4b. While Hiring and Firing produces good results on average, it would be desirable to improve its reliability by better handling these cases. The weaker runs may result from an unfortunate initial set of responses, or could be the result of either the wrong workers being fired, or workers not being fired after they became uninformative, implying that receiving a label from an uninformative worker had erroneously higher utility than receiving a label from a new worker or an informative worker. The following three scenarios describe hypotheses as to when the hiring and firing algorithm could cause the wrong agents to be hired and fired.

(1) The features of the objects contain latent structure, i.e. clusterings, which may be unrelated to the target variables we wish to infer. For example, if we take a document corpus containing news report from two different sources, one clustering may group articles from the same source, but this would not help detect the relevance of articles within each group to a particular search query. If we have a number of uninformative agents, they may assign incorrect labels that identify irrelevant clusters as relevant to the target topic. If these mistakenly-identified clusters are more distinct than the correct groupings, it appears more likely that the correct agents are guessing randomly. Thus at some point the algorithm will confuse the informative and uninformative agents, and can fire the wrong agents. The same issue may continue to affect newly-hired agents.

(2) Having incorrectly learnt to trust the uninformative agents, decisions from informative agents that rectify the situation could cause increases in the estimated uncertainty over some target labels. This is likely to occur if VB exaggerates the confidence in the ini-

tial decisions due to its imposition of independence on the target labels and the confusion matrices for agents and features. The presence of certain features consolidates the belief in particular target values, but the confusion matrices over the features were themselves derived from the same target variables.

(3) Finally, after a large number of random guesses, the features would be believed to be randomly distributed, while the uninformative agents are marked as highly accurate. In the VB inference method, the additional observations reduce uncertainty in the confusion matrices, even if there are no informative features or responses from other agents that confirm them. This again comes from breaking the dependencies between the confusion matrices and target variables. Eventually, if we obtain multiple agents for objects, disagreement between agents should allow IBCC to detect that they are making random guesses. In the models presented in this thesis, there is no prior to encode a belief that features are not random, and indeed, the majority probably bear no relation to the target variables. Since the model infers that the existing uninformative workers are highly accurate, new workers appear less informative so are never hired.

Potential problems could therefore stem from over-confidence in the model, especially during earlier iterations. With (1) and (2), the problem may be over-confidence in an incorrect partition of objects, and with (3), the over-confidence in the confusion matrices of random agents. In problems (1) and (2), the result is that the Hiring and Firing algorithm does not account for the risk of firing the wrong agent. If the algorithm was not over-confident, this risk would be reduced and better accounted for by the utility function. If informative agents are given more time before being fired, any mistakes in firing agents should be overcome more quickly as new, informative agents will be able to supply a greater number of informative labels.

Traditional VB is one likely cause of over-confident estimates (see also [Turner and Sahani, 2010]). Therefore, a possible solution is to consider collapsed variational Bayes (CVB), as suggested in the previous section, to properly account for dependencies between confusion matrices and target variables. With CVB, it is less likely that the solution will

collapse so quickly to assuming one agent is right, and the other is wrong. CVB also reduces fluctuations in parameters, so may reduce the effect of (2), where additional labels result in uncertainty estimates increasing.

Even with more reasonable confidence estimates, it may still be possible to make incorrect hiring decisions that lead to failure modes or very slow learning. Quantifying this risk may be a topic of future research. Choosing the decisions that maximise expected utility should maximise the true utility as the number of iterations tends $\to \infty$. However, it might be necessary to reduce the risk of falling into periods of slow learning, even at some long-term cost. This could be achieved in various ways. First, a cost term in the utility function (see Equation 6.8) could include a specific penalty for lack of diversity in the workforce. Second, the expected utility function could penalise situations according to how different the best and worst cases are. While the current Intelligent Tasking approximation was shown to work well on average, the suggestions in this section could improve the efficiency of learning by avoiding incorrect hiring and firing decisions.

6.5 Optimising Future Rewards

The previous section suggested some possibilities for improving Intelligent Tasking. A further opportunity is to extend intelligent tasking to consider actions that lead to information gain in the future, including training exercises or tasks that motivate a participant to complete more tasks. Considering future utility may also help detect possible failure modes and better quantify the benefits of hiring new workers. At the first iteration, the new worker may not be very informative, but as more iterations progress, there is the chance that the future utility can either be very high or very low. Thus, according to Jensen's inequality, calculating the information gain without explicitly accounting for these possibilities may be an underestimate:

$$I(t_\iota; \mathbb{E}_{c_i^{(k)}, \boldsymbol{c}_{\text{fut}}}[c_i^{(k)}, \boldsymbol{c}_{\text{fut}}]|\boldsymbol{c}, \boldsymbol{y}) \leq \mathbb{E}_{c_i^{(k)}, \boldsymbol{c}_{\text{fut}}}[I(t_\iota; c_i^{(k)}, \boldsymbol{c}_{\text{fut}}|\boldsymbol{c}, \boldsymbol{y})], \tag{6.6}$$

where the left-hand side is calculated from our current expectation and the right-hand side marginalises over possible future outcomes. The utility function of Equation 5.4 can be expanded to optimise over long-term rewards:

$$U(k,i|\mathbf{c}) = \sum_{\iota=1}^{N} I(t_\iota; c_i^{(k)}, \boldsymbol{c}_{\text{fut}}|\boldsymbol{c},\boldsymbol{y}) + \text{Cost}(k,i) \tag{6.7}$$

$$= \sum_{\iota=1}^{N} \Big(I(t_\iota; c_i^{(k)}|\boldsymbol{c},\boldsymbol{y}) + I(t_\iota; \boldsymbol{c}_{\text{fut}}|\boldsymbol{c},\boldsymbol{y}, c_i^{(k)}) \Big) + \text{Cost}(k,i), \tag{6.8}$$

where $\boldsymbol{c}_{\text{fut}}$ is the set of as-yet-unknown observations that the system will receive in the future, and $\text{Cost}(k,i)$ combines the costs of any financial payments to agents, time penalties, or rewards other than information gain. The two terms of Equation 6.7 provide a trade-off between the information gained over the duration of the system by observing agent k completing task i, and any costs the system might incur. Thus any rewards or incentives to agents are balanced against their cost.

In related work, the choice of suitable payments for crowdsourcing workers has been considered as an offline optimisation task before starting a crowdsourcing process. For example, [Tran-Thanh et al., 2013] finds that there is an optimal price to pay workers depending on task difficulty that maximises the performance of a system within a fixed budget, whilst in [Karger et al., 2011], tasks are assigned to specific workers using a scalar reliability score, assuming fixed costs, pre-determined numbers of workers per task and pre-determined numbers of tasks per worker. In the online optimisation setting, [Kamar et al., 2012] considers a fixed cost per response when deciding whether to hire additional workers to classify an object. An adaptive approach that balances costs with the value of information received does not appear to have been developed.

The three terms of Equation 6.8 split the long-term information gain into two parts. The first represents *immediate utility*: the information gained about $\boldsymbol{t}$ by learning $c_i^{(k)}$ when we already know $\boldsymbol{c},\boldsymbol{y}$. The second is *future utility*: the information gained about $\boldsymbol{t}$ through future observations $\boldsymbol{c}_{\text{fut}}$, assuming we have already observed agent k completing task i, and previous responses $\boldsymbol{c},\boldsymbol{y}$. Any improvement in the system through k completing task i

should lead to an increase in future utility. Such improvements may arise through agents gaining experience and training, but also through the system learning about the agents and making better assignments. Therefore, a task that maximises immediate utility may not be optimal if it prevents or delays a sequence of tasks that is better over the long term. We can predict the utility of an assignment by calculating the expectation over Equation 6.8:

$$\begin{aligned}\hat{U}(k,i|\mathbf{c}) &= \mathbb{E}[U(k,i|\mathbf{c})] \\ &= \sum_{\iota=1}^{N} \mathbb{E}_{c_i^{(k)},\boldsymbol{c}_{\text{fut}}}[I(t_\iota; c_i^{(k)}, \boldsymbol{c}_{\text{fut}}|\boldsymbol{c},\boldsymbol{y}) + \text{Cost}(k,i)] \\ &= \int p(c_i^{(k)}|\boldsymbol{c},\boldsymbol{y}) \sum_{\iota=1}^{N} I(t_\iota; c_i^{(k)}|\boldsymbol{c},\boldsymbol{y}) \mathrm{d}c_i^{(k)} \\ &+ \iint p(\boldsymbol{c}_{\text{fut}}|c_i^{(k)},\boldsymbol{c},\boldsymbol{y}) \sum_{\iota=1}^{N} I(t_\iota; \boldsymbol{c}_{\text{fut}}|c_i^{(k)},\boldsymbol{c},\boldsymbol{y}) \mathrm{d}\boldsymbol{c}_{\text{fut}} \mathrm{d}c_i^{(k)} + \text{Cost}(k,i) \end{aligned} \tag{6.9}$$

