UNIVERSITY OF SOUTHAMPTON

FACULTY OF PHYSICAL AND APPLIED SCIENCES

Electronics and Computer Science

Online Machine Learning for Combinatorial Data

by

Shaona Ghosh

Thesis for the degree of Doctor of Philosophy

March 2016

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF PHYSICAL AND APPLIED SCIENCES
Electronics and Computer Science

Doctor of Philosophy

ONLINE MACHINE LEARNING FOR COMBINATORIAL DATA

by Shaona Ghosh

With an ever increasing demand on large scale data, difficulties exist in terms of processing and utilising the information available. In particular, making decisions based upon sequentially acquired data where only limited information is initially known, is an important problem. Often the input data in such problems have a complex combinatorial structure, for example consider an internet advertising system that manages advertisement placement over a network of websites. The ways of placing m different advertisements on n websites with replacement, is an exponential number of m^n possible combinations that scales badly with large n. As a combinatorial problem, the data can be manipulated within a frequently occurring computational object called graph, allowing the structure to be exploited for intelligent automatic processing. Traditionally, machine learning techniques require a separate initial training phase before predictions can occur on unseen data. However, the sequential nature of some problems necessitate real-time prediction, thereby making many existing techniques unsuitable. Online learning is a field of machine learning that has an ensemble of algorithms that learn from sequential streaming data, where the learner cannot control or influence the data collection procedure. Although these existing online methods have theoretical guarantees on performance, in the context of combinatorial complexity of graphical structures they are not yet fully matured. In this thesis, a series of algorithms that attempt to overcome the shortcomings of existing online algorithms are presented. The discrete graphical model, called the Ising model, is explored to develop online approximation algorithms for label prediction. A deterministic approximation algorithm with sequential guarantee is developed, by capturing the persistent structures of maximum flows and minimum cuts in the network and an efficient enumeration of all label consistent minimum cuts. Novel mistake bounds are provided that improve and match previous performance bounds in the literature. Additionally, a variational approximation technique using mean field approximation is built for online prediction of multi-class labelling on the Ising model. An online sequential action selection algorithm for the limited feedback setting (bandit feedback) and side information is developed with a linear programming relaxation of the classic maximal flow problem. Finally, the multiple objective optimization problem with conflicting objectives and full feedback is studied and an online algorithm is built that outperforms the traditional approaches under similar assumptions.

Contents

List of Figures

List of Tables

Declaration of Authorship

I, Shaona Ghosh , declare that the thesis entitled *Online Machine Learning for Combinatorial Data* and the work presented in the thesis are both my own, and have been generated by me as the result of my own original research. I confirm that:

- this work was done wholly or mainly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- parts of this work have been published as: (Ghosh and Prügel-Bennett, 2015a), (Ghosh and Prügel-Bennett, 2015b), (Ghosh et al., 2013) and (Herbster et al., 2015)

Signed:.....

Date:.....

Acknowledgements

My immense gratitude goes out to my supervisor Dr. Adam Prügel-Bennett for his wholehearted confidence in me. He allowed me complete freedom in following my interests, while looking out for me in case I hit a dead end. I thank him for his care, patience and hopes in me. I am also thankful to Prof. Mahesan Niranjan and all the members of the Vision, Learning and Control group. I would like to thank the department of ECS and the faculty for sponsoring my PhD studies. The Iridis high performance cluster maintained by the Computational Modelling Group has been a very useful resource for my experiments. I would like to thank Dr. Mark Herbster at the Computer Science and Machine Learning department(CSML) in University College London (UCL) for inviting me to visit his lab for a greater part of my PhD. My gratitude extends to his student Stephen and colleague Andrew for their support. Personally, I am grateful to my wonderful close friends back in India who have supported me throughout this journey - Jayeeta and Kollol. While many miles away from home, they cared for me every but and looked out for me in case I needed any form of support. I could not have achieved this without their help. I am forever grateful to Elaine and John for being supportive and understanding and for being like my family here in the UK. My family - my sister for believing in me and for having the faith in me and for being my inspiration and hope. My father for taking care of my mother towards the end of my PhD, when she was alone. My lovely nephew who came to this world when I started my PhD and has been my source of amusement and happiness ever since. My mother for her emotional and financial sacrifices, for having the courage to send me 4000 miles away from home to pursue my passion of learning and for believing in me and always standing by me. I am what I am today because of her. She has been my idol, teacher and my friend. Last but not least, I am forever grateful to my best friend, my mentor and the love of my life Chris Lovell, for being my pillar of support. For bearing all the ups and downs during this crazy journey. For believing in me when I had lost hope, for being the voice of reason when I was lost. For putting up with my weird sleeping habits, working into late hours, stressful weeks. Thank you to all of you. I am indebted to all of you for life.

Chapter 1

Introduction

Combinatorics is a branch of discrete mathematics that has many interesting applications in everyday life. With strong connections to computer science, statistics and algebra, combinatorial problems constitute arranging discrete structures such that they satisfy a certain pattern while being able to provide solutions such as validating if the arrangement is feasible, counting of the number of possible patterns or finding the optimal arrangement among others (Lawler, 1985; Cameron, 2007). Some examples of combinatorial problems include the assignment and allocation problems like the knapsack problem, load balancing, travelling salesman problem, transportation and network flow problems among others. Algorithms addressing such combinatorial problems broadly fall into two categories that of, optimization and approximation. One example of combinatorial optimization is finding the best trade-off solution among multiple conflicting objectives.

The fundamental elements of combinatorics are the graphs, which are the structured, flexible and powerful tools that model pairwise relationships among objects in the real-world. The focus of this thesis is to study such combinatorial structures with a purpose of developing machine learning algorithms for them. The central goal of a machine learning algorithm is to produce a model of an observable system or behaviour, which can be used to make future predictions and better decisions. Machine learning is used in almost all applications today where there is an abundance of data that needs to be processed, analysed, and made valuable future predictions from, that can impact the society. An example of such applications of machine learning in the medical sector, is in predicting stroke in patients with atrial fibrillation (Letham et al., 2015) or using deep learning for protein structure prediction (Wang et al., 2016). Whilst combinatorial problems have a different ultimate objective as mentioned in the previous paragraph, there is a significant overlap between the two fields of study where predictive algorithms can be built for combinatorial data by capturing the rich structural information. Networked data are ubiquitous in this era of the social, economical and technological revolution resting on the backbone of the internet; where the data available has a rich structure of

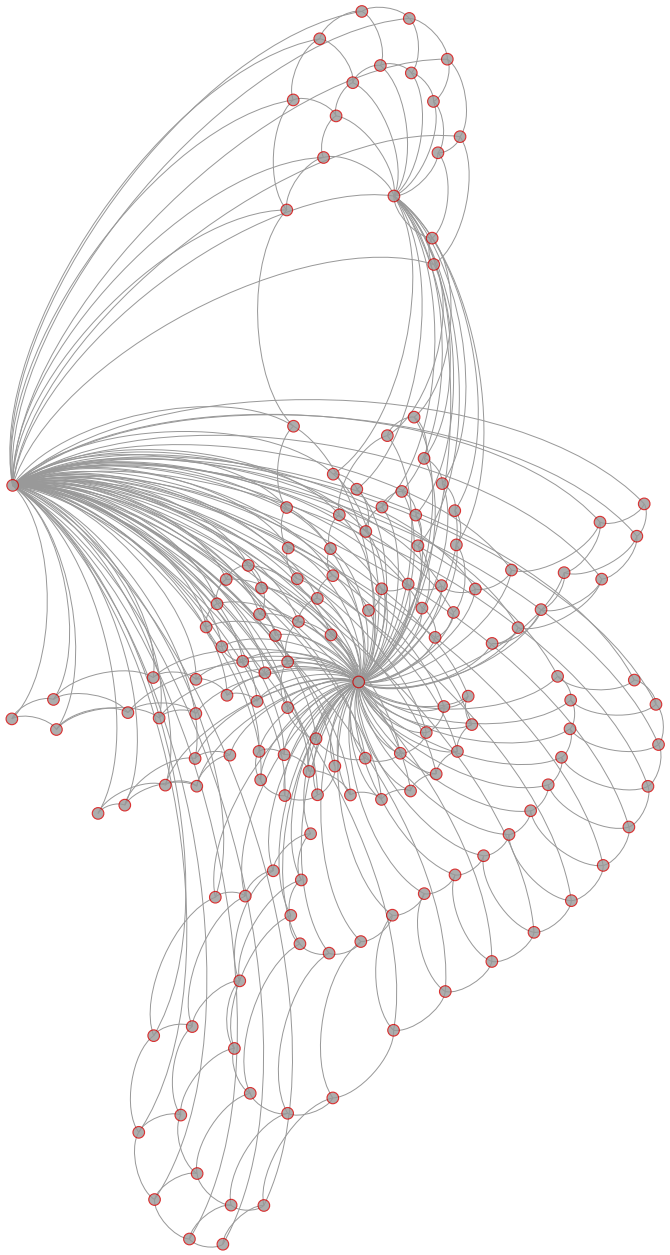


Figure 1.1: An example network constructed from combinatorial concentric circles structured dataset called `Circles` in Chapter 3.

inter-relationships among its instances, resulting in either an extrinsic graphical structure or dependencies that can be modelled as a graph. Figure 1.1 shows an example network constructed from a dataset. With the spread of mobile phones, sensors, embedded devices and industrial robots, the ability to collect and generate interdependent data is on an all time high. This enormity of data is supplemented with social software systems like LinkedIn, Twitter and many others, that connect people around the globe as a network, allowing them to share the data they collect, generate or distribute data

in real-time. Further, with the 'Internet of things', appliances, vehicles and wearable technologies can communicate with each other. The resulting trend is better and bigger ways of "collecting, creating, managing and storing of data" also known as Big Data (Executive Office of the President, 2014). With such a phenomenal rise in data, there is the imperative requirement for computational techniques to analyse, predict and make better decisions from the data. The modern machine learning paradigm provide powerful algorithms that not only analyze the vast amount of data but make accurate future predictions (Ghahramani, 2015; LeCun et al., 2015). However, most of the machine learning methods do not take into account the robustness of the algorithms, which is an important criterion for today's dynamic data. Further, when the input data size is that of Big Data, in the order of millions of data records; having a separate training phase as most conventional machine learning methods practise, is not desirable; predictions need to be made on the fly in real-time.

For example, consider the social network of friends grouped by evolving blog communities with constantly shifting trends or financial markets with constantly varying buy/sell pricing. Both of these are examples of very non stationary systems where algorithms need to dynamically learn to predict trending topics, or which commodity to buy or sell, given the current context and the past. Some might need to simultaneously predict which stock to sell and at what price. Naturally, the questions that arise are: how do the algorithms predict on the fly without having a separate training phase and a prediction phase? Additionally, more complex questions are: how does the learning algorithm capture the sequential nature of events? Can the algorithms guarantee that they will be efficient regardless of any sequence of data they see, in any order? Are these methods adaptive enough to predict based on real-time changes?

In this thesis, theoretically and empirically motivated machine learning algorithms are developed, that address problems like these. Specifically, extensions to a particular machine learning framework called 'online learning' are made, that provide robustness and worst case theoretical guarantees besides being fairly efficient in practice, for sequential decision making, for data on the fly. The online learning techniques are adaptive sequential algorithms that learn and make predictions in real-time. In particular, online learning techniques for data that has an extrinsic graphical structure or that can be represented as a graph with rich combinatorial structure are designed. Further, algorithms for problems where the feedback from the dynamic environment varies from full feedback to partial feedback settings are developed, with a motivation of building robust algorithms. Specific focus is given to semi supervised learning where the algorithm needs to simultaneously learn from a few training and majority of test examples at the time of prediction. Broadly, the goal of this thesis is to create online machine learning methods for structured data that are robust in their performance under various dynamic settings.

1.1 Online Learning Framework

Online learning is a sequential prediction framework where the learner observes elements belonging to a sequence y_1, y_2,..., y_T, while playing a game with the environment within which it functions, for a length of time T. The learning proceeds in a series of trials and at every round $t = 1, 2, ...$, the learner predicts the t-th symbol y_t of the sequence based on the previous $t-1$ observations (Cesa-Bianchi and Lugosi, 2006), before the true prediction is revealed by the environment. In contrast to the standard batch learning, here the learner generates the training set as it plays the game (Seldin et al., 2011). The other contrast to batch learning is that no assumptions are made about the generative process for the outcomes y_t (could be deterministic, stochastic, or adversarial). The classical machine learning statistical theory of sequential prediction uses the loss function (risk) to measure the discrepancy between predicted value and the true outcome in order to evaluate the learner. To avoid assumptions on how the outcomes are generated, the online learning framework does not use a single fixed baseline for comparison, instead it uses a class of reference forecasters or experts. The experts provide their predictions to the learner before the next outcome is revealed. The learner predicts by using the expert's prediction so that the cumulative loss of the learner is close to the best expert in the class. The difference between the learner's cumulative loss and that of an expert is called regret; a measure of how much the learner regrets in not following the advice of a particular expert in hindsight. The abstract expert could be thought of as a black box of unknown computational power, a hypothesis, another machine learning algorithm or a human expert (Herbster and Warmuth, 1998). The learner maintains a weight on every expert; weight implies confidence of the learner in that expert's prediction, the weight is decreased in every round as a function of the loss of the expert (Herbster and Warmuth, 1998).

Online learning can function in two different settings in terms of the information or feedback available to the learner from the environment; full feedback and partial feedback. The learning with experts discussed above is the full information case, where the learner observes the full loss function l_t for trial t as feedback and can use this loss function in choosing its action by employing first order or second order optimization. In the partial information case also known as the bandit setting, feedback at time t, consists of scalar value $l_t(a_t)$, where a_t is the action selected by the learner before suffering the loss enabling only zeroth order optimization to be used (Rakhlin et al., 2009).