The expected future utility term in Equation 6.9 can be calculated from the Shannon entropy:

$$\begin{aligned}\mathbb{E}[I(t_\iota; c_i^{(k)}, \boldsymbol{c}_{\text{fut}}|\boldsymbol{c},\boldsymbol{y})] =& H(t_\iota|\boldsymbol{c},\boldsymbol{y}) - \mathbb{E}_{c_i^{(k)},\boldsymbol{c}_{\text{fut}}}[H(t_\iota|c_i^{(k)},\boldsymbol{c}_{\text{fut}},\boldsymbol{c},\boldsymbol{y})] \\ &= \iiint p(t_\iota|\boldsymbol{c},\boldsymbol{y},c_i^{(k)},\boldsymbol{c}_{\text{fut}}) \ln p(t_\iota|\boldsymbol{c},\boldsymbol{y},c_i^{(k)},\boldsymbol{c}_{\text{fut}}) \mathrm{d}t_\iota \ \mathrm{d}c_i^{(k)} \ \mathrm{d}\boldsymbol{c}_{\text{fut}} \\ &\quad - \int p(t_\iota|\boldsymbol{c},\boldsymbol{y}) \ln p(t_\iota|\boldsymbol{c},\boldsymbol{y}) \mathrm{d}t_\iota \end{aligned} \tag{6.10}$$

Predicting future utility requires significant approximation, as we cannot integrate directly over all possible future task assignments and observations. An additional complication in a multi-agent scenario is that the integration over $\boldsymbol{c}_{\text{fut}}$ must also consider tasks assigned to agents other than k, including those that are already under way. One simplifying assumption is that the agents always choose the optimal assignment in future tasks, assuming they are rational and adhere to the same measure of utility. However, the computational cost of the integration still increases exponentially with the number of future assignments we wish to consider, which in itself may not be known, since each future assignment results in L possible outcomes. One approximation is to consider the expected information gain over a

small set of future tasks (a finite horizon). The future tasks can be chosen greedily or by using approximations that are cheap to compute, for example by using techniques applied to agent planning or coalition formation [Dang et al., 2006]. A different approach is to learn a function that estimates the future utility from the confusion matrices directly, thus avoiding the need for any complex calculations of information gain over multiple time-steps. This is a meta-learning problem, where we regress onto the future utility from the state of confusion matrices. Such a method may also need to consider the distribution over the number of tasks that an agent is likely to complete over the long term, as this affects how much utility can be derived from training exercises.

The problem of optimising over both immediate and future utility is also faced in the field of *Reinforcement Learning (RL)* [Sutton and Barto, 1998]. In this domain, the decision-making agent receives rewards rather than explicit training examples, and must learn a policy for choosing an action in a given state. This scenario can be modelled using a Markov Decision Process, which considers the optimal action to take given only the current state. As with Intelligent Tasking, the core issue is to choose between expected immediate reward through exploitation and reward that is realised over the long term via exploration. In the case of intelligent tasking, future rewards can be realised by the weak control agent exploring the agents' behaviour as well as through the individual agents exploring and learning themselves. An established RL method that can be applied to optimise any MDP is *Q-learning* [Watkins, 1989]. This method learns a function $Q_\tau(s_\tau, a_\tau)$ that gives the expected utility of taking action a_τ at time τ in state s_τ, then making optimal decisions at all subsequent times. The learning algorithm involves an iterative update that relates later rewards to earlier decisions, with a discounting factor that can be used to favour receiving rewards sooner. Under certain conditions, the algorithm is guaranteed to converge to the utility function of the optimal policy [Watkins and Dayan, 1992]. With Q-learning, obtaining the optimal balance between exploration and exploitation remains a challenge, and several variants of the method seek to address this issue, such as [Hasselt, 2010]. Future research could develop RL approaches for the intelligent tasking domain. In this domain,

rewards are defined by Equation 6.8, the state by the set of agents, responses and objects, and the action space covers all pairs of agents and tasks. Adapting RL methods to intelligent tasking requires a policy function that can generalise to new states as responses are received, so a suitable representation of system state is required that can handle the large number of state variations.

In summary, a key research challenge is to choose tractable approximations that satisfactorily account for the future utility. Future work should therefore investigate greedy approximations to optimising 6.9 that limit the number of possible tasks and outcomes that must be integrated over.

Future utility could be used to predict behavioural changes through training, and therefore allow us to predict the utility of assigning a training exercise rather than an analysis task that results in immediate information gain. Careful selection of training tasks is necessary when each exercise carries a cost or takes a significant amount of time, such as when a human or agent must learn by exploring a physical space or actively searching for training data. Intelligent training first requires a method for proposing training exercises and predicting their effect on the current state of an agent, given their experience to date. Similarly, when considering motivating humans, an algorithm is needed to estimate the effect of a task on an agent, which likely depends on not just the current state, but also on other tasks completed recently.

Consider the challenge of predicting the changes to an agent that arise from completing a task. Using DynIBCC, we wish to predict the change $\Delta_\tau^{(k)}$ to the distribution over a confusion matrix, $\Pi_\tau^{(k)}$, as a result of assigning task i to agent k at time-step τ. The change matrix $\Delta_\tau^{(k)}$ could be treated as a set of pseudo-observations taken from other, similar agents. The matrix would represent pseudo-counts of responses that can be added to the Dirichlet hyperparameters $\mathbf{A}_\tau^{(k)}$ using the DGLM in the normal manner described in Chapter 4. However, the state noise variance q_τ should also be increased to reflect our uncertainty over whether the task will have the desired effect. We can estimate $\Delta_\tau^{(k)}$ by observing the effects of i on other agents, and forming a weighted combination of the ob-

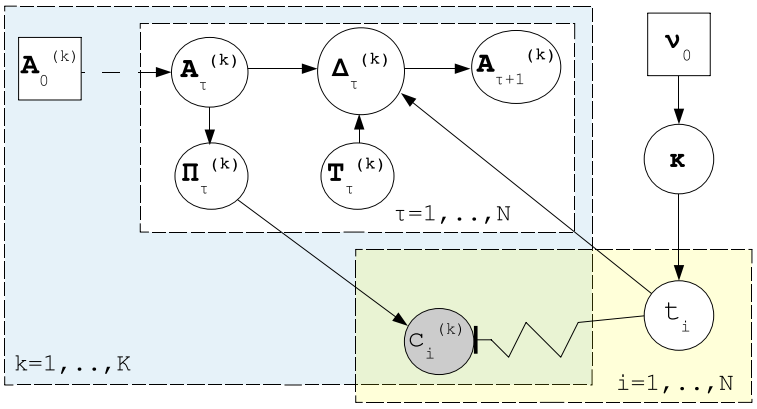


Figure 6.2: DynIBCC augmented to model the change $\Delta_\tau^{(k)}$ between time-steps dependent on the training completed, $T_\tau^{(k)}$.

servations according to the agents' similarity to k. An observation $\Delta_\tau^{(k)}$ can be obtained by subtracting $A_{\tau+1}^{(k)} - A_{\tau-1}^{(k)}$, i.e. the difference between the parameters before and after training. A key challenge here is to determine similarity between each situation. This similarity represents the probability that the current agent k will undergo the same change as a previously observed agent. A matrix $\Delta_{0,\tau}^{(k)}$ represents the prior over the change. The sum of weights over the observations and the prior should therefore equal one. Similarity can be calculated by comparing the confusion matrices of two agents but we also need to consider the tasks that they have recently completed, e.g. previous training exercises or repeating similar tasks, causing boredom.

The weighted sum approach can also be viewed as an addition to the DynIBCC model. In Chapter 4, the distribution $p(A_\tau^{(k)}|A_{\tau-1}^{(k)})$ is left unspecified. We now assume that the change is partly due to the addition of $\Delta_\tau^{(k)}$, the change due to training or motivation. Now, for training exercises, we can specify the distribution over $\Delta_\tau^{(k)}$ as a multivariate Gaussian, dependent on the value of $A_{\tau-1}^{(k)}$ and the set of previously-completed training exercises, $T_\tau^{(k)}$, as well as the training exercise completed at time τ. When learning the distribution $p(\Delta_\tau^{(k)}|A_{\tau-1}^{(k)}, T)$, it may be important to make use of similarities between training tasks and informative priors, e.g. if we design a training exercise to improve a particular skill.

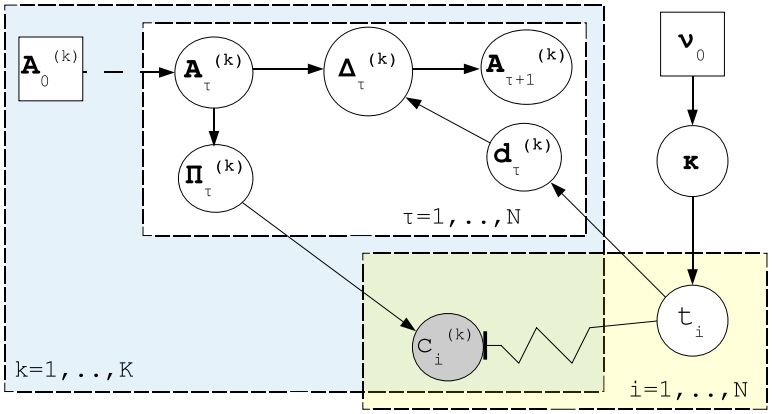


Figure 6.3: Graphical model for DynIBCC extended to model the change $\Delta_\tau^{(k)}$ between time-steps dependent on the task diversity.

This model is depicted in Figure 6.2.

When considering training exercises, it may be desirable to propose a small set of possible tasks, then evaluate their utility. There may not be a fixed bank of training exercises to evaluate, rather the exercises may be tailored to individual agents. It is also unnecessary to consider exercises to improve skills that an agent is already proficient at. A set of proposals can include exercises where $\Delta_\tau^{(k)}$ is high given the current state of the agent, $A_\tau^{(k)}$. The exercises can then be adjusted and the future utility estimated in full.

It is not clear how to choose sequences of training tasks, where one exercise may be a pre-requisite of another but not yield immediate improvements. One solution is to observe that a task i_{pre} has always been completed prior to a task i when i has produced improvements. Then, the two tasks can be proposed as a pair, i.e. we evaluate the utility of assigning both tasks together. A future challenge is to develop these ideas further and evaluate the effectiveness of such strong approximations.

One possibility for measuring motivational value is to use a measure of task diversity, $d_\tau^{(k)}$, for the recent objects observed by agent k up to time-step τ. Diversity can be measured by looking at the entropy in the distribution over the set of recent tasks. We can then model a change $\Delta_\tau^{(k)}$ as dependent on the change in task diversity $d_\tau^{(k)}$ over recent time-steps by

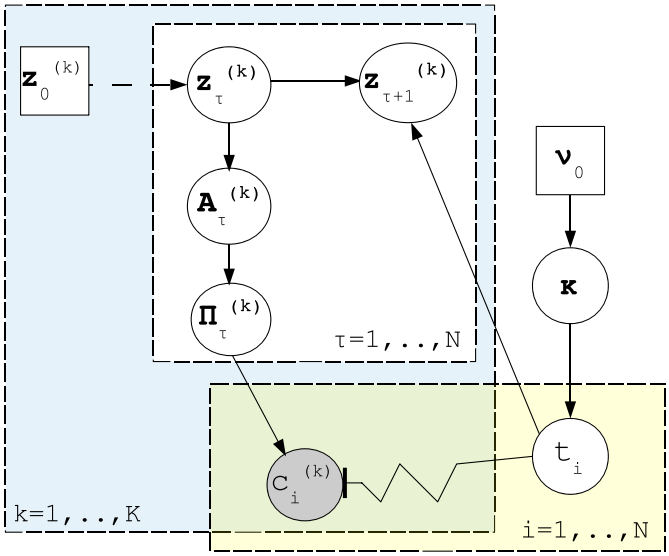


Figure 6.4: A revised graphical model for DynIBCC where the agents have an underlying type $z_\tau^{(k)}$ that evolves according to a Markov process depending on the task presented.

assigning i. Thus the assumption is that diversity is beneficial to agent reliability.

Perhaps more important is to consider how the motivation affects the probability of completing another task or giving up. The DynIBCC model could be updated to include a probability of completing another task at each time-step, which would be dependent on the diverstiy score for previous tasks, and may also depend on additional information, such as the time between completing tasks. Such a model is shown in Figure 6.3. By modelling the probability of dropping out as dependent on diversity, the model is able to remedy boredom in human agents even in cases where no change in reliability is observed.

An alternative that does not involve altering the DynIBCC model is to take weighted averages of the observed information gain for other similar agents completing a particular training exercise. Thus, instead of learning the change to the confusion matrices, we perform the simpler step of learning the future utility directly.

The suggestions so far for intelligent training involve learning distributions over $\Delta_\tau^{(k)}$, potentially introducing new complexities and the need for additional observations from which to learn these distributions. Simpler approximations may be possible, for example,

by introducing latent types of agents to summarise their current state. Thus, rather than depending on $A_\tau^{(k)}$, which may be a large matrix, the change matrix $\Delta_\tau^{(k)}$ depends on a scalar type variable, $z_\tau^{(k)}$, as depicted in Figure 6.4. Chapter 3 showed that latent types of agent exist in Galaxy Zoo Supernovae, by performing community analysis on the volunteers. The modification in Figure 6.4 introduces clustering directly into the model, but no longer models smooth or gradual dynamics. Rather than modelling a change matrix $\Delta_\tau^{(k)}$ that is dependent on the entire confusion matrix, we simply model the transition probabilities between latent types in a similar manner to a Hidden Markov Model (see [Bishop, 2006]). The type $z_\tau^{(k)}$ would be dependent on the previous type $z_{\tau-1}^{(k)}$ and the task completed at time-step $\tau - 1$. Latent types, or communities also indicate possibilities for peer training, where an agent is assigned to help another to improve. Each type has an expected future utility; the aim is to train agents who belong to types with low utility. An aspirational type is chosen for a low-utility agent according to the type with highest future utility, i.e. the type in most demand given the other agents in the current pool. An agent from the aspirational type is then selected to help train a low-utility agent.

When dealing with communities, it may be acceptable to pre-determine estimates of expected utility from completing training exercises. This can be done in an information-theoretic manner by looking at the difference in information gain between responses of agents of different types. Each training exercise would have an associated utility vector, where each element represents the utility of training an agent of a particular type. This would not be able to account for the system's current needs, e.g. if certainty over one target value is now high, the demand for agents who can identify that label accurately will reduce. However, that may be of less importance with a long-running project, where new objects to analyse are continually being added, and the future utility of a particular improvement to an agent remains more or less constant.

A simpler approach to motivation and boredom is to add a penalty for assigning similar tasks repeatedly. This penalty would form part of the cost term of the utility function in Equation 6.8. Starting at zero, it would accumulate the similarity between the task at

$\tau - 1$ and τ, discounting the previous value at each iteration. Similarity between tasks can be calculated from the objects' features, for example using cosine similarity or Euclidean distance [Baeza-Yates and Ribeiro-Neto, 1999]. Further research is needed to derive this penalty in a principled manner.

This section suggests various avenues for developing automatic training suggestions and adjusting for motivation. As well as inferring the need for training and motivation by observing agents' behaviour, an obvious alternative is for human agents to self-diagnose the need for training or variation in tasks, simply by selecting an option for training or different kinds of tasks from a user interface. Future research is required to determine whether the more complex methods are justified, or whether we can build the simpler approximations into the intelligent tasking approach.

6.6 Preference Combination

Humans often provide pattern-matching skills and creative thinking in a multi-agent system. Unlike software agents, people find it difficult to express their uncertainty in a numerical way. To date this problem has been avoided by requesting that human decision makers provide categorical labels, but that rejects people's ability to make finer distinctions between objects. For example, in the Galaxy Zoo Supernovae (GZSN) application, only three scores are possible, with the middle score intended to capture all very uncertain decisions. With many objects it is indeed difficult to assign a label with confidence, but there is a limit to how often a user would wish to pick "I don't know". The community analysis in Chapter 3 showed that different groups of users end up ascribing different meanings to the scores, with some never confident enough to assign the labels that represent certainty. The aim of the GZSN project is to prioritise likely supernovae examples for further analysis. The combined decisions produced by IBCC allow objects to be ranked by probability of supernova, but rely on individual volunteers making meaningful use of the highly restricted scoring system, thereby rejecting any further information the individuals can provide about their uncertainty or comparisons between objects. With the TREC

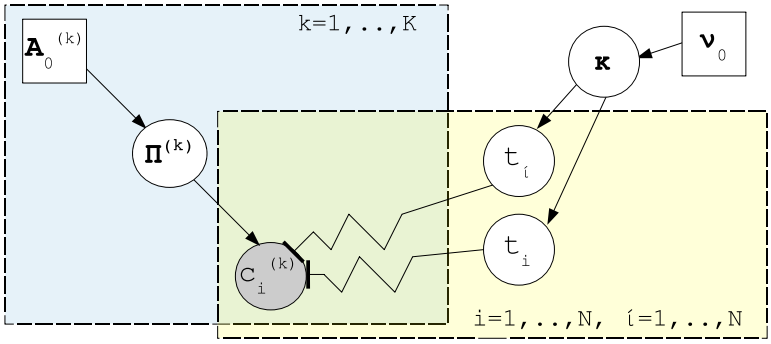


Figure 6.5: A revised graphical model for combining preference pairs in a similar manner to IBCC.

document classification scenario, it was not possible to express uncertainty at all, yet the end goal was to find relevant documents. Again, this leaves the workers to determine the line between "relevant" and "not relevant" themselves, with much room for error and no consideration for more subtle distinctions. It would therefore be desirable to avoid the need for a hard scoring system that some workers may have difficulty understanding.

An alternative to categorical labelling is to ask humans to express a preference between objects in a pair. The preference captures which of two objects an agent thinks is more likely have a particular target label. In some situations, we can observe preferences passively as an agent goes about their natural behaviour, rather than soliciting answers to a question. For example, the route a person chooses may represent certain preferences, which reflect the relative speed or pleasantness of each option. Directly asking for an objective assessment of the quality of each route may not yield informative answers. The number of comparisons required to completely sort all objects scales on average with $\mathcal{O}(N \log N)$, whereas simply labelling all objects is clearly $\mathcal{O}(N)$, where N is the number of objects. Efficient intelligent tasking is especially important in preference combination, as the space of possible pairs is $\mathcal{O}(N^2)$. However, the preference pairs may be faster to obtain from people than categorical labels.