When the learning proceeds in an adversarial environment, the adversary can deliberately try to fool the learner by making the cumulative loss of the on-line learning algorithm arbitrarily large by asking the learner to predict on the same data point every round or by providing a completely wrong value as the true prediction (Shalev-Shwartz, 2012). To restrict the power of the adversary, it is assumed that all outcomes are generated from some target mapping $h^* : X \to Y$. Also h^* belongs to a fixed hypotheses

set $\mathcal{H}$ that is known to the learner; both h^* and the sequence of data points to predict on are chosen by the adversary. Under some relaxations, the learner needs to make minimal number of mistakes with its predictions. On further relaxation of the problem, the learner is required to be competitive with the best predictor from the set $\mathcal{H}$ and acquire as low regret as possible (Shalev-Shwartz, 2012). As mentioned earlier, regret measures how sorry the learner is in not following the advice of some hypothesis $h^* \in \mathcal{H}$ which can be formally defined as:

$$R_T(h^*) = \sum_{t=1}^{T} l(p_t, y_t) - \sum_{t=1}^{T} l(h^*(x_t), y_t), \tag{1.1}$$

where l is the loss function measuring difference between predicted value and true value, p_t is the prediction of the learner, y_t is the true value, $h^*(x_t)$ is the prediction of the best expert in hindsight, and T is the number of examples seen by the learner so far. For non-trivial algorithms it is possible to obtain regret that is sub-linear in the number of rounds played.

Meaningful information can be obtained from the regret by applying assumptions on the feedback received from the adversary. More often in the framework of online learning, to make some progress and for simpler analysis, relaxations are made to the adversary's power as discussed previously. It is assumed that the adversary is oblivious to the learner's strategy. While this approach of having a restricted adversary is more widely adopted, realistic settings do not conform to this assumptions on the adversary's power. In a realistic setting the action of the learner changes the environment adversely for achieving simultaneous objectives and affects the loss suffered by the learner. To model such situations, the adversary can be thought of adopting a stronger role. Several papers have addressed a non-oblivious adversary as evident from literature such as learning with adaptive rates (Auer et al., 2002b), randomization (Hutter and Poland, 2004, 2005), regularization (Cesa-Bianchi and Lugosi, 2006; Rakhlin et al., 2009).

1.2 Thesis Goals and Contributions

The focus of this thesis is to develop online learning algorithms for prediction over graphs and differential feedback settings. At the beginning, the notion of the performance measure inside the online learning framework is introduced. This is followed by introducing the full feedback and limited feedback scenarios. Both the settings address real-life based problems that demand such feedback requirements. Then, inside the semi-supervised structure of learning, the concept of online graph labelling is introduced. Specifically, emphasis is placed on a very important graphical model, that provides an efficient structural insight into the graph labelling problem. In particular, the thesis statement can be stated as

1. Review and build the notion of "cut" as a regularizer that induces a specific distribution on the vertices of a graph and develop sequential algorithms capable of learning from labelled and unlabelled data, while predicting the label of an unlabelled vertex.

2. Develop adaptive learning procedures for other constrained scenarios like full and partial feedback, to enable efficient sequential action selection and multiple objective optimization.

More specially, the central goals of the thesis as listed above are divided into sub goals that span individual chapters in order to:

1. Build semi-supervised online classification algorithms for graph labelling problems, constrained by large sparse graphs, with a few available labels and majority unknown labels, through (a) deterministic approximation algorithm (b) probabilistic variational approximation algorithm.

2. Extend the online algorithms to work under the partial feedback settings, that perform a sequential action selection problem within a graph labelling problem inspired by real life examples.

3. Develop online algorithm for the full feedback setting, based on predicting a Pareto optimal front on multiple objectives.

   An underlying unified objective that is meant to be achieved through all the above goals is to build sequential, adaptive learning algorithms, that are robust to (a) large sparse datasets having rich structure (b) multiple feedback settings motivated from real-life examples and to provide an intuitive, theoretical grounding along with an empirical assessment of standard benchmarks, for the algorithms that are designed in this thesis.

   Essentially, this objective is made possible by the efficient use of the structural information in the data, which forms the core of the main tool for learning and prediction. In one line, this thesis studies the robustness of online algorithms on structured data under a variety of learning conditions inspired from real-life applications.

1.3 Previous Publications

The contributions of this thesis led to the following publications:

- Ghosh, S. and Prügel-Bennett, A. (2015a). Ising bandits with side information. In *Machine Learning and Knowledge Discovery in Databases*, volume 9284 of *Lecture Notes in Computer Science*, pages 448–463. Springer International Publishing

- Ghosh, S. and Prügel-Bennett, A. (2015b). Online mean field approximation for automated experimentation. In *Proceedings of The 4th Workshop on Machine Learning for Interactive Systems*, volume 43, pages 31–35. Journal of Machine Learning Research

- Ghosh, S., Lovell, C. J., and Gunn, S. R. (2013). Towards pareto descent directions in sampling experts for multiple tasks in an on-line learning paradigm. In *Proceedings of the AAAI Spring Symposium Series of Lifelong Machine Learning 2013*, volume 13 of *SS-13-05*. AAAI Press

- Herbster, M., Pasteris, S., and Ghosh, S. (2015). Online prediction at the limit of zero temperature. In *Advances in Neural Information Processing Systems 28*, pages 2917–2925. Curran Associates, Inc

1.4 Organization of the Thesis

Chapter 2, provides a background of the field of research in this thesis and lays the foundation for the work discussed in the succeeding chapters. It begins with an overview of the field of machine learning leading on to online learning. Specifically, the methods on how to treat the data with intrinsic and extrinsic graphical structure are highlighted. This leads on to a description of the graph labelling background under a semi-supervised setting. The chapter is concluded with a review of classical methods for improving performance on such problems by adopting the online learning methodology.

Chapter 3 is joint work with Mark Herbster and Stephen Pasteris, first published in (Herbster et al., 2015). This chapter investigates the online label prediction at the limit of zero temperature on an Ising model (Ising, 1925) through a deterministic approximation technique. My first main contribution in this published work is in building simulation experiments to validate our intuition and formulate our idea. The second main contribution is in developing and implementing our approximation algorithm from scratch and other competitor algorithms in parallel. The third main contribution is in conducting of the extensive empirical evaluation of our methods on multitude of datasets, followed by validation and analysis against the state-of-the-art. Mark Herbster is responsible for the main idea, and the main author for the published version of the paper. Both Mark Herbser and Stephen Pasteris are responsible for the theoretical analysis of our method.

Chapter 4 studies the variational mean field approximation technique for online semi supervised graph labelling, published in (Ghosh and Prügel-Bennett, 2015b). A decoupled factored distribution is used to approximate the posterior distribution at low temperatures on an Ising (Ising, 1925) model.

In Chapter 5, the partial feedback environment of online bandits is studied, within the contextual game of latent semi-supervised graph labelling. This work was first published in (Ghosh and Prügel-Bennett, 2015a). The online bandits serve as a combinatorial action set for the learning algorithm to play the sequential decision making game.

Chapter 6 investigates a slightly different but extremely important combinatorial problem of that of multiple objective optimization in the light of online learning with experts or full feedback. The work addresses the important requirement of adaptive sequential methods in the context of our case study. This work is published in (Ghosh et al., 2013).

Chapter 2

Online Learning Perspective

Understanding the connections between different categories of learning, under various assumptions is imperative for the efficient use of machine learning to solve real problems. The machine learning field of study is primarily a data driven field. Not every machine learning solution is suited for every realm of application. Being able to manipulate the data efficiently, interpret, analyze and make valuable predictions from it, depend on a multitude of factors: nature of the problem that the algorithm is trying to solve, the data itself (structured, noisy, sequential, generative), feedback from the environment, constraints, computational power among others.

In this chapter, we draw the connections between the interconnecting elements of machine learning within which our work functions. In addition to reviewing the standard approaches, we elucidate how they are related to each other and why they are important in the context of the problems we are interested in. This chapter lays the foundation for our work in the chapters that follow.

2.1 Machine Learning Techniques

Standard machine learning approaches fall within the purview of two schools of learning: inductive and transductive. Consider a function h that maps input (instance) $\boldsymbol{x}$ to output (label) y by $y = h(\boldsymbol{x})$ where $y \in \{-1, 1\}$. Given an input sequence, $\{(\boldsymbol{x}_i, y_i) : i = 1, \cdots, n\}$; the objective of inductive learning is to deduce the function h and then predict the label y_{n+1} of unseen test instance $\boldsymbol{x}_{n+1}$ (Delalleau et al., 2005; Chapelle et al., 2006a,b; Joachims, 1999; Bousquet, 2002), where n is the number of instances. An example of this type of learning is in pattern recognition, when the algorithm trains on training image patterns and predicts on unseen test patterns. The other name for this method of learning is supervised learning. Examples of supervised learning are neural networks, decision trees, support vector machines and linear regression among

others. The limitations of this method is in the requirement of a separate training phase and the availability of training labels, that is expensive to obtain. If with the similar sequence, there are no training labels, and the goal of the learning algorithm is to simply learn the mapping function h without predicting the labels as $\{(\boldsymbol{x}_i, y_i) : i \in 1, \cdots, n\} \cup \{\boldsymbol{x}_1^{'}, \cdots, \boldsymbol{x}_m^{'}\} \rightarrow h$, then the type of learning is induction with unlabelled data or unsupervised learning. $\boldsymbol{x}_1^{'}$ is the unlabelled data. Examples of unsupervised learning are clustering (k-means), dimensionality reduction (PCA).

Semi Supervised learning is the middle-ground between supervised and unsupervised where the goal is to learn from the labelled and unlabelled data, then predict, $(\boldsymbol{x}_i, y_i) : i \in 1, \cdots, n\} \cup \{\boldsymbol{x}_1^{'}, \cdots, \boldsymbol{x}_m^{'}\} \rightarrow (y_1^{'}, \cdots, y_m^{'})$, where $(y_1^{'}, \cdots, y_m^{'})$ (Chapelle et al., 2006a; Grandvalet and Bengio, 2004; Chapelle et al., 2006b; Joachims, 1999; Bousquet, 2002) are the predictions. This type of learning is also known as transductive learning. This is particularly useful in real-life applications where a few labels can be obtained but there are many unlabelled instances and a separate training phase is expensive. The methods in this setting, can efficiently use unlabelled data for better prediction. The underlying assumptions of this framework are (a) there is information in the data distribution (b) the data is separable (Chapelle et al., 2006b; Joachims, 1999; Ghahramani, 2012).

In a sense, supervised learning is the general setting of semi-supervised learning as it learns a generalized mapping that maps previously unseen data. In this thesis, we begin with the semi-supervised structure in the first three chapters and transition to the generalized full supervised learning in the last chapter.

2.2 Semi Supervised Learning over Graphs

The graph based semi supervised learning constructs a graph from the dataset by connecting similar data-points. The observed and unobserved labels are random variables on the vertices of the graph. Each vertex corresponds to a data instance. In other words the graph is a Markov Random Field (MRF). The graphical structure may be intrinsic in the data, as in the case of protein interaction network or a social network of friends, alternatively the graphical structure can be built via the distance $d(i, j)$ between datapoints i and j. A sparse unweighted graph can be generated from linking k-nearest neighbours. A complete weighted graph is constructed from having a weighted edge w_{ij} as $w_{ij} = \frac{1}{d(i,j)}$. The intuition of semi-supervised labelling on the graph is that information from the labelled vertices propagates to the unlabelled vertices. The graph is the encoding of this intuition that linked vertices must be similarly labelled. Some graph based techniques estimate a function f on the graph such that it satisfies the available labels and is smooth over the entire graph. Typically, most semi supervised methods use some form of regularization, where the specific form depends on the graph construction

and the problem itself (Belkin et al., 2004; Zhu, 2005; Chapelle et al., 2006b; Herbster and Lever, 2009; Herbster et al., 2005).

For example, in (Blum and Chawla, 2001), a *st*-cut problem is defined with an objective to deduce the minimum set of edges which when removed from the graph stops the flow from source (postive labelled vertices) to sink (negative labelled vertices). Interestingly, the minimum set function is the mode of the Markov random field with boolean labels. The loss function is a quadratic loss with a weight of infinity: $\infty \sum_{i \in L}(y_i - y_{i|L})^2$; with L as the number of available labels, such that the labelled vertices are considered as ground truth and fixed in their values. The regularizer is given by (Zhu, 2005)

$$\frac{1}{2}\sum_{i,j} w_{ij}|y_i - y_j| = \frac{1}{2}\sum_{i,j} w_{ij}(y_i - y_j)^2.$$

We will refer to similar minimum set of edges or minimum cut based regularization in the context of the problems we study throughout the thesis. Eventually, in (Blum and Chawla, 2001), the authors minimize (Zhu, 2005)

$$\infty \sum_{i \in L}(y_i - y_{i|L})^2 + \frac{1}{2}\sum_{i,j} w_{ij}(y_i - y_j)^2$$

where the labels $y_i \in \{0, 1\}, \forall i$. The drawback of this approach is the classification does not provide any confidence bounds (the maximum a posteori of the posterior is computed rather than the marginal probabilities) (Zhu, 2005).

In order to find the labelling of a vertex, the proper method is to compute the marginal probability of the vertex. However, this is a hard inference problem. When the labels are from two classes as in the minimum cut method, the MRF is called a discrete MRF or Boltzmann Machines. As we will discuss later, our interest is also in the areas of discrete MRFs in this thesis. We discuss the difficulty in inference within this problem in the later part of this chapter. In (Zhu et al., 2003), the authors were restricted in terms of the sampling method. In (Getz et al., 2006), the authors employ Multi canonical Monte Carlo method instead if Metropolis-Wang method used by (Zhu et al., 2003).

2.3 Online Learning and Batch Learning

Traditional machine learning methods conduct learning in two phases, training and testing (as in the case of supervised methods discussed above). The online learning framework of algorithms learn from data on the fly (there is no separation between training and test data) and predicts on the data seen. The algorithms do not see the data beforehand; and receive one datapoint at a time sequentially. The objective of such algorithms is to be close to the hypothetical best predictor over the entire time horizon (Cesa-Bianchi and Lugosi, 2006; Bubeck and Cesa-Bianchi, 2012). Contrary to

traditional machine learning, in online learning, no guarantees can be made about the loss, as the performance is with respect to the optimal algorithm in hindsight. As seen in the preceding chapter, the performance is measured in terms of the notion of 'regret' or how far the algorithm is in its performance from the best given as

$$R_T(h^*) = \sum_{t=1}^{T} l(p_t, y_t) - \sum_{t=1}^{T} l(h^*(x_t), y_t), \tag{2.1}$$

where, p_t is the prediction of the online algorithm at trial t, R is the regret of the algorithm with respect to the theoretical optimal algorithm h^*. Making effective progress in learning and storing the improvement is more memory efficient that storing all previous samples seen. The objective of the online algorithms is to ensure that the regret reduces to zero sublinearly.