Probabilistic methods for ranking objects by combining pairwise preferences include the Bradley-Terry model [Bradley and Terry, 1952], which was extended by [Chen et al., 2013] to handle unreliable and inconsistent preferences from a crowd. However, inference

was not handled in a fully Bayesian manner. In many applications the aim is to estimate the the probability of a particular target variable, so a further research challenge is to combine the complete set of preferences to infer a distribution over a target variable for every object. To achieve this, the IBCC model can be augmented to account for preferences:

$$p(\boldsymbol{t}, \boldsymbol{c}, \boldsymbol{\kappa}, \boldsymbol{\Pi}|\boldsymbol{\nu}_0, \boldsymbol{A}_0) = \prod_{i=1}^{N} \kappa_{t_i} \prod_{\iota=1}^{N} \left\{ \prod_{k=1}^{K} \left(\pi^{(k)}_{t_i,t_\iota}\right)^{c^{(k)}(i,\iota)} \left(1 - \pi^{(k)}_{t_i,t_\iota}\right)^{1-c^{(k)}(i,\iota)} \right\}$$
$$p(\boldsymbol{\kappa}|\boldsymbol{\nu}) \prod_{j=1}^{J} \prod_{l=1}^{J} \prod_{k=1}^{K} p\left(\pi^{(k)}_{j,l}|\boldsymbol{\alpha}^{(k)}_{0,j,l}\right). \tag{6.11}$$

where $\boldsymbol{\Pi}^{(k)} = [\boldsymbol{\pi}^{(k)}_1, .., \boldsymbol{\pi}^{(k)}_J]$ is a preference confusion matrix, in which each each row j corresponds to the target value of the first object presented, and each column corresponds to the target value of the second object. Each entry in a row j corresponds to the probability of preferring the first object i over the second object ι: $\pi^{(k)}_{j,l} = p\left(\text{prefer}(k, i > \iota)|t_i = j, t_\iota = l\right)$, and has a Beta prior $p(\pi^{(k)}_{j,l}|\boldsymbol{\alpha}^{(k)}_{0,j,l})$. Thus a response $c^{(k)}(i, \iota)$ from an agent k is either 1, which indicates that i is preferred to ι, or 0, which indicates the opposite. A revised graphical model is shown in Figure 6.5. The new joint distribution has induced dependencies between the target labels as they are no longer independent given the responses, so a collapsed variational approach may therefore be applicable.

As with categorical labelling problems, features can be used to interpolate between objects according to their similarity, so that we can generalise an individual's preferences having observed a sample. Various methods for preference learning over a single individual have been explored. Examples include [Brochu et al., 2008], who consider actively sampling informative pairs to learn a preference manifold efficiently; [Herbrich et al., 1998; Wong and Yao, 1988], who have applied preference learning to information retrieval and relevance problems; and [Chu and Ghahramani, 2005], who introduce Gaussian processes to provide a Bayesian treatment of preference learning.

There remain a number of research opportunities in bringing the existing work on preference learning into the space of decision combination with multiple unreliable workers.

This could prove a valuable contribution to applications such as citizen science, crowdsourcing, and automatically organising teams of mobile agents and humans.

6.7 Summary of Future Work

There are many other natural extensions to this work, including moving away from categorical target labels or responses, richer models of agents' expertise, and other methods for improving the scalability of Intelligent Tasking. The extensions proposed in this chapter cover general Machine Learning issues such as accuracy, scalability, and exploiting data through sharing information; they also considered some problems specific to human agents, including training and motivation. Common to all of these developments is the use of Bayesian methods, albeit with computationally-efficient approximations based on clear simplifying assumptions. Future work can therefore build on methods such as DynIBCC that were developed in this thesis, exploiting the core idea that a Bayesian model of agents' behaviour is the foundation for robust and efficient aggregation of information.

6.8 Limits to Decision Combination

Although the methods proposed in this thesis achieved strong performance in a range of experiments, we can conceive of situations where decision making with multiple agents would not produce reliable decisions. First, we rely fundamentally on some agents producing decisions that are more informative than a random guess. As we increase the number of informative agents whose errors are uncorrelated, we can increase the accuracy of the combined decisions. Thus when aggregating a small number of very noisy decision-makers, we cannot expect to attain a high prediction accuracy.

Secondly, methods such as IBCC could perform badly if there is insufficient information to discriminate between reliable and unreliable agents. In effect, such a situation could cause IBCC to fail, since the reliable agents' contributions may be ignored. This would lead to the Hiring and Firing algorithm excluding the wrong agents, and thus the al-

gorithm would be unlikely to obtain new information that could correct the problem. This problem can be avoided by providing some training labels that allow us to detect a number of reliable agents, to which other agents can be compared. The correct function of IBCC in Citizen science and crowdsourcing applications could therefore be ensured by including some known tasks for the workers. If such training data is unavailable, the algorithms operate in unsupervised mode, and rely on latent structure in the data to detect the trustworthy agents. This latent structure consists of correlations in the responses of different agents to similar tasks, which suggest they are not guessing randomly. If a number of malicious agents collaborate to give the same, uninformative responses, they may erroneously be detected as reliable agents by IBCC. In situations where trustworthy agents do not complete similar tasks, it may also be difficult to detect any latent structure and thus identify the reliable agents.

In summary, the approach espoused in this thesis depends on either sufficient training data, prior knowledge of individuals' behaviour, or detecting correlations among a cohort of decision makers who are at least weakly informative. In many systems, this requirement can be satisfied through the use of a small number of trusted expert labels, or by obtaining responses from multiple workers for similar tasks.

Appendix A

Notation and Glossaries

This section is intended to provide the reader with an easy reference to some notations used in this thesis. A list of commonly-used acronyms used in this work is given in Table A.1. Table A.2 lists the notational conventions used in equations in the preceding section. A reference to the symbols used in the various model descriptions is shown in Table A.3.

Acronym	Meaning
MAS	Multi-agent system.
LinOP	Linear opinion pool.
LogOP	Logarithmic opinion pool.
BCC	Bayesian Classifier Combination [Ghahramani and Kim, 2003].
IBCC	Independent Bayesian Classifier Combination.
DynIBCC	Dynamic Independent Bayesian Classifier Combination.
VB	Variational Bayes.
NB	Naïve Bayes.
MCMC	Markov-chain Monte Carlo.
DGLM	Dynamic generalised linear model.

Table A.1: Symbols used in the model descriptions and derivations in this thesis.

Name	Example	Description
Lower case letter, unbolded	b	A scalar variable.
Upper case letter, unbolded	N	A count, the number of objects in an iteration.
Lower case letter, bold	$\boldsymbol{\pi}$	A vector.
Upper case letter, bold	$\boldsymbol{\Pi}$	A matrix or an ordered set of matrices.
Upper case letter, bold, underlined	$\underline{\boldsymbol{\Pi}}$	An ordered set of matrices with more than three dimensions.
Subscript indices	$\pi_{j,l}$	Indices into a matrix or vector. The order of the subscripts corresponds to the dimensions they refer to.
Superscript indices	$\boldsymbol{\pi}_j^{(k)}$	Index into an ordered set of matrices.
Square brackets	$[a, b, c]$	Vector, or where the context is made clear, an inclusive interval.
Sequence of indices	$1, .., J$	An ordered set of integer values starting from the first number to the last.
Curly brackets	$\{A, B\}$	A set, which may be ordered depending on the context.
Enumerated set	$\{x^{(k)} \vert k = 1, .., K\}$	An ordered set where the elements are specified by a sequence separated by '|'.
Hat/caret	$\hat{w}$	An approximate mean or variance.
Tilde	$\tilde{\lambda}$	An approximation to a function of a variable.

Table A.2: Conventions for mathematical notation used in this thesis.

Symbol	Description
b	Continuous belief value.
a	The log-odds or activation of a predictions.
c	Categorical label, e.g. a class.
w	Weights on an agent's response.
t	Target label denoting a combined decisions.
π, $\boldsymbol{\Pi}$	Likelihood of agent's responses, agent confusion matrices.
κ	Proportions of each target value.
α, $\boldsymbol{A}$	Hyperparameter over agent confusion matrices.
ν	Hyperparameter over target value proportions.
l	Value of agent's response.
L	Number of possible agent's response values.
J	Number of target values, e.g. possible decisions.
N	A count of objects or responses.
k	Index of an agent.
K	Total number of agents.
i	Index of an object or data point.
$q()$	Approximate distribution (e.g. for variational inference).
$q^*()$	Optimal variational factor given current values for other factors.
τ	Index of a time-step.
$s(k, i)$	Time-step at which k responded to i.
$T(k, \tau)$	Index of data point responded to at time τ by k.
$\boldsymbol{h}$	An input vector; in DynIBCC this is the target label represented as a vector of binary values.
$\boldsymbol{v}$	State noise.
q	State noise variance.
$\boldsymbol{I}$ Identity matrix.	
$\boldsymbol{P}$	Covariance of state variables or weights.
$\boldsymbol{c}_\tau$	Vector of responses up to time-step τ.
$\Psi()$	The standard digamma function [Davis, 1965].
$x_{a\|b}$	The value of x at time-step a given observations up to time-step b.
ρ, $\boldsymbol{R}$	Likelihood of features, feature confusion matrices.
β, $\boldsymbol{B}$	Hyperparameter over feature confusion matrices.
$\boldsymbol{\nu}_0, \boldsymbol{\alpha}_{0,j}$	Subscript zero denotes a prior hyperparameter.
x	Binary feature; other generic variables.
y	Probability of a feature.
ϵ	Error.
δ	Delta function; 1 when its argument is 0, and 0 otherwise.
M	A model.
z, $\boldsymbol{Z}$	Latent variables in a model.
$\boldsymbol{X}$	Data or observations.
$\boldsymbol{\theta}$	Model parameters.
$\mathbb{E}_x[...]$	An expectation with respect to x.
$\mathbb{V}$	Variance.

Table A.3: Symbols used in the model descriptions and derivations in this thesis.

Appendix B

Algorithms

This section provides pseudo-code to enable an easier understanding of two key algorithms introduced in this thesis. Listing B.1 details the IBCC-VB algorithm from Section 3.2.3, implementing the necessary equations from Section 3.2.2. Below in Listing B.2 is the pseudo-code required for the DynIBCC-VB algorithm described in Section 4.4, which shows how to implement the filter and smoother steps of Section 4.3.4. For DynIBCC-VB, only the pseudo-code for the method "updateAlpha()" is given, since the rest of the algorithm can be implemented using the same pseudo-code specified in Listing B.1.

Listing B.1: Pseudo-code for the IBCC-VB algorithm.

```
//hyperparameters -- must be set before running
A_0, nu_0, max_its = 1000

function combine(c_all, t_known):
    //initialise confusion matrices
    ln_Pi = initPi(t_known)
    //initialise class proportions
    ln_kappa = initKappa(t_known)

    change = 0
    t_all_prev = null
    num_its = 0
```

```
    while(change>0 and numIts<max_its): //iterate till convergence
        //update target labels
        t_all = updateT(c_all, ln_Pi, ln_kappa, t_known)
        //update confusion matrices
        ln_Pi = updatePi(t_all, c_all)
        //update class proportions
        ln_kappa = updateKappa(t_all)

        //check for convergence
        if (t_all_prev != null):
            //sum squared differences of probabilities for all
                classes & all data points
            change = sum(sum( (t_all - t_all_prev)^2 ))
        num_its++

function updateT(c_all, ln_Pi, ln_kappa, t_known):
    for each object i=1,..,N:
        t_i = [vector of length J]
        if i in t_known: //training label
            t_i = t_known(i,:)
        else //test data point
            for each class j=1,..,J:
                t_i(j) = ln_kappa(j); //init to log joint prob.
                for each agent k=1,..,K:
                    c = c_all(i,k)
                    t_i(j) += ln_Pi(j,c,k)
            t_i = exp(t_i) / sum(exp(t_i)) //normalise

function updateKappa(t_all):
    N_j = sum_rows(t_all)
    nu = nu_0 + N_j
    return ln_kappa = digamma(sum(nu)) - digamma(nu)

function updatePi(t_all, c_all):
```

```
    A = updateAlpha(t_all, c_all)
    //Calculate ln Pi from hyperparameters
    return lnPi = digamma(sum_columns(A)) - digamma(A)

function updateAlpha(t_all, c_all):
    A = [3-D matrix with dimensions J, L and K]
    for each agent k=1,..,K:
        N_jlk = [all-zero matrix with J rows and L columns]
        for each object i=1,..,N:
            //obtain response from k for i
            c = c_all(i, k)
            for each class j=1,..,J:
                //update count of responses with p(t_i=j)
                N_jlk(j,l) += t_all(i, j)
        //update Alpha hyperparameters for k
        A(:,:,k) = A_0(:,:,k) + N_jlk
    return A
```

Listing B.2: Pseudo-code for the "updateAlpha()" method in the DynIBCC-VB algorithm.

```
function updateAlpha(t_all, c_all):
    A = [4-D matrix with dimensions J, L, K and N]
    for each agent k=1,..,K:
        for each response value l=1,..,L:
            A(:,l,k,:) = run_filter_smoother(t_all,c_all,k,l)
    return A

function run_filter_smoother(t_all,c_all,k,l):
    //initialise
    A = [2-D matrix with dimensions J and N]
    W_mean_prev = ln( digamma(sum_columns(A_0))-digamma(A_0(:,l,:)) )
    P_prev = [2-D square matrix with dimensions J,J]
    for each class j=1,..,J:
        P_prev(j,j) = 1/A_0(j,l,k) + 1/(sum_columns(A_0(j,:,k)))
```

```
//Filtering
for each time-step tau=1,..,N:
    i = get_object(tau,k) //find object at current time-step
    c = c_all(i, k) //obtain response from k for i

    h = t_all(i,:)
    //Filter Step 1
    W_mean_prior = W_mean_prev //state prior mean
    P_prior = P_prev + qt*I // state prior covariance

    //Filter Step 2
    eta_prior = h*Wmean_prior
    r_prior(i) = h*P_prior*h.T // the ''.T'' represents transpose
    //prior pseudo-counts
    alpha_tilde_prior = 1/r_prior(i) * (1+exp(eta_prior))
    sum_alpha = alpha_tilde_prior*exp(-eta_prior)

    //Filter Step 3: add pseduo-count if k answered "1"
    alpha_tilde_post = alpha_tilde_prior + (c==1)

    //Filter Step 4: update state distribution
    K(i) = P_prior.T * h.T ./ r_tminus_n
    eta_post = ln(alpha_tilde_post / sum_alpha)
    r_post(i) = 1/alpha_tilde_post + 1/sum_alpha

    W_mean_post(i) = W_mean_prior + K * (eta_post-eta_minus)
    P_post(i) = (I-K(i)*h) * P_prior * (1-(r_post(i)/r_prior(i)))

    //Filter Step 5: update noise variance estimate
    u_post = variance( c|alpha_tilde_post )
    u_prior = variance( c|alpha_tilde_prior )
    qt = max( u_post - u_prior, 0)
```

```
    W_mean_prev = W_mean_post(i)
    P_prev = P_post(i)

//Smoothing
lambda = 0
Lambda = 0
for each time-step tau=N,..,1:
    h = t_all(i,:)
    //update state moments calculated during filtering
    W_mean_post_all = W_mean_post(i) - P_post(i)*lambda
    P_post_all = P_post(i) - P_post(i)*Lambda*P_post(i)

    //find the posterior hyperparameters
    eta_post_all = h*W_mean_post_all
    r_post_all = h*P_post_all*h.T
    A(:,i) = (1+exp(eta_post_all))/r_post_all

    //calculate changes to pass on to previous time-steps
    lambda = h.T/r_prior(i)*(eta_post(i)-eta_minus(i)) ...
             + (I-K(i)*h)*lambda
    Lambda = h.T*h/r_prior(i)*(1-(r_post(i)/r_prior(i))) ...
             + (I-K(i)*h)*Lambda*(I-K(i)*h)
return A
```

Bibliography

[Adams et al., 2010] Adams, R. P., Wallach, H. M., and Ghahramani, Z. (2010). Learning the structure of deep sparse graphical models. In Teh, Y. W. and Titterington, D. M., editors, *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 9 of *JMLR Proceedings*, pages 1–8. JMLR.org.

[Attias, 2000] Attias, H. (2000). A variational Bayesian framework for graphical models. In *Advances in Neural Information Processing Systems 12*, pages 209–215. MIT Press.

[Baeza-Yates and Ribeiro-Neto, 1999] Baeza-Yates, R. and Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. Addison Wesley.

[Bashir et al., 2012] Bashir, M., Anderton, J., Wu, J., Ekstrand-Abueg, M., Golbus, P. B., Pavlu, V., and Aslam, J. A. (2012). Northeastern university runs at the TREC12 crowdsourcing track. In *The Twenty-First Text REtrieval Conference (TREC 2012)*. NIST.

[Benediktsson and Swain, 2002] Benediktsson, J. A. and Swain, P. H. (2002). Consensus–theoretic classification methods. *Systems, Man and Cybernetics, IEEE Transactions on*, 22(4):688–704.

[Berger, 1985] Berger, J. O. (1985). *Statistical Decision Theory and Bayesian Analysis*. Springer Series in Statistics. Springer Science+Business Media.

[Bergstra and Bengio, 2012] Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13:281–305.

[Berners-Lee, 2006] Berners-Lee, T. (2006). Linked data. *International Journal on Semantic Web and Information Systems*, 4(2).

[Berners-Lee et al., 2001] Berners-Lee, T., Hendler, J., Lassila, O., et al. (2001). The Semantic Web. *Scientific American*, 284(5):34–43.

[Bierman, 1973] Bierman, G. J. (1973). Fixed interval smoothing with discrete measurements. *International Journal of Control*, 18(1):65–75.

[Bishop, 2006] Bishop, C. M. (2006). *Pattern recognition and machine learning*. Information Science and Statistics. Springer, 4th edition.

[Blattenberger and Lad, 1985] Blattenberger, G. and Lad, F. (1985). Separating the Brier score into calibration and refinement components: a graphical exposition. *The American Statistician*, 39(1):26–32.

[Blei et al., 2003] Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent Dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022.

[Bloodgood and Callison-Burch, 2010] Bloodgood, M. and Callison-Burch, C. (2010). Using Mechanical Turk to build machine translation evaluation sets. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazons Mechanical Turk*, pages 208–211. Association for Computational Linguistics.

[Bordley, 1982] Bordley, R. F. (1982). A multiplicative formula for aggregating probability assessments. *Management Science*, 28(10):1137–1148.

[Bradley and Terry, 1952] Bradley, R. A. and Terry, M. E. (1952). Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345.

[Brier, 1950] Brier, G. W. (1950). Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3.

[Brochu et al., 2008] Brochu, E., de Freitas, N., and Ghosh, A. (2008). Active preference learning with discrete choice data. In *Advances in Neural Information Processing Systems 20*, pages 409–416. MIT Press.

[Brown et al., 2005] Brown, G., Wyatt, J., Harris, R., and Yao, X. (2005). Diversity creation methods: a survey and categorisation. *Information Fusion*, 6(1):5–20.