On the other hand, batch learning problems in classic machine learning, look at the training samples and return a hypothesis; the environment is assumed to be stationary and the examples are independent and identically (i.i.d) drawn from a probability distribution. The goal of such algorithms is to minimize the expected error of the function h estimated as $\mathbb{E}_{y\sim q} l(h, y)$, where q is the distribution, the instances are drawn from. When the environment is dynamic, there are large quantities of data that have no correlations, meaning that the statistical assumptions on the data generation process do not make any sense. There is no stationary distribution and adaptive methods are required for decision making up against the adversarial nature or environment (Rakhlin, 2008; Cesa-Bianchi and Lugosi, 2006; Bubeck and Cesa-Bianchi, 2012). In reality, in online learning, there are no assumptions on the data; the nature can be adversarial, deterministic or stochastic.

2.4 Online Learning Algorithms

Typically, there are three broad set of learning paradigms inside online learning based on the feedback available. These three sets are: full; partial; and limited feedback. The full information case is known as online learning with experts. Partial information is alternatively known as semi-bandit while limited feedback is the bandit feedback. In general, the bandit structure is the worst case scenario and generalizes the other feedback settings.

In contrast to minimizing expected loss, online algorithms focus of reducing the cumulative loss gap with respect to the best predictor as we see in 2.1. Without any assumption on the data generation process, there is no baseline to compare with; in the experts learning, a set of reference predictors are considered as experts, the learner uses the expert's advice before prediction. The performance of the learner is measured in comparison to the best expert. The notion of regret here is interpreted as how much

the forecaster regrets in not following the particular expert's advice (Cesa-Bianchi and Lugosi, 2006; Freund and Schapire, 1997, 1999, 1996). It is important to note that once the feedback is received, since every expert ξ_i 's prediction is known, they can be evaluated on their prediction quality with respect to the optimal expert. The expert can be a black box predictor, a hypothesis (regressor, classifer) or a human expert. The strategies of the algorithms in this type of learning, is in bounding the regret by paying close attention to the size and structure of the experts (Cesa-Bianchi and Lugosi, 2006). Some famous experts algorithm are the 'Prediction with Expert Advice', 'Halving Algorithm' (Littlestone, 1988) and 'Exponential Weights or Weighted Majority' (Littlestone and Warmuth, 1994; Agarwal, 2011). In all the algorithms, $\mathcal{X}$ is the set of instances, $\hat{y}_t$ is the learning algorithm's prediction, y_t is the true prediction revealed by the environment, l is a convex loss function. There are N experts and the time horizon is denoted by T. The basic online prediction with experts algorithm is listed in Algorithm 1. The

Algorithm 1 Prediction with Expert Advice Online Algorithm.

for $t = 1, \ldots, T$ **do**
  Receive: Instance $x_t \in \mathcal{X}$
  Receive: Expert Predictions $\xi^1(x_t), \cdots, \xi^N(x_t) \in \{0, 1\}$
  Predict : $\hat{y}_t \in \{0, 1\}$
  Receive: True label $y_t \in \{0, 1\}$
  Incur : Loss $l(y_t, \hat{y}_t)$
end

Halving algorithm is as follows. It assumes that there is one expert that is always going to give the correct label over all instances. The idea of the simple yet elegant algorithm is that at every update only the consistent experts are retained for continuing with prediction in the next rounds. In this, a probability distribution or weights $\boldsymbol{w}$ are maintained over the entire set of experts. The algorithm is listed in Algorithm 2. The weighted majority algorithm is derived from the Halving algorithm for applications to problems where there is no single expert that is always right. In weighted majority algorithm as shown in Algorithm 3, an equal weight of 1 is assigned to each expert. In the update step, the weight of the predictors are reduced using a multiplicative update step based on a parameter η, when they make a mistake.

All the algorithms discussed above, describe the 0/1 loss scenario. For other convex loss functions, the update step, becomes $w^i_{t+1} = w^i_t \mathrm{e}^{-\eta l(\xi^i_t, y_t)}$. The regret bound in the case of $0-1$ loss is also known as the mistake bound. A mistake bound is defined as,

$$M(A, \mathcal{H}) = \max_{h \in \mathcal{H}, T, x_{1:T}} \sum_{t=1}^{T} \left[\!\!\left[h_t(x_t) \neq y_t \right]\!\!\right] \tag{2.2}$$

where $\mathcal{A}$ is the learning algorithm, $\mathcal{H}$ is the hypothesis class (class of experts), h_t is the prediction of the learner, such that $h_t(x_t) = \hat{y}_t$.

Algorithm 2 Online Halving Algorithm.

Initialization Weights on experts $w_0^i = 1 \quad \forall i \in [N]$
for $t = 1, \ldots, T$ **do**
  Receive: Instance $x_t \in \mathcal{X}$
  Receive: Expert predictions $\xi^1(x_t), \cdots, \xi^N(x_t) \in \{0, 1\}$
  Predict : $\hat{y}_t = sign(\sum_{j=1}^{N} w_t^j \xi^j(x_t))$ (majority vote)
  Receive: True label $y_t \in \{0, 1\}$
  Incur : Loss $l(y_t, \hat{y}_t)$
  Update: $\forall i \in 1, \cdots, N$
    if $\xi^i(x_t) \neq y_t$ **then**
      $w_{t+1}^i \leftarrow 0$
    else
      $w_{t+1}^i \leftarrow w_t^i$
end

Definition 2.1. The algorithm $\mathcal{A}$ for a hypothesis class $\mathcal{H}$ has a mistake bound B iff $M(A, \mathcal{H}) \leq B$.

The mistake bound in the case of the Halving algorithm is given by $M(\text{Halving}, \mathcal{H}) \leq \log |\mathcal{H}|$. We prove this mistake bound later in this chapter under mistake bound analysis. In the case of the weighted majority algorithm, the regret bound for a non 0/1 loss and a choice of $\eta = \sqrt{\frac{8 \ln N}{T}}$ is given by

$$R_T \leq \sqrt{\frac{T}{2} \ln N}.$$

In Chapter 6 we revisit the weighted majority algorithm in order to enable prediction on multiple conflicting objectives. In this case, the hypotheses set is a collection of experts making multiple predictions. It is important to note that the mistake bound model is a more general model of learning theory than the regret in the online learning with experts. The mistake bound model aims to bound the mistakes of the learner against an adversary or nature who can control how the learner performs. This kind of model is more applicable for label prediction problems, which we use in Chapters 3, 4, and 5.

The bandit feedback algorithms are mainly used for sequential action selection problems, for example placing an advert from a set of advertisements onto a website. In the limited feedback structure of bandits, the feedback is only available for the action played for that particular round. This is the generalized version of the online prediction with experts and a much harder problem. In the bandit framework, a probability distribution is maintained on the actions, also called 'arms', at every trial. The learning algorithm chooses an action to perform, or an arm to pull, from the probability distribution, receives the feedback on that arm and updates the distribution. The bandit framework

is also called the Multi Arm Bandit(MAB) (Berry and Fristedt, 1985; Auer et al., 2002a, 1995; Allenberg et al., 2006; Gittins et al., 2011). These algorithms are known for their natural ability to address the exploitation-exploration trade-off (Auer et al., 1995, 2002a; Auer, 2003); balance between playing the same arm that receives good reward or play some other arm with unknown (potentially more) reward. Broadly speaking,

Algorithm 3 Weighted Majority Algorithm.

Initialization Weights on experts $w_0^i = 1 \quad \forall i \in [N]$
for $t = 1, \ldots, T$ **do**
  Receive: Instance $x_t \in \mathcal{X}$
  Receive: Expert predictions $\xi^1(x_t), \cdots, \xi^N(x_t) \in \{0, 1\}$
  Predict : $\hat{y}_t = sign(\sum_{j=1}^{N} w_t^j \xi^j(x_t))$ (majority vote)
  Receive: True label $y_t \in \{0, 1\}$
  Incur : Loss $l(y_t, \hat{y}_t)$
  Update: $\forall i \in 1, \cdots, N$
    if $\xi^i(x_t) \neq y_t$
      $w_{t+1}^i \leftarrow w_t^i \mathrm{e}^{-\eta \left[\!\left[y_t \neq \xi^i(x_t) \right]\!\right]}$
end

the field of research separates into stochastic bandits and adversarial bandits. Under the purview of stochastic bandits, the assumption is each arm belongs to an unknown probability distribution and the rewards are drawn from their respective distribution. Without any stochastic assumption on the way the reward corresponding to each action is generated, the problem becomes adversarial where rewards are generated based on the actions (Bubeck and Cesa-Bianchi, 2012). In Chapter 5, we study the adversarial structure of the bandits. The regret in this set-up is given by

$$R_T = \sum_{t=1}^{T} l_{I_t} - \min_{i=1,\cdots,K} \sum_{t=1}^{T} l_{i,t} \tag{2.3}$$

where I_t is the arm played by the algorithm at trial t, l is the loss suffered and K denotes the arms. The famous algorithm in the adversarial bandit structure is the Exp3 (Auer et al., 1995; Bubeck and Cesa-Bianchi, 2012) which we list in Algorithm 4.

It is important to note that in this thesis, the work is presented within the online Learning framework as a progression from the more general online learning settings (that of mistake bound model in Chapters 3, 4) to regret bound model (in Chapters 5, 6). Also, within the regret bound model we move from the more general bandit feedback (Chapter 5) to the expert feedback (Chapter 6).

Algorithm 4 Exp3 Online Bandit Algorithm

Parameters Non-increasing real number sequence η_t, $t \in \mathbb{N}$
Initialization Initial probability distribution p_1 over the arms $\{1, \cdots, K\}$
for $t = 1, \ldots, T$ **do**
Play: Arm I_t from p_t
Compute: Estimated loss for each arm $i \in \{1, \cdots, K\}$ as $\tilde{l}_{i,t} = \frac{l_{i,t}}{p_{i,t}} [\![I_t = i]\!]$
Cumulative loss $\tilde{L}_{i,t} = \tilde{L}_{i,t-1} + \tilde{l}_{i,t}$
Update : $p_{t+1} = (p_{1,t+1}, \cdots, p_{K,t+1})$, where $p_{i,t+1} = \frac{e^{-\eta_t \tilde{L}_{i,t}}}{\sum_{k=1}^{K} e^{-\eta_t \tilde{L}_{k,t}}}$
end

2.5 Online Labelling over Graphs

When the online learning game is played on a graph, the learning proceeds is as follows: `Nature` presents a graph $\mathcal{G}$; `Nature` queries a vertex $i_1 \in V(\mathcal{G}) = \mathbb{N}_n$; the `learner` predicts the label of the vertex $\hat{y}_1 \in \{0, 1\}$; `nature` presents a label y_1; `nature` queries a vertex i_2; the `learner` predicts $\hat{y}_2$; and so forth. The learner's goal is to minimize the total number of mistakes $M = |\{t : \hat{y}_t \neq y_t\}|$. If nature is adversarial, the learner will always make a "mistake", but if nature is regular, there is hope for the learning to succeed. Minimizing mistakes naturally fall within the purview of mistake bound model. This type of online learning model is one of the main themes running through this thesis mainly in Chapters 3, 4, and 5.

In online learning, the adversary plays the role on the nature or environment within which the learner functions. The primary goal of online learning is to develop algorithms such that total number of mis-classifications are bounded relative to the randomized complexity of the adversary (Herbster et al., 2009). Typically, the number of mistakes made by the optimal algorithms in this framework is bounded by $O(\ln n)$, where n is the number of vertices.

Let $\mathcal{G} = (V, E)$ denote the n-vertex graph such that $V = \{v_1, ..., v_n\}$ is the vertex set and E is the (possibly weighted) edge set denoted by $E = \{(i, j) \,|\, i \sim j\}$, where i, j are the unordered vertex indices for the edge. Associated with every edge is a function denoted by $w : E \to \mathbb{R}^+$. Here, we consider the unweighted graph equivalent to having unit weight on every edge. The adjacency matrix for the graph is denoted by $\boldsymbol{A}_{\mathcal{G}}$ and given by:

$$A_{\mathcal{G}}(i, j) = \begin{cases} w(i, j), & \text{if } (i, j) \in E \\ 0, & \text{otherwise.} \end{cases} \tag{2.4}$$

Typically, $\boldsymbol{A}_{\mathcal{G}}$ is the (weighted) symmetric adjacency matrix. In our work, $\mathcal{G}$ is unweighted where $\boldsymbol{A}_{\mathcal{G}} \in \{0, 1\}^{n \times n}$. The degree matrix of the graph is given by:

$$D_{\mathcal{G}}(i, i) = \sum_{j} A_{\mathcal{G}}(i, j). \tag{2.5}$$

From the degree matrix and the adjacency matrix, the un-normalized graph Laplacian can be derived and is defined by:

$$\boldsymbol{L}_{\mathcal{G}} = \boldsymbol{D}_{\mathcal{G}} - \boldsymbol{A}_{\mathcal{G}}. \tag{2.6}$$

In our work, we assume that the graph is undirected with one edge in every direction (i, j) and (j, i). For $\boldsymbol{u} \in \mathbb{R}^n$, the Laplacian satisfies the following property (Von Luxburg, 2007):

$$\boldsymbol{u}' L \boldsymbol{u} = \frac{1}{2} \sum_{i,j=1}^{n} w_{i,j} \left(u_i - u_j\right)^2. \tag{2.7}$$

The graph labelling vector $\boldsymbol{u}$ in the relaxed problem, is a vector of a n-vertex graph given by the function $\boldsymbol{u} : V \to \mathbb{R}$ defined on the vertices of the graph, where u_i corresponds to the label of v_i. If for a graph $\mathcal{G} = (V, E = \{(i_1, j_1), \ldots, (i_m, j_m)\})$, the weighted oriented incidence matrix or the edge map is given by a linear map, $\boldsymbol{\phi}_{\mathcal{G}} = \mathbb{R}^n \to \mathbb{R}^m$ then:

$$\boldsymbol{\phi}_{\mathcal{G}} \boldsymbol{u} = \left(A_{i_i,j_1}^{\frac{1}{p}} \left(u_{i_1} - u_{j_1}\right), \ldots, A_{i_m,j_m}^{\frac{1}{p}} \left(u_{i_m} - u_{j_m}\right) \right)^T. \tag{2.8}$$