[Chen et al., 2013] Chen, X., Bennett, P. N., Collins-Thompson, K., and Horvitz, E. (2013). Pairwise ranking aggregation in a crowdsourced setting. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 193–202. ACM.

[Choudrey and Roberts, 2003] Choudrey, R. and Roberts, S. (2003). Variational mixture of Bayesian independent component analysers. *Neural Computation*, 15(1).

[Chu and Ghahramani, 2005] Chu, W. and Ghahramani, Z. (2005). Preference learning with Gaussian processes. In *Proceedings of the 22nd International Conference on Machine learning*, ICML '05, pages 137–144. ACM.

[Clarke, 2003] Clarke, B. (2003). Comparing Bayes model averaging and stacking when model approximation error cannot be ignored. *The Journal of Machine Learning Research*, 4:683–712.

[Dang et al., 2006] Dang, V. D., Dash, R. K., Rogers, A., and Jennings, N. R. (2006). Overlapping coalition formation for efficient data fusion in multi-sensor networks. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, page 635. AAAI.

[Daniel et al., 2009] Daniel, K., Dusza, B., Lewandowski, A., and Wietfeld, C. (2009). AirShield: a system-of-systems MUAV remote sensing architecture for disaster response. In *Systems Conference, 2009 3rd Annual IEEE*, pages 196–200. IEEE.

[Dasgupta, 2004] Dasgupta, S. (2004). Analysis of a greedy active learning strategy. In *Advances in Neural Information Processing Systems 17*, pages 337–344. MIT Press.

[Davis, 1965] Davis, P. J. (1965). *Handbook of Mathematical Functions: with Formulas, Graphs, and Mathematical Tables*, chapter 6, pages 258–259. Dover Publications.

[Dawid and Skene, 1979] Dawid, A. P. and Skene, A. M. (1979). Maximum likelihood estimation of observer error-rates using the EM algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):20–28.

[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):1–38.

[Dietterich, 1997] Dietterich, T. G. (1997). Machine Learning research: four current directions. *AI magazine*, 18(4):97.

[Dietterich, 2000] Dietterich, T. G. (2000). Ensemble methods in Machine Learning. In *Multiple classifier systems*, pages 1–15. Springer.

[Donmez et al., 2010] Donmez, P., Carbonell, J., and Schneider, J. (2010). A probabilistic framework to learn from multiple annotators with time-varying accuracy. In *SIAM International Conference on Data Mining (SDM)*, pages 826–837. Society for Industrial and Applied Mathematics.

[Ebden et al., 2012] Ebden, M., Huynh, T. D., Moreau, L., Ramchurn, S., and Roberts, S. (2012). Network analysis on provenance graphs from a crowdsourcing application. In *Provenance and Annotation of Data and Processes*, pages 168–182. Springer.

[Fawcett, 2006] Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874.

[Fortson et al., 2011] Fortson, L., Masters, K., Nichol, R., Borne, K., Edmondson, E., Lintott, C., Raddick, J., Schawinski, K., and Wallin, J. (2011). Galaxy Zoo: morphological classification and citizen science. *arXiv:1104.5513*.

[Fox and Roberts, 2011] Fox, C. and Roberts, S. J. (2011). A tutorial on variational Bayesian inference. *Artificial Intelligence Review*, pages 1–11.

[Freund et al., 2004] Freund, Y., Mansour, Y., and Schapire, R. E. (2004). Generalization bounds for averaged classifiers. *Annals of Statistics*, pages 1698–1722.

[Freund and Schapire, 1995] Freund, Y. and Schapire, R. (1995). A desicion-theoretic generalization of on-line learning and an application to boosting. In *Computational learning theory, Second European Conference, EuroCOLT*, pages 23–37. Springer Berlin Heidelberg.

[Gallagher and Eliassi-Rad, 2008] Gallagher, B. and Eliassi-Rad, T. (2008). Classification of HTTP attacks: a study on the ECML/PKDD 2007 discovery challenge. In *Center for Advanced Signal and Image Sciences (CASIS) Workshop*.

[Geman and Geman, 1984] Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-6(6):721 –741.

[Genest and Schervish, 1985] Genest, C. and Schervish, M. J. (1985). Modeling expert judgments for Bayesian updating. *The Annals of Statistics*, 13(3):1198–1212.

[Genest and Zidek, 1986] Genest, C. and Zidek, J. V. (1986). Combining probability distributions: A critique and an annotated bibliography. *Statistical Science*, 1(1):114–135.

[Ghahramani and Kim, 2003] Ghahramani, Z. and Kim, H. (2003). Bayesian classifier combination. *Gatsby Computational Neuroscience Unit Technical Report GCNU-T., London, UK*.

[Giacinto and Roli, 2001] Giacinto, G. and Roli, F. (2001). Dynamic classifier selection based on multiple classifier behaviour. *Pattern Recognition*, 34(9):1879–1882.

[Gilks and Wild, 1992] Gilks, W. R. and Wild, P. (1992). Adaptive rejection sampling for Gibbs sampling. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 41(2):337–348.

[Girvan and Newman, 2002] Girvan, M. and Newman, M. E. J. (2002). Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826.

[Givens and Roback, 1999] Givens, G. H. and Roback, P. J. (1999). Logarithmic pooling of priors linked by a deterministic simulation model. *Journal of Computational and Graphical Statistics*, 8(3):452–478.

[Gneiting and Raftery, 2007] Gneiting, T. and Raftery, A. E. (2007). Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378.

[Gorman et al., 2009] Gorman, B., Resseguie, D., and Tomkins-Tinch, C. (2009). Sensorpedia: information sharing across incompatible sensor systems. In *Collaborative Technologies and Systems, 2009. CTS'09, International Symposium on*, pages 448–454. IEEE.

[Haker et al., 2005] Haker, S., Wells, W. M., Warfield, S. K., Talos, I. F., Bhagwat, J. G., Goldberg-Zimring, D., Mian, A., Ohno-Machado, L., and Zou, K. H. (2005). Combining classifiers using their receiver operating characteristics and maximum likelihood estimation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2005*, Lecture notes in Computer Science, pages 506–514. Springer.

[Harris and Srinivasan, 2012] Harris, C. and Srinivasan, P. (2012). Using hybrid methods for relevance assessment in TREC Crowd '12. In *The Twenty-First Text REtrieval Conference (TREC 2012)*. NIST.

[Hasselt, 2010] Hasselt, H. V. (2010). Double Q-learning. In *Advances in Neural Information Processing Systems 23*, pages 2613–2621. MIT Press.

[Haupt et al., 1996] Haupt, G. T., Kasdin, N. J., Keiser, G. M., and Parkinson, B. W. (1996). Optimal recursive iterative algorithm for discrete nonlinear least-squares estimation. *Journal of guidance, control, and dynamics*, 19(3):643–649.

[He and Garcia, 2009] He, H. and Garcia, E. (2009). Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284.

[Hendler, 2001] Hendler, J. (2001). Agents and the Semantic Web. *IEEE Intelligent systems*, 16(2):30–37.

[Herbrich et al., 1998] Herbrich, R., Graepel, T., Bollmann-sdorra, P., and Obermayer, K. (1998). Learning preference relations for information retrieval. In *ICML-98 Workshop: text categorization and machine learning*, pages 80–84. The International Machine Learning Society.

[Heskes, 1998] Heskes, T. (1998). Selecting weighting factors in logarithmic opinion pools. In *Advances in Neural Information Processing Systems 10*, page 266. MIT Press.

[Hinton, 1999] Hinton, G. E. (1999). Products of experts. In *Artificial Neural Networks 1999 (ICANN 99), Ninth International Conference on*, volume 1, pages 1–6. IET.

[Ho, 2002] Ho, T. K. (2002). Multiple classifier combination: lessons and next steps. *Hybrid methods in pattern recognition*, 74(1):171–198.

[Ho et al., 2002] Ho, T. K., Hull, J. J., and Srihari, S. N. (2002). Decision combination in multiple classifier systems. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(1):66–75.

[Hofmann and Puzicha, 1999] Hofmann, T. and Puzicha, J. (1999). Latent class models for collaborative filtering. In *International Joint Conference on Artificial Intelligence, IJCAI 1999*, volume 16, pages 688–693. IJCAI.

[Hovland and McCarragher, 2002] Hovland, G. E. and McCarragher, B. J. (2002). Dynamic sensor selection for robotic systems. In *Proceedings of Robotics and Automation 1997, IEEE International Conference on*, volume 1, pages 272–277. IEEE.

[Hu et al., 2012] Hu, Q., Xu, Z., Huang, X., and Ye, Z. (2012). York university at TREC 2012: Crowdsourcing track. In *The Twenty-First Text REtrieval Conference (TREC 2012)*. NIST.

[Huynh et al., 2013] Huynh, T. D., Ebden, M., Venanzi, M., Ramchurn, S., Roberts, S., and Moreau, L. (2013). Interpretation of crowdsourced activities using provenance network analysis. In *Proceedings of the First AAAI Conference on Human Computation and Crowdsourcing (HCOMP)*. AAAI.

[Ipeirotis et al., 2010] Ipeirotis, P. G., Provost, F., and Wang, J. (2010). Quality management on Amazon Mechanical Turk. In *Proceedings of the ACM SIGKDD Workshop on Human Computation*, pages 64–67. ACM.

[Jacobs, 1995] Jacobs, R. A. (1995). Methods for combining experts' probability assessments. *Neural computation*, 7(5):867–888.

[Jacobs et al., 1991] Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87.

[Jain et al., 1999] Jain, A. K., Murty, M. N., and Flynn, P. J. (1999). Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323.

[Jaynes, 2003] Jaynes, E. T. (2003). *Probability theory: the logic of science*. Cambridge University Press.

[Jazwinski, 1969] Jazwinski, A. H. (1969). Adaptive filtering. *Automatica*, 5(4):475–485.

[Jazwinski, 1970] Jazwinski, A. H. (1970). *Stochastic processes and filtering theory*. Courier Dover Publications.

[Jordan et al., 1999] Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. (1999). An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233.

[Julier and Uhlmann, 1997] Julier, S. J. and Uhlmann, J. K. (1997). New extension of the Kalman filter to nonlinear systems. In *AeroSense'97*, pages 182–193. International Society for Optics and Photonics.

[Kahn, 2004] Kahn, J. M. (2004). A generative Bayesian model for aggregating experts' probabilities. In *Proceedings of the 20th conference on Uncertainty in Artificial Intelligence*, pages 301–308. AUAI Press.

[Kamar et al., 2012] Kamar, E., Hacker, S., and Horvitz, E. (2012). Combining human and machine intelligence in large-scale crowdsourcing. In *Proceedings of the 11th International Conference on Autonomous Agents and Multi-Agent Systems, AAMAS '12*, pages 467–474. International Foundation for Autonomous Agents and Multi-Agent Systems.

[Karger et al., 2011] Karger, D. R., Oh, S., and Shah, D. (2011). Iterative learning for reliable crowdsourcing systems. In *Advances in Neural Information Processing Systems 24*, pages 1953–1961. MIT Press.

[Khintchine, 1929] Khintchine, A. (1929). Sur la loi des grands nombres. *Comtes rendus de lAcadémie des sciences*, 188:477–479.

[Kittler, 1998] Kittler, J. (1998). Combining classifiers: A theoretical framework. *Pattern Analysis & Applications*, 1(1):18–27.

[Krogh and Vedelsby, 1995] Krogh, A. and Vedelsby, J. (1995). Neural network ensembles, cross validation, and active learning. In *Advances in Neural Information Processing Systems 8*. MIT Press.

[Kullback and Leibler, 1951] Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86.

[Kuncheva, 2002] Kuncheva, L. I. (2002). Switching between selection and fusion in combining classifiers: An experiment. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 32(2):146–156.

[Lal and Niyogi, 2013] Lal, A. K. and Niyogi, R. (2013). A multiagent planning approach to model a tele-surgery domain. *International Journal of Intelligent Systems and Applications (IJISA)*, 5(9):27.

[Langdon and Buxton, 2001] Langdon, W. B. and Buxton, B. F. (2001). Genetic programming for combining classifiers. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 66–73. ACM SigEVO.

[Law et al., 2009] Law, N. M., Kulkarni, S. R., Dekany, R. G., Ofek, E. O., Quimby, R. M., Nugent, P. E., Surace, J., Grillmair, C. C., Bloom, J. S., Kasliwal, M. M., et al. (2009). The Palomar Transient Factory: System overview, performance, and first results. *Publications of the Astronomical Society of the Pacific*, 121(886):1395–1408.

[Lee et al., 2009] Lee, H., Grosse, R., Ranganath, R., and Ng, A. Y. (2009). Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 609–616. ACM.

[Lee and Roberts, 2010] Lee, S. M. and Roberts, S. J. (2010). Sequential dynamic classification using latent variable models. *The Computer Journal*, 53(9):1415–1429.

[Leonard, 1977] Leonard, T. (1977). Bayesian simultaneous estimation for several multinomial distributions. *Communications in Statistics-Theory and Methods*, 6(7):619–630.

[Levenberg et al., 2013] Levenberg, A., Simpson, E., Roberts, S., and Gottlob, G. (2013). Economic prediction using heterogeneous data streams from the world wide web. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, ECML PKDD*, Lecture Notes in Computer Science. Springer.

[Lindley, 1983] Lindley, D. (1983). Reconciliation of probability distributions. *Operations Research*, 31(5):866–880.

[Littlestone and Warmuth, 1989] Littlestone, N. and Warmuth, M. K. (1989). The weighted majority algorithm. In *Foundations of Computer Science, 1989., 30th Annual Symposium on*, pages 256–261. IEEE.

[Liu et al., 2012] Liu, Q., Peng, J., and Ihler, A. (2012). Variational inference for crowdsourcing. In *Advances in Neural Information Processing Systems 25*, pages 701–709. MIT Press.

[Liu and Yuan, 2001] Liu, R. and Yuan, B. (2001). Multiple classifiers combination by clustering and selection. *Information Fusion*, 2(3):163–168.

[Lowne et al., 2010] Lowne, D., Roberts, S., and Garnett, R. (2010). Sequential non-stationary dynamic classification with sparse feedback. *Pattern Recognition*, 43(3):897–905.

[Lum et al., 2007] Lum, M. J., Rosen, J., King, H., Friedman, D., Donlin, G., Sankaranarayanan, G., Harnett, B., Huffman, L., Doarn, C., Broderick, T., et al. (2007). Telesurgery via unmanned aerial vehicle (UAV) with a field deployable surgical robot. *Studies in Health Technology and Informatics*, 125:313–5.

[McInerney et al., 2012] McInerney, J., Rogers, A., and Jennings, N. R. (2012). Improving location prediction services for new users with probabilistic latent semantic analysis. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 906–910. ACM.

[Metsis et al., 2006] Metsis, V., Androutsopoulos, I., and Paliouras, G. (2006). Spam filtering with naïve Bayes–Which naïve Bayes? In *Proceedings of Third Conference on Email and Anti-Spam (CEAS)*. CEAS.

[Michalak et al., 2010] Michalak, T., Sroka, J., Rahwan, T., Wooldridge, M., McBurney, P., and Jennings, N. R. (2010). A Distributed Algorithm for Anytime Coalition Structure

Generation. In *Proceedings of the Ninth International Joint Conference on Autonomous Agents and Multi-Agent Systems, AAMAS '10*, pages 1007–1014. International Foundation for Autonomous Agents and Multi-Agent Systems.

[Minka, 2000] Minka, T. (2000). Bayesian model averaging is not model combination. *MIT Media Lab note (7/6/00), http://www.stat.cmu.edu/minka/papers/bma.html*.

[Mo et al., 2013] Mo, K., Zhong, E., and Yang, Q. (2013). Cross-task crowdsourcing. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM.

[Monteith et al., 2011] Monteith, K., Carroll, J. L., Seppi, K., and Martinez, T. (2011). Turning Bayesian model averaging into Bayesian model combination. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 2657–2663. IEEE.

[Neal, 1993] Neal, R. M. (1993). Probabilistic inference using markov chain monte carlo methods. *Technical Report CRG-TR-93-1, Department of Computer Science, University of Toronto*.

[Nellapati et al., 2012] Nellapati, R., Peerreddy, S., and Singhal, P. (2012). Skierarchy: Extending the power of crowdsourcing using a hierarchy of domain experts, crowd and machine learning. In *The Twenty-First Text REtrieval Conference (TREC 2012)*. NIST.

[Opper and Haussler, 1991] Opper, M. and Haussler, D. (1991). Calculation of the learning curve of Bayes optimal classification algorithm for learning a perceptron with noise. In *Computational Learning Theory: Proceedings of the Fourth Annual Workshop, COLT '91*, pages 75–87. Morgan Kaufmann Publishers.

[Parisi, 1988] Parisi, G. (1988). *Statistical field theory*. Addison-Wesley.

[Penny and Roberts, 1999] Penny, W. D. and Roberts, S. J. (1999). Dynamic logistic regression. In *Neural Networks (IJCNN), The 1999 International Joint Conference on*, volume 3, pages 1562–1567. IEEE.

[Psorakis et al., 2011] Psorakis, I., Roberts, S. J., Ebden, M., and Sheldon, B. (2011). Overlapping community detection using Bayesian non-negative matrix factorization. *Physical Review E*, 83(6).

[Quinn et al., 2010] Quinn, A. J., Bederson, B. B., Yeh, T., and Lin, J. (2010). Crowdflow: Integrating machine learning with Mechanical Turk for speed-cost-quality flexibility. *Technical Report HCIL-2010-09, University of Maryland, College Park*.

[Quinonero-Candela et al., 2006] Quinonero-Candela, J., Rasmussen, C. E., Sinz, F., Bousquet, O., and Schölkopf, B. (2006). Evaluating predictive uncertainty challenge. In *Machine Learning Challenges*, Lecture Notes in Computer Science, pages 1–27. Springer.

[Rahwan et al., 2012] Rahwan, T., Michalak, T., and Jennings, N. R. (2012). A hybrid algorithm for coalition structure generation. In *Twenty Sixth Conference on Artificial Intelligence (AAAI-12)*, Toronto, Canada. AAAI.

[Rahwan et al., 2011] Rahwan, T., Michalak, T. P., Elkind, E., Faliszewski, P., Sroka, J., Wooldridge, M., and Jennings, N. R. (2011). Constrained coalition formation. In *Twenty Fifth AAAI Conference on Artificial Intelligence (AAAI)*, pages 719–725.

[Ramchurn et al., 2013] Ramchurn, S. D., Huynh, T. D., Venanzi, M., and Shi, B. (2013). Collabmap: crowdsourcing maps for emergency planning. In *ACM Web Science 2013*. ACM.