With $p = 2$, the $n \times n$ matrix given by ${\boldsymbol{\phi}_{\mathcal{G}}}^T \boldsymbol{\phi}_{\mathcal{G}}$ is the graph Laplacian. Essentially, a p-seminorm (Herbster and Lever, 2009) is defined on the space of graph labellings given by:

$$||\boldsymbol{u}||_{\mathcal{G},p} = ||\boldsymbol{u}||_{\boldsymbol{\phi}_{\mathcal{G},p}} = \left(\sum_{(i,j) \in E_{\mathcal{G}}} A_{ij} \left|u_i - u_j\right|^p \right)^{\frac{1}{p}}. \tag{2.9}$$

The Laplacian based methods Belkin et al. (2004); Zhu et al. (2003) suffer from a limitation in the case of graphs with large diameter, the number of mistakes made by them is proportional to the square root of the number of vertices (Herbster et al., 2009) In (Herbster, 2008; Herbster and Pontil, 2006), the authors also exploit the cluster structure using the kernel perceptron when the perceptron is the inverse of the Laplacian of the graph but the mistake bounds similarly vary proportional to the graph diameter; especially in the worst case when the diameter is disproportionately larger than the number of vertices. In the best case, the mistake bounds vary proportional to the resistance diameter of the predicted vertices (Herbster et al., 2009). The semi-norm is defined a measure of smoothness of the labelling vector $\boldsymbol{u}$. It generalizes the smoothness functional $\boldsymbol{u}^T \boldsymbol{L} \boldsymbol{u}$ that measures the complexity of the graph labelling (Herbster and Lever, 2009). Typically, an edge in this graph indicates the expectation that its end point vertices or data points are more likely to have the same label. One method to take advantage of this representation is to use the semi-norm induced by the Laplacian on the graph. When the labelling are unrelaxed to be in the set such that $\boldsymbol{u} \in \{0, 1\}^n$, the edge becomes a "cut" if $u_i \neq u_j$, then the weighted cutsize of the labelling u is defined

as (Herbster and Lever, 2009):

$$\Phi_{\mathcal{G}}(\boldsymbol{u}) = \frac{1}{2^p} ||\boldsymbol{u}||^p_{\mathcal{G},p} = \frac{1}{2^p} \sum_{(i,j)\in E} A_{ij} |u_i - u_j|^p \tag{2.10}$$

where the smoothness of a boolean labelling of the graph is measured via the *"cut"*. The cut-size is the measure of the complexity which is independent of p and with the un-weighted graph, the cut-size is the total number of cut edges.

When $p \to 1$, the situation is similar to inducing a discrete distribution over the vertices. We discuss the connection to the Ising model distribution in the next section.

In this thesis, we study the online graph labelling problem when $p \to 1$ is Chapter 3 and Chapter 4. Chapter 5 addresses the graph labelling at $p \to 1$ as a contextual game within which the bandit game is played.

2.6 Ising Model

Here, we briefly review the background on Ising model (Ising, 1925) in the context of graph labelling. We assume that the underlying average behaviour of the system under observation is known. Suppose there are a multitude of graphs and that for the whole set, there is some average number of cuts c^* observed. A "cut" as we have seen before, is the number of edges with disagreeing labels. Let the cut of any graph in the set of graphs be defined by,

$$\phi_{\mathcal{G}}(\boldsymbol{u}) = \sum_{(i,j)\in E} \left[\!\left[u_i \neq u_j \right]\!\right] \tag{2.11}$$

where, $\boldsymbol{u}$ is the labelling of the graph, E is the number of edges. If the "cut" observed is given by c^*, on average the expected number of cuts should be equal to the cut observed:

$$\mathbb{E}\left[\phi_{\mathcal{G}}(\boldsymbol{u})\right] = c^*. \tag{2.12}$$

Now, if the probability of a particular labelling of the graph is given by $P(\boldsymbol{u})$, then the goal is to compute the probability of the labelling with the assumption that the average number of cuts c^* is observed. Alternatively, we have

$$\sum_{\boldsymbol{u}} P(\boldsymbol{u}) \phi_{\mathcal{G}}(\boldsymbol{u}) = c^*. \tag{2.13}$$

Typically, for inferring the probability distribution over all possible labellings $P(\boldsymbol{u})$, one would look at the entropy (Shannon, 2001; Shannon and Weaver, 2015; Shannon, 1949) of the distribution given by

$$H(\boldsymbol{u}) = -\sum_{\boldsymbol{u}} P(\boldsymbol{u}) \log\left(P(\boldsymbol{u})\right). \tag{2.14}$$

Maximum entropy of the distribution is the measure of the maximum uncertainty in the distribution. Naturally, the maximum uncertainty in the distribution is when all possible labellings are equally likely. In other words, maximum entropy counts the number of all possible ways of having cuts. Taking the Lagrangian of 2.14 and incorporating the constraints in 2.13 and $\sum_{\boldsymbol{u}} P(\boldsymbol{u}) = 1$, one is interested in maximizing the Lagrangian, as shown:

$$L = -\sum_{\boldsymbol{u}} P(\boldsymbol{u}) \log\left(P(\boldsymbol{u})\right) + \beta\left(\sum_{\boldsymbol{u}} P(\boldsymbol{u})\,\phi_{\mathcal{G}}(\boldsymbol{u}) - c^*\right) + \lambda\left(\sum_{\boldsymbol{u}} p(\boldsymbol{u}) - 1\right). \quad (2.15)$$

Maximizing with respect to the probability of each labelling,

$$\frac{\partial L}{\partial P(\boldsymbol{u})} = -\log P(\boldsymbol{u}) - 1 + \beta\phi_{\mathcal{G}}(\mathbf{S}) + \lambda = 0.$$

Rearranging and taking exponential, we have,

$$P(\boldsymbol{u}) = \mathrm{e}^{\beta\phi(\mathbf{S})+\lambda-1}. \quad (2.16)$$

For the parameters, we derive λ, from the second constraint,

$$\begin{aligned}
\sum_{\boldsymbol{u}} P(\boldsymbol{u}) = \sum_{\boldsymbol{u}} \mathrm{e}^{\beta\phi_{\mathcal{G}}(\boldsymbol{u})+\lambda-1} = 1 \\
\Rightarrow \mathrm{e}^{\lambda-1}\sum_{\boldsymbol{u}} \mathrm{e}^{\beta\phi_{\mathcal{G}}(\boldsymbol{u})} = 1 \\
\Rightarrow \mathrm{e}^{\lambda-1} = \frac{1}{\sum_{\boldsymbol{u}} \mathrm{e}^{\beta\phi_{\mathcal{G}}(\boldsymbol{u})}} \\
\Rightarrow P(\boldsymbol{u}) = \frac{\mathrm{e}^{\beta\phi_{\mathcal{G}}(\boldsymbol{u})}}{\sum_{\boldsymbol{u}} \mathrm{e}^{\beta\phi_{\mathcal{G}}(\boldsymbol{u})}}. \quad (2.17)
\end{aligned}$$

Let the partition function is defined as,

$$Z = \sum_{\boldsymbol{u}} \mathrm{e}^{\beta\phi_{\mathcal{G}}(\boldsymbol{u})}. \quad (2.18)$$

Therefore,

$$P(\boldsymbol{u}) = \frac{\mathrm{e}^{\beta\phi_{\mathcal{G}}(\boldsymbol{u})}}{Z}. \quad (2.19)$$

The probability distribution as derived in 2.19, is the model of maximum entropy distribution also known as the Ising model (Ising, 1925) or the Boltzmann distribution. Exact inference on the Ising model is an intractable problem (Murphy, 2012). β is the Lagrange multiplier that behaves as a crucial parameter to control the uncertainty in the model making a single labelling on a single graph preferable over others based on the cost variable $\phi(\boldsymbol{u})$; which in this case is the cut. Alternatively, $\beta = \frac{1}{\tau}$, where τ plays the role of that of temperature in a physical system. At the exact zero temperature $\tau = 0$, the probability distribution favours low cost solutions with minimum number of

cuts with overwhelming probability. At lowest temperatures or the highest uncertainty setting, it becomes very difficult to make a jump from low to high cost solutions, quite possibly leading to locally optimum low cost solutions. Very low temperatures push down on the landscape towards equally probable minimum cut solutions, while high temperatures could allow more movement in different directions (including the wrong directions) and equilibrate over equally likely high cost solutions. Interestingly, at lower temperatures but not the lowest, it is unlikely to converge to either very low solutions or very high cost solutions, allowing just enough flexibility to move around to avoid getting stuck in local optima. Typically, increasing the temperature very slowly (annealing), is likely to let it converge to low cost global solutions.

Chapter 4 in this thesis studies the Ising model in the context of online graph labelling within a semi supervised learning structure.

2.7 Exact, Approximation and Optimization Techniques

In Section 2.5 above, we discuss the online graph labelling problem. When $p \to 1$, it is equivalent to finding the number of cuts or edges with disagreeing labels $\sum_{ij \in E} |u_i - u_j|$. Further, in Section 2.6, the cost or energy function for the labelling is given by $\phi_{\mathcal{G}}(\boldsymbol{u}) = \sum_{(i,j) \in E} [\![u_i \neq u_j]\!]$. The Figure 2.1 below describes two example energy configurations. The computation of this cost in the partition function Z is a hard problem. The summation over all possible configurations for the cost function is intractable for large sized data or graphs. The posterior distribution P of the labelling is computed to determine the best fit. Computing the posterior involves the normalization over all possible labellings over all possible states of the vertices. A network of 10 vertices where each vertex is allowed to take a binary label, has 1024 possible labellings. This results in the computation being intractable. The natural direction is using approximation techniques where the posterior distribution is approximated with the help of a simpler distribution.

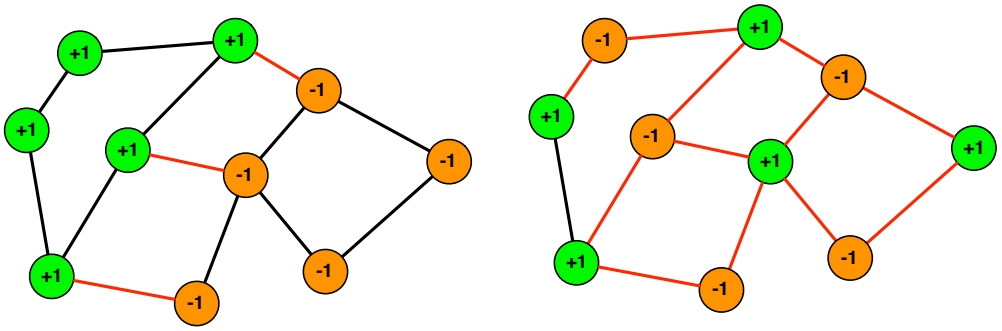


Figure 2.1: Minimum energy configuration is preferable. Left network has a minimum cut $\phi_{\mathcal{G}}(\boldsymbol{u}) = 3$. Right network has minimum cut $\phi_{\mathcal{G}}(\boldsymbol{u}) = 12$. Image courtesy Mark Herbster.

2.7.1 Exact Technique

A well established technique to calculate the minimum cut or maximum flow in the network is the Ford Fulkerson method (Ford and Fulkerson, 1956). At $p \to 1$, the objective is to find the minimum number of edges with disagreeing labels or the minimum cut. From linear programming duality, minimum cut is equal to maximum flow by Menger's theorem (Dantzig and Fulkerson, 2003). In Chapter 3, we therefore resort to an exact linear time computation of the maximum flow in the network. However, as we will see later, at the limit of zero temperature, there is a multiplicity of minimum cuts possible and inferring the labelling with the majority is NP hard. We therefore resort to a clever approximation that reduces the multiplicity and allows us to make progress towards prediction designing a quadratic time algorithm.

2.7.2 Mean Field Variational Approximation

A very well known mechanism of variational technique approximates the effect of the neighbourhood to infer the labelling of a vertex (Wainwright and Jordan, 2008). This is the direction we undertake in Chapter 4, moving on to a probabilistic setting for inference on the Ising model at low temperatures.

For an intuitive explanation of the mean field approximation from the perspective of a dynamic system, consider the neighbourhood system in Figure 2.2, where each spin (colour) is connected to every other spin with the spins allowed to fluctuate. The probability of an overall spin configuration or labelling $\boldsymbol{S}$ of the neighbourhood system changing to another configuration $\boldsymbol{S}'$ depends on the change in costs associated with individual spins getting flipped in the neighbourhood.

$$w\left(\boldsymbol{S} \to \boldsymbol{S}'\right) = \frac{\mathrm{e}^{\beta(\phi(\mathbf{S})-\phi(\mathbf{S}'))}}{\sum_{\boldsymbol{S}} \mathrm{e}^{\beta(\phi(\mathbf{S})-\phi(\mathbf{S}'))}} \tag{2.20}$$

The state of a particular spin depends on the state of its neighbourhood that is allowed to fluctuate. The state of the spin is influenced by the mean of the fluctuations of its neighbourhood. On the lattice as shown in Figure 2.2, each spin feels a field from each of its neighbours, which can be replaced by the mean field of its neighbours. Its spin can be approximated from the mean of the neighbourhood. With a large enough neighbourhood, the approximation tends to be closer to the true spin configuration.

2.7.3 Linear Programming Optimization

In Chapter 5, we use a relaxation technique to the linear programming (LP)optimization problem for computing the maximum flow in the network (Trevisan, 2011). A polynomial time LP relaxation computes the minimum cut labelling that is consistent with the labels

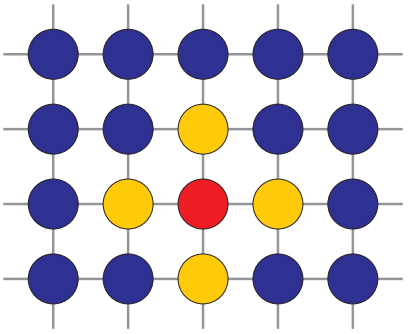


Figure 2.2: Ising 2d Lattice Spin Configuration.

seen over the sequence. A polynomial time relaxation introduces auxillary variables for the vertices and the edges. In principle, such relaxations to the original optimization problem are much easier to analyze. It is important to note that whereas in Chapter 3, on the Ising model, we study the behaviour at the limit of zero temperature, in Chapters 4 and 5, we study the low temperature setting of the Ising model.