[Ranawana and Palade, 2006] Ranawana, R. and Palade, V. (2006). Multi-Classifier systems: Review and a roadmap for developers. *International Journal of Hybrid Intelligent Systems*, 3(1):35–61.

[Raykar and Yu, 2012] Raykar, V. C. and Yu, S. (2012). Eliminating spammers and ranking annotators for crowdsourced labeling tasks. *Journal of Machine Learning Research*, 13:491–518.

[Raykar et al., 2010] Raykar, V. C., Yu, S., Zhao, L. H., Valadez, G. H., Florin, C., Bogoni, L., and Moy, L. (2010). Learning from crowds. *Journal of Machine Learning Research*, 11:1297–1322.

[Rodriguez et al., 2006] Rodriguez, P. A., Geckle, W. J., Barton, J. D., Samsundar, J., Gao, T., Brown, M. Z., and Martin, S. R. (2006). An emergency response UAV surveillance system. In *AMIA Annual Symposium Proceedings*, volume 2006, page 1078. American Medical Informatics Association.

[Russell and Norvig, 2009] Russell, S. J. and Norvig, P. (2009). *Artificial intelligence: a modern approach*. Prentice hall.

[Sarcevic et al., 2012] Sarcevic, A., Palen, L., White, J., Starbird, K., Bagdouri, M., and Anderson, K. (2012). Beacons of hope in decentralized coordination: learning from on-the-ground medical twitterers during the 2010 Haiti earthquake. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, pages 47–56. ACM.

[Schneider, 2004] Schneider, K.-M. (2004). On word frequency information and negative evidence in naive bayes text classification. In *Advances in Natural Language Processing, 4th International Conference, EsTAL 2004*, pages 474–485. Springer.

[Scott et al., 1998] Scott, M. J. J., Niranjan, M., and Prager, R. W. (1998). Realisable classifiers: improving operating performance on variable cost problems. In *Proceedings of the Ninth British Machine Vision Conference*, volume 1, pages 304–315. BMVC.

[Si and Jin, 2003] Si, L. and Jin, R. (2003). Flexible mixture model for collaborative filtering. In *Proceedings of the Twentieth International Conference on Machine Learning*, volume 20 of *ICML '03*, pages 704—711. ACM.

[Simpson et al., 2013] Simpson, E., Reece, S., Penta, A., Ramchurn, G., and Roberts, S. (2013). Using a Bayesian model to combine LDA features with crowdsourced responses. In *The Twenty-First Text REtrieval Conference (TREC 2012), Crowdsourcing Track*. NIST.

[Smith and Lintott, 2010] Smith, A. and Lintott, C. (2010). Web-scale citizen science: from Galaxy Zoo to the Zooniverse. In *Proceedings of the Royal Society Discussion Meeting 'Web Science: A New Frontier'*. The Royal Society.

[Smith et al., 2010] Smith, A. M., Lynn, S., Sullivan, M., Lintott, C. J., Nugent, P. E., Botyanszki, J., Kasliwal, M., Quimby, R., Bamford, S. P., Fortson, L. F., Schawinski, K., Hook, I., Blake, S., Podsadlowski, P., onsson, J. J., Gal-Yam, A., Arcavi, I., Howell, D. A., Bloom, J. S., Jacobsen, J., Kulkarni, S. R., Law, N. M., Ofek, E. O., and Walters, R. (2010). Galaxy Zoo Supernovae. *Monthly Notices of the Royal Astronomical Society*.

[Smucker et al., 2012a] Smucker, M. D., Kazai, G., and Lease, M. (2012a). Overview of the TREC 2012 crowdsourcing track. In *The Twenty-First Text REtrieval Conference (TREC 2012)*. NIST.

[Smucker et al., 2012b] Smucker, M. D., Kazai, G., and Lease, M. (2012b). TREC 2012 crowdsourcing track TRAT task results. In *The Twenty-First Text REtrieval Conference (TREC 2012)*. NIST.

[Spaan and Lima, 2009] Spaan, M. T. and Lima, P. U. (2009). A decision-theoretic approach to dynamic sensor selection in camera networks. In *International Conference on Automated Planning and Scheduling*, pages 279–304. AAAI Press.

[Stigler, 1986] Stigler, S. M. (1986). *The history of statistics: The measurement of uncertainty before 1900*. Harvard University Press.

[Su and Khoshgoftaar, 2009] Su, X. and Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, 2009:4.

[Sutton and Barto, 1998] Sutton, R. S. and Barto, A. G. (1998). *Reinforcement learning: An introduction*, volume 1. Cambridge University Press.

[Tan and Févotte, 2009] Tan, V. Y. F. and Févotte, C. (2009). Automatic relevance determination in nonnegative matrix factorization. In *SPARS'09 - Signal Processing with*

Adaptive Sparse Structured Representations. Inria Rennes-Bretagne Atlantique, Saint Malo, Royaume-Uni.

[Teh et al., 2006] Teh, Y. W., Newman, D., and Welling, M. (2006). A collapsed variational Bayesian inference algorithm for latent Dirichlet allocation. In *Advances in Neural Information Processing Systems 19*, pages 1353–1360. MIT Press.

[Tran-Thanh et al., 2013] Tran-Thanh, L., Venanzi, M., Rogers, A., and Jennings, N. R. (2013). Efficient budget allocation with accuracy guarantees for crowdsourcing classification tasks. In *Proceedings of the Twelfth International Conference on Autonomous Agents and Multi-Agent Systems, AAMAS '13*, pages 901–908. International Foundation for Autonomous Agents and Multi-Agent Systems.

[Tsoumakas et al., 2004] Tsoumakas, G., Angelis, L., and Vlahavas, I. (2004). Clustering classifiers for knowledge discovery from physically distributed databases. *Data & Knowledge Engineering*, 49(3):223–242.

[Tulyakov et al., 2008] Tulyakov, S., Jaeger, S., Govindaraju, V., and Doermann, D. (2008). Review of classifier combination methods. *Machine Learning in Document Analysis and Recognition*, pages 361–386.

[Turner and Sahani, 2010] Turner, R. E. and Sahani, M. (2010). Two problems with variational expectation maximisation for time-series models. In *Inference and Learning in Dynamic Models*. Cambridge University Press.

[Venanzi et al., 2013] Venanzi, M., Rogers, A., and Jennings, N. R. (2013). Crowdsourcing spatial phenomena using trust-based heteroskedastic Gaussian processes. In *Proceedings of the First AAAI Conference on Human Computation and Crowdsourcing (HCOMP)*. AAAI.

[Vieweg et al., 2010] Vieweg, S., Hughes, A. L., Starbird, K., and Palen, L. (2010). Microblogging during two natural hazards events: what Twitter may contribute to situ-

ational awareness. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1079–1088. ACM.

[Wan and Van Der Merwe, 2000] Wan, E. A. and Van Der Merwe, R. (2000). The unscented Kalman filter for nonlinear estimation. In *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000, AS-SPCC, The IEEE 2000*, pages 153–158. IEEE.

[Wang et al., 2010] Wang, S., Chen, H., and Yao, X. (2010). Negative correlation learning for classification ensembles. In *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pages 1–8. IEEE.

[Watkins and Dayan, 1992] Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine learning*, 8(3-4):279–292.

[Watkins, 1989] Watkins, C. J. C. H. (1989). *Learning from delayed rewards*. PhD thesis, University of Cambridge.

[Weiss and Provost, 2001] Weiss, G. M. and Provost, F. (2001). The effect of class distribution on classifier learning: an empirical study. *Technical Report ML-TR-44, Department of Computer Science, Rutgers University*.

[West and Harrison, 1997] West, M. and Harrison, J. (1997). *Bayesian forecasting and dynamic models*. Springer.

[Wolpert, 1992] Wolpert, D. H. (1992). Stacked generalization. *Neural networks*, 5(2):241–259.

[Wong and Yao, 1988] Wong, S. K. M. and Yao, Y. Y. (1988). Linear structure in information retrieval. In *Proceedings of the 11th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '88, pages 219–232, Grenoble, France. ACM.

[Woods et al., 2002] Woods, K., Kegelmeyer Jr, W. P., and Bowyer, K. (2002). Combination of multiple classifiers using local accuracy estimates. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(4):405–410.

[Wooldridge and Jennings, 1995] Wooldridge, M. and Jennings, N. R. (1995). Intelligent agents: theory and practice. *The Knowledge Engineering Review*, 10(02):115–152.

[Yan et al., 2011] Yan, Y., Fung, G. M., Rosales, R., and Dy, J. G. (2011). Active learning from crowds. In *Proceedings of the 28th International Conference on Machine Learning*, ICML '11, pages 1161–1168.

[Zhang et al., 2012] Zhang, C., Zeng, M., Sang, X., Zhang, K., and Kang, H. (2012). BUPT_PRIS at TREC 2012 crowdsourcing track 1. In *The Twenty-First Text REtrieval Conference (TREC 2012)*. NIST.

[Zhang, 2004] Zhang, H. (2004). The optimality of naive bayes. In *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2004*. AAAI Press.

[Zilli et al., 2013] Zilli, D., Parson, O., Merrett, G., and Rogers, A. (2013). A hidden markov model-based acoustic cicada detector for crowdsourced smartphone biodiversity monitoring. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence, IJCAI 2013*. IJCAI.