2.8 Mistake Bounds in Online Learning Framework

In the previous sections, we highlighted the background behind the methods that we choose for addressing the problems that we are interested in. Naturally, the next direction after characterizing the set of all label consistent minimum cuts in the graph labelling problem, is to apply a heuristic for making the prediction. This section provides insight into the standard approaches taken in this direction, which motivates the premise of the algorithm presented in Chapter 3. For the analysis here, we assume $0/1$ loss.

Theorem 2.2. *For any finite hypothesis class $\mathcal{H}$ of size N, where one hypothesis is always right, the mistake bound of the Halving algorithm is given by*

$$M \leq \log_2(N).$$

Proof. Say we have N experts, each expert capable of making a prediction of a 1 or a 0 label. The assumption is that there is a single expert that makes no mistake. We maintain a set ξ_t of experts at round t of the online learning. Let ξ_t^0, ξ_t^1 be the subsets of ξ_t that predict a 1 or a 0. At round t, the learner has to predict a label as a 1 or a 0. If $\left|\xi_t^0\right| > \left|\xi_t^1\right|$, the learner predicts 0 else predicts 1. The environment reveals the true label $y^{'}$. After receiving the true label, the learner updates each subset $\xi_{t+1} = \xi_t y^{'}$. This implies that, if the learner made a mistake, $|\xi_{t+1}| \leq \frac{1}{2} |\xi_t|$. We know by our assumption,

the number of experts remaining $\xi_{t+1} \geq 1$. Therefore, if total number of mistakes in round t is M, then, $|\xi_{M+1}| \leq \frac{1}{2}|\xi_M| \implies 1 \leq |\xi_M| \leq \frac{1}{2^M}|\xi_0| \implies 2^M \leq |\xi_0| = N \implies M \leq \log_2(N)$. □

Next, we establish the connection between the Weighted-Halving experts and the Halving algorithm.

Theorem 2.3. *For all trials t and weighed experts i, the number of mistakes the Weighted-Halving makes is at most $M \leq \log_2(\frac{W_0}{m_i})$, where w_i is the weight of expert i, W_0 is the total weight of all experts.*

Proof. Let there be N experts $\xi_1, \xi_2, \ldots, \xi_n$ capable of making a prediction of a 0 or a 1. Each expert ξ_i has a weight w_i and makes mistake m_i. Let E_t^0, E_t^1 be the subset of experts of E_t that predict a 0 or a 1 at time t. Let the sum of weights of all the experts that predicted a 0 be given by $w_t^0 = \sum_{i:\xi_i \in E_t^0} w_i$ and $w_t^1 = \sum_{i:\xi_i \in E_t^1} w_i$. Total weight of all experts at time t is $W_t = w_t^0 + w_t^1$. The learner predicts $0 \iff w_t^0 > w_t^1$. The environment reveals the true label y. The learner updates the set of experts at time $t+1$ as $E_{t+1} = E_t y$. If the learner made a mistake, then the total sum of weights will be halved at every t such that $W_{t+1} \leq \frac{1}{2}W_t$. Assuming there is an expert i that is always right, after M mistakes, $W_M \geq w_i$. Since ξ_i is always in the set, sum of all experts is at least the weight of the right expert. So, $W_{M+1} \leq \frac{1}{2}W_M \implies W_M \leq \frac{1}{2^M}W_0 \implies w_i \leq W_M \leq \frac{1}{2^M}W_0 \implies M = \log_2(\frac{W_0}{w_i})$. □

At the beginning of the trials, since every expert has a weight 1, $W_0 = N$. Note, the Weighted Halving experts and the Halving experts are related as in an update after making a mistake, expert $i^{'}$ suffers a weight update $w_{i'} = aw_i$, where w_i is the best expert and in the case of halving experts it suffers $w_i^{'} = 0$. Halving experts have $a = 0$, where a is the constant term.

2.8.1 Ising Model and Weighted Experts

Here, we show the Ising Model is connected to the weighted experts. This is the first of our connections drawn towards motivating a logarithmic mistake bound in online learning approaches in graphs. The Ising model can be viewed a a weighted graph where every edge in the graph contributes a factor. The joint probability distribution factorizes over the edges. In the simplest case, the Ising model with equal weight in every edge gives a probability distribution over the labellings of the vertices of the graph, where each vertex can have either of the two possible labels. In the above section, the initial weights sum W_0 is a normalization term and can be ignored as it only adds a constant factor.

Theorem 2.4. *For an expert i with weight w_i, if the probability of it being correct is $p(i)$, then the cut-size of an Ising model is given by $\phi(x)$, then $p(i) = 2^{-\phi(x)}$ which is the Bayesian prediction on an Ising model*

Proof. Let us now consider we have a probability distribution of weights of the experts such that $w_i = p(i)$, where $p(i)$ is the probability that expert i is correct. So, w_t^0 is the probability of getting a 0 and w_t^1 is the probability of getting a 1. The learner predicts by the size of the maximum probability. (Note: This is essentially Bayesian inference. It can be shown that Bayes optimal classifier gives the weighted experts). Ignoring the normalization term, the total number of mistakes by i $M_i = \log(\frac{1}{p(i)})$. In the case of the Ising model, the complexity is given by the cutsize which is the total number of edges where the labels of the vertices disagree. Now, if we correspond weights to cutsize, then it is possible to obtain logarithmic mistake bounds upto a constant factor times cutsize. Therefore, if the cutsize of the Ising model is given by $\phi(x)$, we have $\log(\frac{1}{p(i)}) \propto c\phi(x)$, where c is the constant factor which $\implies$ $p(i) = 2^{-\phi(x)}$. This is nothing but Bayesian prediction on the Ising Model. □

2.8.2 Halving implementing Nearest Neighbour on a Path Graph

It is shown with a probability distribution over the space of labellings $\boldsymbol{u} \in \{-1, 1\}^n$ on any path graph, the Halving algorithm with these probabilities implements the nearest neighbour algorithm and achieves logarithmic mistake bounds and in fact is a Bayesian Optimal classifier. Please refer to (Herbster et al., 2009) Section 5, for the proofs.

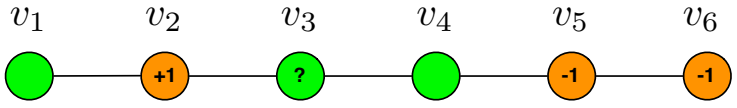


Figure 2.3: 1-NN Prediction on Path Graph.

In Figure 2.3, the nearest neighbour strategy on a path graph is shown. Vertices v_2, v_5 and v_6 are the available labels, and nature queries vertex v_3. The 1-NN algorithm predicts the label of v_3 as the label $+1$ of its nearest neighbour v_2. The rationale behind the strategy is in case the prediction is a mistake, that is the label of v_3 should be -1, from the minimum energy principle, the minimum cut is 1, hence the label of v_4 is automatically deduced making the total number of mistakes to be 1. However, if the algorithm predicted by the next nearest neighbour's label of -1 of that of vertex v_5, in case of a mistake, no information is deduced for vertex v_4 as v_4 can be still be a -1 in the next round and maintain the minimum cut. Hence, total number of possible mistakes is 2 which is more than the 1-NN prediction.

Further, in the paper by Gärtner and Garriga (2007), the authors describe how the Halving algorithm on any fixed path on a generic graph achieves a logarithmic mistake bound on every such fixed path. The total mistakes is the sum of mistakes on each path. If $p^1, \ldots, p^k$ is the set of vertex disjoint directed paths that cover the graph completely, then the mistake bounds are optimal and given by $M \leq \sum_{i=1}^{k} \log \left|p^i\right|$ where, $\left|p^i\right|$ is the length (total number of vertices or edges) of the individual path and there are k such paths. Every neighbouring vertex's prediction on that path is considered an expert prediction. Hence, the analogy with the Halving experts. Figure 2.4, shows the disjoint path cover of one such graph. The path on which the unlabelled vertex in question is present, if selected for prediction, can make at most its length (number of vertices on that path) number of mistakes. The two labelled vertices are indicated in the Figure 2.4. Note, that in the case of the zero temperature limit on the Ising model, there

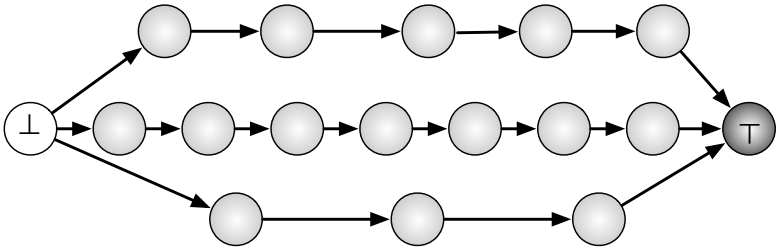


Figure 2.4: Disjoint Path Cover. Image courtesy Mark Herbser.

are multiple label consistent minimum cut labellings. As we saw, ideally the mistake bound $M \leq \log |\mathcal{H}|$, where $\mathcal{H}$ is the set of all experts. However, the total number of minimum cuts is non-unique and given a path cover for the multiple minimum cut labelling is upper bounded as $|\mathcal{H}| \leq \text{argmin}_{p \in P(G)} \prod_{i=1}^{k} \left|p^i\right|$, having the desired mistake bound is non-trivial.

This seeds the intention of our interests in the mistake bound model. In particular, in Chapter 3, we show a successful technique of characterizing the set of all label consistent minimum cuts. Once characterised, the next step is typically designing a prediction heuristic for choosing a path(s) and predicting using nearest neighbour such that the mistake bound is logarithmic in the size of the number of the path(s).

Notation Throughout the thesis, with slight abuse of notation, we refer to a learning algorithm as the learner. Predictor and forecaster are also used interchangeably to mean the learner. In Chapter 3, we interchangeably use longest-path and longest to denote the same algorithm. Further, for the competitor algorithm $p = 2$, it is referred by the following: `p2`, `labelProp` and harmonic energy minimization.

Chapter 3

Online Prediction at the Limit of Zero Temperature

3.1 Introduction

This chapter focusses on the problem of sequential classification of the unlabelled vertices of a graph with few available labels; also known as semi-supervised learning. Semi-supervised graph labelling solutions, induce regularization by exploiting the underlying structure of the graph. Often, these methods associate a complexity to the labelling and aim to optimize the "cut", which is the number of edges with disagreeing labels. This is equivalent to minimizing the seminorm on the "cut" as a measure of the smoothness of the solution. When the labelling is unrelaxed such that the labels can only take boolean values, it induces an Ising distribution over the vertices of the graph, where the labels of the vertices can be inferred based on the maximum marginal probability of the vertices. However, exact inference on the Ising model is an intractable problem.

Here, we define and discuss a technique for the sequential deterministic approximate inference to predict the labels of the queried unlabelled vertices on a graph. Our technique has proven to be useful in a variety of standard empirical studies. In addition, we also provide online mistake bounds that improve and match the previous bounds in the literature for this setting.

3.2 Background

Let $\mathcal{G} = (V, E)$ denote the n-vertex graph such that $V = \{v_1, ..., v_n\}$ is the vertex set and E is the (possibly weighted) edge set denoted by $E = \{(i, j) \,|i \sim j\}$, where i, j are the unordered vertex indices for the edge. Associated with every edge is a function denoted

by $w : E \to \mathbb{R}^+$. Here, we consider unweighted graph equivalent to having unit weight on every edge. The adjacency matrix for the graph is denoted by $\boldsymbol{A}_{\mathcal{G}}$ and given by:

$$A_{\mathcal{G}}(i,j) = \begin{cases} w(i,j), & \text{if } (i,j) \in E \\ 0, & \text{otherwise.} \end{cases} \tag{3.1}$$

Typically, $\boldsymbol{A}_{\mathcal{G}}$ is the (weighted) symmetric adjacency matrix. In our work, $\mathcal{G}$ is unweighted where $\boldsymbol{A}_{\mathcal{G}} \in \{0,1\}^{n \times n}$. The degree matrix of the graph is given by:

$$D_{\mathcal{G}}(i,i) = \sum_j A_{\mathcal{G}}(i,j). \tag{3.2}$$

From the degree matrix and the adjacency matrix, the un-normalized graph Laplacian can be derived and is defined by:

$$\boldsymbol{L}_{\mathcal{G}} = \boldsymbol{D}_{\mathcal{G}} - \boldsymbol{A}_{\mathcal{G}}. \tag{3.3}$$

In our work, we assume that the graph is undirected with one edge in every direction (i,j) and (j,i). For $\boldsymbol{u} \in \mathbb{R}^n$, the Laplacian satisfies the following property (Von Luxburg, 2007; Herbster et al., 2009):

$$\boldsymbol{u}' L \boldsymbol{u} = \frac{1}{2} \sum_{i,j=1}^{n} w_{i,j} (u_i - u_j)^2. \tag{3.4}$$

The factor of 2 in the above equation is to take into account the double counting of the edges in the graph. The graph labelling vector $\boldsymbol{u}$ in the relaxed problem, is a vector of the n-vertex graph given by the function $\boldsymbol{u} : V \to \mathbb{R}$ defined on the vertices of the graph, where u_i corresponds to the label of v_i. If $\mathcal{G} = (V, E = \{(i_1, j_1), \ldots, (i_m, j_m)\})$ is an edge map also known as weighted oriented incidence matrix $\boldsymbol{\phi}_{\mathcal{G}} = \mathbb{R}^n \to \mathbb{R}^m$ is a map given by:

$$\boldsymbol{\phi}_{\mathcal{G}} \boldsymbol{u} = \left(A_{i_1,j_1}^{\frac{1}{p}} (u_{i_1} - u_{j_1}), \ldots, A_{i_m,j_m}^{\frac{1}{p}} (u_{i_m} - u_{j_m}) \right)^T. \tag{3.5}$$

With $p = 2$, the $n \times n$ matrix given by ${\boldsymbol{\phi}_{\mathcal{G}}}^T \boldsymbol{\phi}_{\mathcal{G}}$ is the graph Laplacian. Essentially, a p-seminorm (Herbster and Lever, 2009) is defined on the space of graph labellings given by:

$$||\boldsymbol{u}||_{\mathcal{G},p} = ||\boldsymbol{u}||_{\boldsymbol{\phi}_{\mathcal{G},p}} = \left(\sum_{(i,j) \in E_{\mathcal{G}}} A_{ij} |u_i - u_j|^p \right)^{\frac{1}{p}}. \tag{3.6}$$

The seminorm is defined as a measure of smoothness of the labelling vector $\boldsymbol{u}$. It generalizes the the smoothness functional $\boldsymbol{u}^T \boldsymbol{L} \boldsymbol{u}$ that measures the complexity of the graph labelling (Herbster and Lever, 2009). Typically, an edge in this graph indicates

the expectation that its end point vertices or data points are more likely to have the same label. One method to take advantage of this representation is to use the seminorm induced by the Laplacian on the graph. When the labelling is unrelaxed it belongs to the set $\boldsymbol{u} \in \{0,1\}^n$. In this case, the edge becomes a "cut" if $u_i \neq u_j$ and the weighted cutsize of the labelling u is defined as (Herbster and Lever, 2009):

$$\Phi_{\mathcal{G}}(\boldsymbol{u}) = \frac{1}{2^p} ||\boldsymbol{u}||_{\mathcal{G},p}^p = \frac{1}{2^p} \sum_{(i,j)\in E} A_{ij} |u_i - u_j|^p . \tag{3.7}$$

The smoothness of the boolean labelling of the graph is then measured via the "cut". The cut-size is the measure of the complexity which is independent of p and with the un-weighted graph, the cut-size is the total number of cut edges.

3.3 Motivation

Our main motivation for this current work stems from our interest in the p-seminorm of the graph labelling when $p \to 1$, which to the best of our knowledge, has not been studied before in a similar setting. Optimal bounds are known only for approximations on a tree that does not capture the graph connectivity structure. Specifically, we are motivated to study if with $p \to 1$, on boolean labelling, we can find interesting graph structure that gives better prediction and mistake bounds.

From the linear programming knowledge, at $p \to 1$ in the limit, the situation is akin to obtaining the solution to the maximum flow in the network with unit capacities on the edges. As seen in Chapter 2, the model that naturally respects the complexity of the labelling (number of edges with disagreeing labels) is the Ising model. The Ising probability distribution over labellings of $\mathcal{G}$ is defined by:

$$p_T^{\mathcal{G}}(\boldsymbol{u}) \propto \exp\left(-\frac{1}{\tau}\Phi_{\mathcal{G}}(\boldsymbol{u})\right) \tag{3.8}$$

where τ is the temperature, $\Phi_{\mathcal{G}}(\boldsymbol{u})$ is the complexity of labelling or the "cut-size".

3.4 Related Work

Semi-supervised learning is now a standard methodology in machine learning. A common approach in semi-supervised learning is to build a graph (Blum and Chawla, 2001) from a given set of labelled and unlabelled data with each datum represented as a vertex. The hope is that the constructed graph will capture either the cluster (Chapelle et al., 2003) or manifold (Belkin and Niyogi, 2004) structure of the data. Typically, an edge in this graph indicates the expectation that the joined data points are more likely to

have the same label. One method to exploit this representation is to use the seminorm induced by the Laplacian of the graph (Zhu et al., 2003; Belkin and Niyogi, 2004; Zhou et al., 2003; Szummer and Jaakkola, 2001). A shared idea of the Laplacian seminorm based approaches is that the smoothness of a boolean labelling of the graph is measured via the "cut", which is just the number of edges that connect disagreeing labels. In practice the seminorm is then used as a regularizer in which the optimization problem is relaxed from boolean to real values. Our approach also uses the "cut", but unrelaxed, to define an Ising distribution over the vertices of the graph.

Predicting with the vertex marginals of an Ising distribution in the limit of zero temperature was shown to be optimal in the mistake bound model (Cesa-Bianchi et al., 2009, Section 4.1) when the graph is a tree. The exact computation of marginal probabilities in the Ising model is intractable on non-trees (Goldberg and Jerrum, 2007). However, in the limit of zero temperature, a rich combinatorial structure called the Picard-Queyranne graph (Picard and Queyranne, 1980) emerges. We exploit this structure to give an algorithm which 1) is optimal on trees, 2) has a quadratic computational complexity, and 3) has a mistake bound on generic graphs that is stronger than previous bounds in many natural cases.

In the remainder of this section, we introduce the Ising model and lightly review previous work in the online mistake bound model for predicting the labelling of a graph. In Section 3.6 we review our key technical tool the Picard-Queyranne graph (Picard and Queyranne, 1980) and explain the required notation. In the body of Section 3.7 we provide a mistake bound analysis of our algorithm as well as the intractable `0-Ising` algorithm and then conclude with a detailed comparison to the state of the art. In Section 3.8, we provide an extensive set of experimental results of the empirical evaluation of our method.

Predicting the labelling of a graph in the mistake bound model. We prove performance guarantees for our method in the mistake bound model introduced by Littlestone (Littlestone, 1988). On the graph this model corresponds to the following game. `Nature` presents a graph $\mathcal{G}$; `Nature` queries a vertex $i_1 \in V(\mathcal{G}) = \mathbb{N}_n$; the `learner` predicts the label of the vertex $\hat{y}_1 \in \{0, 1\}$; `nature` presents a label y_1; `nature` queries a vertex i_2; the `learner` predicts $\hat{y}_2$; and so forth. The learner's goal is to minimize the total number of mistakes $M = |\{t : \hat{y}_t \neq y_t\}|$. If nature is adversarial, the learner will always make a "mistake", but if nature is regular or simple, there is hope that a learner may incur only a few mistakes. Thus, a central goal of online learning is to design algorithms whose total mistakes can be bounded relative to the complexity of nature's labelling. The graph labelling problem has been studied extensively in the online literature. Here we provide a rough discussion of the two main approaches for graph label prediction, and in Section 3.7.3 we provide a more detailed comparison. The first approach is based on the graph Laplacian (Herbster et al., 2005; Herbster, 2008; Herbster and Lever, 2009); it provides bounds that utilize the additional connectivity

of non-tree graphs, which are particularly strong when the graph contains uniformly-labelled clusters of small (resistance) diameter. The drawbacks of this approach are that the bounds are weaker on graphs with large diameter and that the computation times are slower. The second approach is to estimate the original graph with an appropriately selected tree or "path" graph (Herbster et al., 2009; Cesa-Bianchi et al., 2010, 2009; Vitale et al., 2011); this leads to faster computation times, and bounds that are better on graphs with large diameters. The algorithm `treeOpt` (Cesa-Bianchi et al., 2009) is optimal on trees. These algorithms may be extended to non-tree graphs by first selecting a spanning tree uniformly at random (Cesa-Bianchi et al., 2010) and then applying the algorithm to the sampled tree. This randomized approach exploits the cluster structure in the graph, thereby reducing the expected mistake bound.

The bounds we prove for the NP-hard `0-Ising` prediction and our heuristic are most similar to the "small p" bounds proven for the p-seminorm interpolation algorithm (Herbster and Lever, 2009). Although these bounds are not strictly comparable, a key strength of our approach is that the new bounds often improve when the graph contains uniformly-labelled clusters of varying diameters. Furthermore, when the graph is a tree we match the optimal bounds of (Cesa-Bianchi et al., 2009). Finally, the cumulative time required to compute the complete labelling of a graph is quadratic in the size of the graph for our algorithm, while (Herbster and Lever, 2009) requires the minimization of a non-strongly convex function (on every trial) which is not differentiable when $p \to 1$.

3.5 Preliminaries

In the rest of the chapter, $\mathcal{G}$ is an (undirected) graph of a pair of sets (V, E), where E is a set of *unordered* pairs of distinct elements from V. We use $\mathcal{R}$ to denote the subgraph $\mathcal{R} \subseteq \mathcal{G}$ iff $V(\mathcal{R}) \subseteq V(\mathcal{G})$ and $E(\mathcal{R}) = \{(i,j) : i,j \in V(\mathcal{R}), (i,j) \in E(\mathcal{G})\}$. Given any subgraph $\mathcal{R} \subseteq G$, we define its *boundary* (or inner border) $\partial_0(\mathcal{R})$, its *neighbourhood* (or exterior border) $\partial_e(\mathcal{R})$ respectively as $\partial_0(\mathcal{R}) := \{j : i \notin V(\mathcal{R}), j \in V(\mathcal{R}), (i,j) \in E(\mathcal{G})\}$, and $\partial_e(\mathcal{R}) := \{i : i \notin V(\mathcal{R}), j \in V(\mathcal{R}), (i,j) \in E(\mathcal{G})\}$, and its exterior edge border $\partial_e^E(\mathcal{R}) := \{(i,j) : i \notin V(\mathcal{R}), j \in V(\mathcal{R}), (i,j) \in E(\mathcal{G})\}$. $|\mathcal{P}| := |E(\mathcal{P})|$ is the length of a subgraph $\mathcal{P}$ and we denote the diameter of a graph by $D(\mathcal{G})$. A pair of vertices $v, w \in V(\mathcal{G})$ are *κ-connected* if there exist κ edge-disjoint paths connecting them. The *connectivity of a graph*, $\kappa(\mathcal{G})$, is the maximal value of κ such that every pair of points in $\mathcal{G}$ is κ-connected. The *atomic number* $\mathcal{N}_\kappa(\mathcal{G})$ of a graph at connectivity level κ is the minimum cardinality c of a partition of $\mathcal{G}$ into subgraphs $\{\mathcal{R}_1, \ldots, \mathcal{R}_c\}$ such that $\kappa(\mathcal{R}_i) \geq \kappa$ for all $1 \leq i \leq c$.

Our results also require the use of *directed-, multi-,* and *quotient-* graphs. Every undirected graph also defines a directed graph where each undirected edge (i,j) is represented by directed edges (i,j) and (j,i). An *orientation* of an undirected graph is an assignment of a direction to each edge, turning the initial graph into a directed graph. In a

multi-graph the edge set is now a multi-set and thus there may be multiple edges between two vertices. A *quotient-graph* $\mathbb{G}$ is defined from a graph $\mathcal{G}$ and a partition of its vertex set $\{V_i\}_{i=1}^N$ so that $V(\mathbb{G}) := \{V_i\}_{i=1}^N$ (we often call these vertices *super-vertices* to emphasize that they are sets) and the multiset $E(\mathbb{G}) := \{(I,J) : I,J \in V(\mathbb{G}), I \neq J, i \in I, j \in J, (i,j) \in E(\mathcal{G})\}$. We commonly construct a quotient-graph $\mathbb{G}$ by "merging" a collection of super-vertices, for example, in Figure 3.2 from 3.2a to 3.2b where 6 and 9 are merged to "6/9" and also the five merges that transforms 3.2c to 3.2d.

The set of all *label-consistent minimum-cuts* in a graph with respect to an example sequence $\mathcal{S}$ is $\mathcal{U}^*_{\mathcal{G}}(\mathcal{S}) := \operatorname{argmin}_{\boldsymbol{u}\in\{0,1\}^n} \phi_{\mathcal{G}}(\boldsymbol{u}|\mathcal{S})$. The minimum is typically non-unique. For example in Figure 3.2a, the vertex sets $\{v_1,\ldots,v_4\},\{v_5,\ldots,v_{12}\}$ correspond to one label-consistent minimum-cut and $\{v_1,\ldots,v_5,v_7,v_8\},\{v_6,v_9\ldots,v_{12}\}$ to another (the cutsize is 3). The (uncapacitated) *maximum flow* is the number of edge-disjoint paths between a source and target vertex. Thus in Figure 3.2b between vertex "1" and vertex "6/9" there are at most 3 simultaneously edge-disjoint paths; these are also not unique, as one path must pass through either vertices $\langle v_{11}, v_{12}\rangle$ or vertices $\langle v_{11}, v_{10}, v_{12}\rangle$. Figure 3.2c illustrates one such flow $\mathcal{F}$ (just the directed edges). For convenience it is natural to view the maximum flow or the label-consistent minimum-cut as being with respect to only two vertices as in Figure 3.2a transformed to Figure 3.2b so that $\mathcal{H} \leftarrow \texttt{merge}(\mathcal{G}, \{v_6, v_9\})$. The "flow" and the "cut" are related by Menger's theorem which states that the minimum-cut with respect to a source and target vertex is equal to the max flow between them. Given a connected graph $\mathcal{H}$ and source and target vertices s,t the Ford-Fulkerson algorithm (Ford and Fulkerson, 1956) can find k edge-disjoint paths from s to t in time $O(k|E(\mathcal{H})|)$ where k is the value of the max flow.

3.6 Ising Model in the Limit Zero Temperature

In our setting, the parameters of the Ising model are an n-vertex graph $\mathcal{G} = (V(\mathcal{G}), E(\mathcal{G}))$ and a temperature parameter $\tau > 0$, where $V(\mathcal{G}) = \{1,\ldots,n\}$ denotes the vertex set and $E(\mathcal{G})$ denotes the edge set. Each vertex of this graph may be labelled with one of two states $\{0,1\}$ and thus a labelling of a graph may be denoted by a vector $\boldsymbol{u} \in \{0,1\}^n$ where u_i denotes the label of vertex i. The *cutsize* of a labelling $\boldsymbol{u}$ is defined as

$$\phi_{\mathcal{G}}(\boldsymbol{u}) := \sum_{(i,j)\in E(\mathcal{G})} |u_i - u_j|. \tag{3.9}$$

The Ising probability distribution over labellings of $\mathcal{G}$ is then defined as $p^{\mathcal{G}}_\tau(\boldsymbol{u}) \propto \exp\left(-\frac{1}{\tau}\phi_{\mathcal{G}}(\boldsymbol{u})\right)$ where $\tau > 0$ is the temperature parameter. In our online setting at the beginning of trial $t+1$ we will have already received an *example sequence*, $\mathcal{S}_t$, of t vertex-label pairs $(i_1,y_1),\ldots,(i_t,y_t)$ where pair $(i,y) \in V(\mathcal{G}) \times \{0,1\}$. We use $p^{\mathcal{G}}_\tau(u_v = y|\mathcal{S}_t) := p^{\mathcal{G}}_\tau(u_v = y|u_{i_1} = y_1,\ldots,u_{i_t} = y_t)$ to denote the marginal probability

that vertex v has label y given the previously labelled vertices of $\mathcal{S}_t$. For convenience we also define the marginalized *cutsize* $\phi_{\mathcal{G}}(\boldsymbol{u}|\mathcal{S}_t)$ to be equal to $\phi_{\mathcal{G}}(\boldsymbol{u})$ if $u_{i_1} = y_1, \ldots, u_{i_t} = y_t$ and equal to undefined otherwise. Our prediction $\hat{y}_{t+1}$ of vertex i_{t+1} is then the label with maximal marginal probability in the limit of zero temperature, thus

$$\hat{y}^{0\mathrm{I}}_{t+1}(i_{t+1}|\mathcal{S}_t) := \operatorname*{argmax}_{y\in\{0,1\}} \lim_{\tau\to 0} p^{\mathcal{G}}_{\tau}(u_{i_{t+1}} = y | u_{i_1} = y_1, \ldots, u_{i_t} = y_t)\,. \qquad \texttt{[0-Ising]} \tag{3.10}$$

Note the prediction is undefined if the labels are equally probable. In low temperatures the mass of the marginal is dominated by the labellings consistent with $\mathcal{S}_t$ and the proposed label of vertex i_{t+1} of minimal cut; as we approach zero, $\hat{y}_{t+1}$ is the label consistent with the maximum number of labellings of minimal cut. Thus if

$$k := \min_{\boldsymbol{u}\in\{0,1\}^n} \phi_{\mathcal{G}}(\boldsymbol{u}|\mathcal{S}) \tag{3.11}$$

then we have that:

$$\hat{y}^{0\mathrm{I}}(v|\mathcal{S}) = \begin{cases} 0 \,|\boldsymbol{u} \in \{0,1\}^n : \phi_{\mathcal{G}}(\boldsymbol{u}|(\mathcal{S},(v,0))) = k| > |\boldsymbol{u} \in \{0,1\}^n : \phi_{\mathcal{G}}(\boldsymbol{u}|(\mathcal{S},(v,1))) = k| \\ 1 \,|\boldsymbol{u} \in \{0,1\}^n : \phi_{\mathcal{G}}(\boldsymbol{u}|(\mathcal{S},(v,0))) = k| < |\boldsymbol{u} \in \{0,1\}^n : \phi_{\mathcal{G}}(\boldsymbol{u}|(\mathcal{S},(v,1))) = k| \end{cases}$$

The problem of counting minimum label-consistent cuts was shown to be #P-complete in (Provan and Ball, 1983) and further computing $\hat{y}^{0\mathrm{I}}(v|\mathcal{S})$ is also NP-hard. In Section 3.6.1 we introduce the Picard-Queyranne graph (Picard and Queyranne, 1980) which captures the combinatorial structure of the set of minimum-cuts. We then use this simplifying structure as a basis to design a heuristic approximation to $\hat{y}^{0\mathrm{I}}(v|\mathcal{S})$ with a mistake bound guarantee.

3.6.1 The Picard-Queyranne graph

Given a set of labels there may be multiple label-consistent minimum-cuts as well as multiple maximum flows in a graph. The Picard-Queyranne (PQ) graph (Picard and Queyranne, 1980) reduces this multiplicity as far as is possible with respect to the indeterminacy of the maximum flow. The vertices of the PQ-graph are defined as a super-vertex set on a partition of the original graph's vertex set. Two vertices are contained in the same super-vertex iff they have the same label in *every* label-consistent minimum-cut. An edge between two vertices defines an analogous edge between two super-vertices iff that edge is conserved in *every* maximum flow. Furthermore the edges between super-vertices strictly orient the labels in any label-consistent minimum-cut as may be seen in the formal definition that follows.

First we introduce the following useful notations: let $k_{\mathcal{G},\mathcal{S}} := \min\{\phi_{\mathcal{G}}(\boldsymbol{u}|\mathcal{S}) : \boldsymbol{u} \in \{0,1\}^n\}$ denote the minimum-cutsize of $\mathcal{G}$ with respect to $\mathcal{S}$; let $i \overset{\mathcal{S}}{\sim} j$ denote an equivalence relation between vertices in $V(\mathcal{G})$ where $i \overset{\mathcal{S}}{\sim} j$ iff $\forall \boldsymbol{u} \in \mathcal{U}^*_{\mathcal{G}}(\mathcal{S}) : u_i = u_j$; and then we define,

Definition 3.1 (Picard and Queyranne (1980))**.** The *Picard-Queyranne graph* $\mathbb{G}(\mathcal{G}, \mathcal{S})$ is derived from graph $\mathcal{G}$ and non-trivial example sequence $\mathcal{S}$. The graph is an orientation of the quotient graph derived from the partition $\{\bot, I_2, \ldots, I_{N-1}, \top\}$ of $V(\mathcal{G})$ induced by $\overset{\mathcal{S}}{\sim}$. The edge set of $\mathbb{G}$ is constructed of $k_{\mathcal{G},\mathcal{S}}$ edge-disjoint paths starting at source vertex $\bot$ and terminating at target vertex $\top$. A labelling $\boldsymbol{u} \in \{0,1\}^n$ is in $\mathcal{U}^*_{\mathcal{G}}(\mathcal{S})$ iff

1. $i \in \bot$ implies $u_i = 0$ and $i \in \top$ implies $u_i = 1$
2. $i, j \in H$ implies $u_i = u_j$
3. $i \in I$, $j \in J$, $(I, J) \in E(\mathbb{G})$, and $u_i = 1$ implies $u_j = 1$

where $\bot$ and $\top$ are the source and target vertices and $H, I, J \in V(\mathbb{G})$.

Figure 3.3 shows an illustration of the PQ graph construction process. As $\mathbb{G}(\mathcal{G}, \mathcal{S})$ is a DAG it naturally defines a partial order $(V(\mathbb{G}), \leq_{\mathbb{G}})$ on the vertex set where $I \leq_{\mathbb{G}} J$ if there exists a path starting at I and ending at J. The least and greatest elements of the partial order are $\bot$ and $\top$. The notation $\uparrow R$ and $\downarrow R$ denote the *up set* and *down set* of R. Given the set $\mathcal{U}^*$ of all label-consistent minimum-cuts then if $\boldsymbol{u} \in \mathcal{U}^*$ there exists an antichain $A \subseteq V(\mathbb{G}) \setminus \{\top\}$ such that $u_i = 0$ when $i \in I \in \downarrow A$ otherwise $u_i = 1$; furthermore for every antichain there exists a label-consistent minimum-cut. The simple structure of $\mathbb{G}(\mathcal{G}, \mathcal{S})$ was utilized by Picard and Queyranne (1980) to enable the efficient algorithmic enumeration of minimum-cuts. However, the cardinality of this set of all label-consistent minimum-cuts is potentially exponential in the size of the PQ-graph and the exact computation of the cardinality was later shown #P-complete in (Provan and Ball, 1983). In Figure 3.1 we give the algorithm from (Picard and Queyranne,

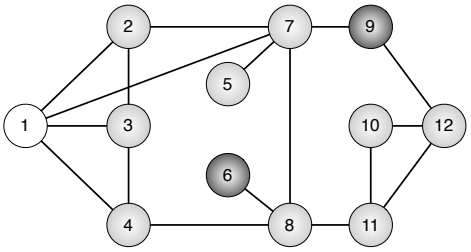
PicardQueyranneGraph(*graph:* $\mathcal{G}$; *example sequence:* $\mathcal{S} = (v_k, y_k)_{k=1}^t$)

1. $(\mathcal{H}, s, t) \leftarrow \text{SourceTargetMerge}(\mathcal{G}, \mathcal{S})$
2. $\mathcal{F} \leftarrow \text{MaxFlow}(\mathcal{H}, s, t)$
3. $\mathcal{I} \leftarrow (V(\mathcal{I}), E(\mathcal{I}))$ where $V(\mathcal{I}) := V(\mathcal{H})$ and $E(\mathcal{I}) := \{(i,j) : (i,j) \in E(\mathcal{H}), (j,i) \notin \mathcal{F}\}$
4. $\mathbb{G}^0 \leftarrow \text{QuotientGraph}(\text{StronglyConnectedComponents}(\mathcal{I}), \mathcal{H})$
5. $E(\mathbb{G}) \leftarrow E(\mathbb{G}^0)$; $V(\mathbb{G}) \leftarrow V(\mathbb{G}^0)$ except $\bot(\mathbb{G}) \leftarrow \bot(\mathbb{G}^0) \cup \{v_k : k \in \mathbb{N}_t, y_k = 0\}$ and $\top(\mathbb{G}) \leftarrow \top(\mathbb{G}^0) \cup \{v_k : k \in \mathbb{N}_t, y_k = 1\}$

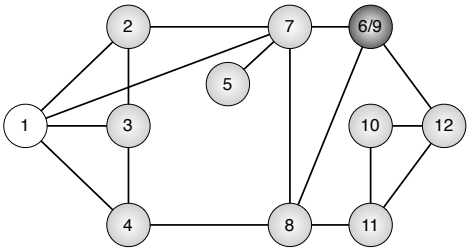
Return: *directed graph:* $\mathbb{G}$

Figure 3.1: Computing the Picard-Queyranne graph

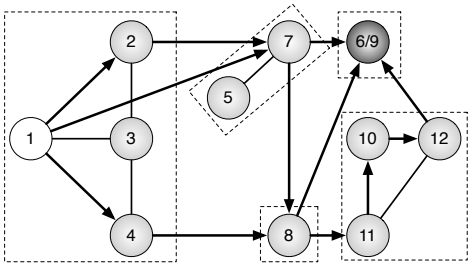
1980; Ball and Provan, 1983) to compute a PQ-graph. We illustrate the computation in Figure 3.2. The algorithm operates first on $(\mathcal{G}, \mathcal{S})$ (step 1) by " merging" all vertices which share the same label in $\mathcal{S}$ to create $\mathcal{H}$. In step 2 a max flow graph $\mathcal{F} \subseteq \mathcal{H}$ is computed by the Ford-fulkerson algorithm. It is well-known in the case of unweighted graphs that a max flow graph $\mathcal{F}$ may be output as a DAG of k edge-disjoint paths where k is the value of the flow. In step 3 all edges in the flow become directed edges creating $\mathcal{I}$. The graph $\mathbb{G}^0$ is then created in step 4 from $\mathcal{I}$ where the strongly connected components become the super-vertices of $\mathbb{G}^0$ and the super-edges correspond to a subset



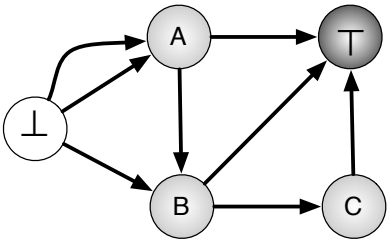
(a) Original graph $\mathcal{G}$ and the sequence $\mathcal{S} = \langle (v_1, 0), (v_6, 1), (v_9, 1) \rangle$



(b) Graph $\mathcal{H}$ (step 1 in Figure 3.1)



(c) Graph $\mathcal{I}$ (step 3 in Figure 3.1)



(d) PQ Graph $\mathbb{G}$ (step 4 in Figure 3.1)

Figure 3.2: Building a Picard-Queyranne graph

of flow edges from $\mathcal{F}$. Finally, in step 5, we create the PQ-graph $\mathbb{G}$ by "fixing" the source and target vertices so that they also have as elements the original labelled vertices from $\mathcal{S}$ which were merged in step 1. The correctness of the algorithm follows from arguments in (Picard and Queyranne, 1980); an independent proof is provided in the supplementary materials of Herbster et al. (2015).

Theorem 3.2 (Picard and Queyranne (1980))**.** *The algorithm in Figure 3.1 computes the unique Picard-Queyranne graph $\mathbb{G}(\mathcal{G}, \mathcal{S})$ derived from graph $\mathcal{G}$ and non-trivial example sequence $\mathcal{S}$.*

3.7 Mistake Bounds Analysis

In this section we analyze the mistakes incurred by the intractable `0-Ising` strategy (see Equation (3.10)) and the strategy `longest-path` (see Figure 3.4). Our analysis splits into two parts. Firstly, we show in (Section 3.7.1, Theorem 3.4), for a sufficiently *regular* graph label prediction algorithm, it is possible to analyze *independently* the mistake bound of each uniformly-labelled cluster (connected subgraph). Secondly, the per-cluster analysis then separates into three cases, the result of which is summarized in Theorem 3.10 in Section 3.7.3. For a given cluster $\mathcal{C}$, when its internal connectivity is larger than the number of edges in the boundary ($\kappa(\mathcal{C}) > |\partial_e^E(\mathcal{C})|$), we will incur no more than one mistake in that cluster. On the other hand for smaller connectivity clusters ($\kappa(\mathcal{C}) \leq |\partial_e^E(\mathcal{C})|$), we incur up to quadratically in mistakes via the edge boundary size. When $\mathcal{C}$ is a tree we incur $\mathcal{O}(|\partial_e^E(\mathcal{C})| \log D(\mathcal{C}))$ mistakes.

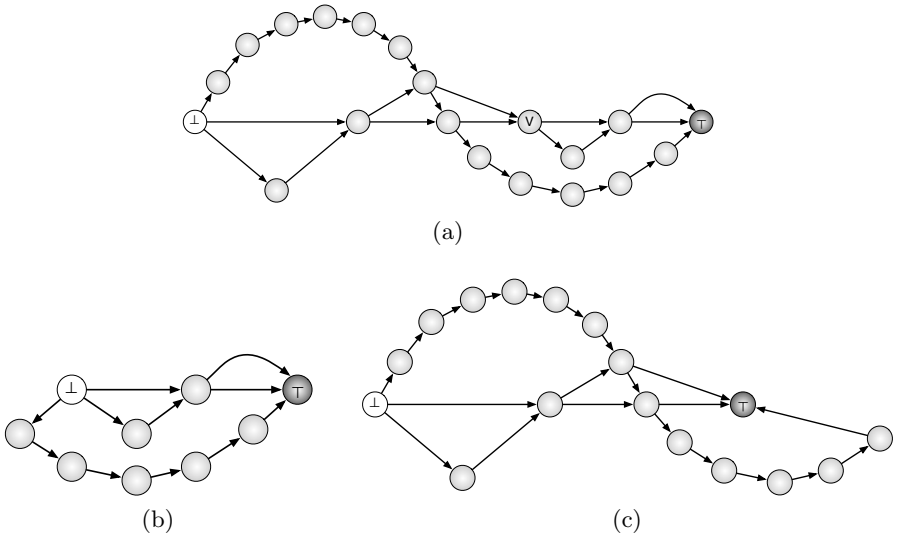


Figure 3.3: Illustration for PQ graph collapse upon prediction at queried vertex v. (a) Initial PQ graph and queried vertex v (b) Prediction at v with label 0. (c) Prediction at v with label 1. Image courtesy Mark Herbster.

The analysis of smaller connectivity clusters separates into two parts. First, a sequence of trials in which the label-consistent minimum-cut does not increase, we call a *PQ-game* (Section 3.7.2), as in essence it is played on a PQ-graph. We give a mistake bound for a PQ-game for the intractable `0-Ising` prediction and a comparable bound for the strategy `longest-path` in Theorem 3.8 in Section 3.7.2. Second, when the label-consistent minimum-cut increases, the current PQ-game ends and a new one begins, leading to a sequence of PQ-games. The mistakes incurred over a sequence of PQ-games is addressed in the aforementioned Theorem 3.10 and finally Section 3.7.3 concludes with a discussion of the combined bounds of Theorems 3.4 and 3.10 with respect to other graph label prediction algorithms.

3.7.1 Per-cluster mistake bounds for *regular* graph label prediction algorithms

An algorithm is called *regular* if it is *permutation-invariant*, *label-monotone*, and *Markov*. An algorithm is *permutation-invariant* if the prediction at any time t does not depend on the order of the examples up to time t; *label-monotone* if for every example sequence if we insert an example "between" examples t and $t+1$ with label y then the prediction at time $t+1$ is unchanged or changed to y; and *Markov* with respect to a graph $\mathcal{G}$ if for any disjoint vertex sets P and Q and separating set R then the predictions in P are independent of the labels in Q given the labels of R. A subgraph is *uniformly-labelled* with respect to an example sequence iff the label of each vertex is the same and these labels are consistent with the example sequence. The following definition characterizes the worst-case example sequences for regular algorithms with respect to uniformly-labelled clusters.

Definition 3.3. Given an online algorithm $\mathcal{A}$ and a uniformly-labelled subgraph $\mathcal{C} \subseteq \mathcal{G}$, then $\mathcal{B}_{\mathcal{A}}(\mathcal{C}; \mathcal{G})$ denotes the maximal mistakes made only in $\mathcal{C}$ for the presentation of any permutation of examples in $\partial_e(\mathcal{C})$, each with label y, followed by any permutation of examples in $\mathcal{C}$, each with label $1-y$.

The following theorem enables us to analyze the mistakes incurred in each uniformly-labelled subgraph $\mathcal{C}$ *independently* of each other and *independently* of the remaining graph structure excepting the subgraph's exterior border $\partial_e(\mathcal{C})$.

Theorem 3.4 (Proof in the supplementary materials of Herbster et al. (2015))**.** *Given an online permutation-invariant label-monotone Markov algorithm $\mathcal{A}$ and a graph $\mathcal{G}$ which is covered by uniformly labelled subgraphs $\mathcal{C}_1, \ldots, \mathcal{C}_c$ the mistakes incurred by the algorithm may be bounded by $M \leq \sum_{i=1}^{c} \mathcal{B}_{\mathcal{A}}(\mathcal{C}_i; \mathcal{G})$.*

The above theorem paired with Theorem 3.10 in Section 3.7.3 completes the mistake bound analysis of our algorithms.

3.7.2 PQ-games

Given a PQ-graph $\mathbb{G} = \mathbb{G}(\mathcal{G}, \mathcal{S})$, the derived online PQ-game is played between a `player` and an `adversary`. The aim of the player is to minimize their mistaken predictions; for the adversary it is to maximize the player's mistaken predictions. Thus to play the adversary proposes a vertex $z \in Z \in V(\mathbb{G})$, the player then predicts a label $\hat{y} \in \{0, 1\}$, then the adversary returns a label $y \in \{0, 1\}$ and either a *mistake* is incurred or not. The only restriction on the adversary is to not return a label which increases the label-consistent minimum-cut. As long as the adversary does not give an example $(z \in \bot, 1)$ or $(z \in \top, 0)$, the label-consistent minimum-cut does not increase no matter the value of y; which also implies the player has a trivial strategy to predict the label of $z \in \bot \cup \top$. After the example is given, we have an updated PQ-graph with new source and target super-vertices as seen in the proposition below.

Proposition 3.5. *If $\mathbb{G}(\mathcal{G}, \mathcal{S})$ is a PQ-graph and $(z, y = 0)$ $(z, y = 1)$ is an example with $z \in Z \in V(\mathbb{G})$ and $z \notin \top$ ($z \notin \bot$) then let $\mathbb{Z} = \downarrow\{Z\}$ ($\mathbb{Z} = \uparrow\{Z\}$) then $\mathbb{G}(\mathcal{G}, \langle \mathcal{S}, (z, y) \rangle) =$ `merge`$(\mathbb{G}(\mathcal{G}, \mathcal{S}), \mathbb{Z})$.*

Thus given the PQ-graph $\mathbb{G}$ the PQ-game is independent of $\mathcal{G}$ and $\mathcal{S}$, since a "play" $z \in V(\mathcal{G})$ induces a "play" $Z \in V(\mathbb{G})$ (with $z \in Z$).

Mistake bounds for PQ-games. Given a *single* PQ-game, in the following we will discuss the three strategies `fixed-paths`, `0-Ising`, and `longest-path` that the `player` may adopt for which we prove online mistake bounds. The first strategy `fixed-paths` is merely motivational: it can be used to play a *single* PQ-game, but not a sequence. The second strategy `0-Ising` is computationally infeasible. Finally, the `longest-path`

strategy is "dynamically" similar to `fixed-paths` but is also permutation-invariant. Common to all our analyses is a *k-path cover* P of PQ-graph $\mathbb{G}$ which is a partitioning of the edge-set of $\mathbb{G}$ into k edge-disjoint directed paths $P := \{p^1, \ldots, p^k\}$ from $\bot$ to $\top$. Note that the cover is not necessarily unique; for example, in Figure 3.2d, we have the two unique path covers $P_1 := \{(\bot, A, \top), (\bot, A, B, \top), (\bot, B, C, \top)\}$ and $P_2 := \{(\bot, A, \top), (\bot, A, B, C, \top), (\bot, B, \top)\}$. We denote the set of all path covers as $\mathcal{P}$ and thus we have for Figure 3.2d that $\mathcal{P} := \{P_1, P_2\}$. This cover motivates a simple mistake bound and strategy. Suppose we had a single path of length $|p|$ where the first and last vertex are the "source" and "target" vertices. So the minimum label-consistent cut-size is "1" and a natural strategy is simply to predict with the "nearest-neighbor" revealed label and trivially our mistake bound is $\log |p|$. Generalizing to multiple paths we have the following strategy.

Strategy `fixed-paths`$(\widetilde{P})$**:** Given a PQ-graph choose a path cover $\{\tilde{p}^1, \ldots, \tilde{p}^k\} = \widetilde{P} \in \mathcal{P}(\mathbb{G})$. If the path cover is also vertex-disjoint except for the source and target vertex we may directly use the "nearest-neighbor" strategy detailed above, achieving the mistake upper bound $M \leq \sum_{i=1}^{k} \log |\tilde{p}^i|$. Unsurprisingly, in the vertex-disjoint case it is a mistake-bound optimal (Littlestone, 1988) algorithm. If, however, $\widetilde{P}$ is not vertex-disjoint and we need to predict a vertex V we may select a path in $\widetilde{P}$ containing V and predict with the nearest neighbour and also obtain the bound above. In this case, however, the bound may not be "optimal". Essentially the same technique was used in (Gärtner and Garriga, 2007) in a related setting for learning "directed cuts". A limitation of the `fixed-paths` strategy is that it does not seem possible to extend into a strategy that can play a *sequence* of PQ-games and still meet the regularity properties, particularly permutation-invariance as required by Theorem 3.4.

Strategy `0-Ising`**:** The prediction of the Ising model in the limit of zero temperature (cf. Equation (3.10)), is equivalent to those of the well-known *Halving* algorithm (Barzdin and Frievald, 1972; Littlestone and Warmuth, 1994) where the hypothesis class $\mathcal{U}^*$ is the set of label-consistent minimum-cuts. The mistake upper bound of the *Halving algorithm* is just $M \leq \log |\mathcal{U}^*|$ where this bound follows from the observation that whenever a mistake is made at least "half" of concepts in $\mathcal{U}^*$ are no longer consistent. We observe that we may upper bound $|\mathcal{U}^*| \leq \operatorname{argmin}_{P \in \mathcal{P}(\mathbb{G})} \prod_{i=1}^{k} |p^i|$ since the product of path lengths from *any* path cover P is an upper bound on the cardinality of $\mathcal{U}^*$ and hence we have the bound in (3.12). And in fact this bound may be a significant improvement over the `fixed-paths` strategy's bound as seen in the following proposition.

Proposition 3.6 (Proof in the supplementary materials of Herbster et al. (2015))**.** *For every $c \geq 2$ there exists a PQ-graph $\mathbb{G}_c$, with a path cover $P' \in \mathcal{P}(\mathbb{G}_c)$ and a PQ-game example sequence such that the mistakes $M_{\texttt{fixed-paths}(P')} = \Omega(c^2)$, while for all PQ-game example sequences on $\mathbb{G}_c$ the mistakes $M_{\texttt{0-Ising}} = \mathcal{O}(c)$.*

Unfortunately the `0-Ising` strategy has the drawback that counting label-consistent minimum-cuts is #P-complete and computing the prediction (see Equation (3.10)) is NP-hard (see proof in the supplementary materials of Herbster et al. (2015)).

Strategy `longest-path`: In our search for an efficient and *regular* prediction strategy it seems natural to attempt to "dynamize" the `fixed-paths` approach and predict with a nearest neighbor along a dynamic path. Two such permutation-invariant methods are the `longest-path` and `shortest-path` strategies. The strategy `shortest-path` predicts the label of a super-vertex Z in a PQ-game $\mathbb{G}$ as 0 iff the shortest directed path $(\bot, \ldots, Z)$ is shorter than the shortest directed path $(Z, \ldots, \top)$. The strategy `longest-path` predicts the label of a super-vertex Z in a PQ-game $\mathbb{G}$ as 0 iff the longest directed path $(\bot, \ldots, Z)$ is shorter than the longest directed path $(Z, \ldots, \top)$. The

Input: Graph: $\mathcal{G}$, Example sequence: $\mathcal{S} = \langle (i_1, 0), (i_2, 1), (i_3, y_3), \ldots, (i_\ell, y_\ell) \rangle \in (\mathbb{N}_n \times \{0,1\})^\ell$
Initialization: $\mathbb{G}_3 = \texttt{PicardQueyranneGraph}(\mathcal{G}, \mathcal{S}_2)$
for $t = 3, \ldots, \ell$ **do**
  Receive: $i_t \in \{1, \ldots, n\}$
  $I_t = V \in V(\mathbb{G}_t)$ with $i_t \in V$
  Predict (`longest-path`):

$$\hat{y}_t = \begin{cases} 0 & |\texttt{longest-path}(\mathbb{G}_t, \bot_t, I_t)| \leq |\texttt{longest-path}(\mathbb{G}_t, I_t, \top_t)| \\ 1 & \textbf{otherwise} \end{cases}$$

  Predict (`0-Ising`): $\hat{y}_t = \hat{y}^{\text{IO}}(i_t | \mathcal{S}_{t-1})$ % as per equation (3.10)
  Receive: y_t
  if ($i_t \notin \bot_t$ or $y_t \neq 1$) and ($i_t \notin \top_t$ or $y_t \neq 0$) **then** % cut unchanged

$$\mathbb{G}_{t+1} = \begin{cases} \texttt{merge}(\mathbb{G}_t, \downarrow\{I_t\}) & y_t = 0 \\ \texttt{merge}(\mathbb{G}_t, \uparrow\{I_t\}) & y_t = 1 \end{cases}$$

  else % cut increases
    $\mathbb{G}_{t+1} = \texttt{PicardQueyranneGraph}(\mathcal{G}, \mathcal{S}_t)$
end

Figure 3.4: `Longest-path` and `0-Ising` online prediction

strategy `shortest-path` seems to be intuitively favored over `longest-path` as it is just the "nearest-neighbor" prediction with respect to the geodesic distance. However, the following proposition shows that it is strictly worse than any `fixed-paths` strategy in the worst case.

Proposition 3.7 (Proof in the supplementary materials of Herbster et al. (2015))**.** *For every $c \geq 4$ there exists a PQ-graph $\mathbb{G}_c$ and a PQ-game example sequence such that the mistakes $M_{\texttt{shortest-path}} = \Omega(c^2 \log(c))$, while for every path cover $P \in \mathcal{P}(\mathbb{G}_c)$ and for all PQ-game example sequences on $\mathbb{G}_c$ the mistakes $M_{\texttt{fixed-paths}(P)} = \mathcal{O}(c^2)$.*

In contrast, for the strategy `longest-paths` in the proof of Theorem 3.8 on the next page, we show that there always exists some retrospective path cover $P_{\text{lp}} \in \mathcal{P}(\mathbb{G})$ such that $M_{\texttt{longest-paths}} \leq \sum_{i=1}^{k} \log |p^i_{\text{lp}}|$. Computing the "longest-path" has time complexity linear in the number of edges in a DAG.

Summarizing the mistake bounds for the three PQ-game strategies for a single PQ-game we have the following theorem.

Theorem 3.8 (Proof in the supplementary materials of Herbster et al. (2015))**.** *The mistakes, M, of an online PQ-game for player strategies* `fixed-paths`$(\widetilde{P})$, `0-Ising`, *and* `longest-path` *on PQ-graph $\mathbb{G}$ and k-path cover $\widetilde{P} \in \mathcal{P}(\mathbb{G})$ is bounded by*

$$M \leq \begin{cases} \sum_{i=1}^{k} \log |\tilde{p}^i| & \texttt{fixed-paths}(\widetilde{P}) \\ \operatorname{argmin}_{P \in \mathcal{P}(\mathbb{G})} \sum_{i=1}^{k} \log |p^i| & \texttt{0-Ising} \\ \operatorname{argmax}_{P \in \mathcal{P}(\mathbb{G})} \sum_{i=1}^{k} \log |p^i| & \texttt{longest-path} \end{cases} . \tag{3.12}$$

The detailed proof of 3.8 is in the supplementary materials of Herbster et al. (2015), here we provide a proof sketch for the sake of completion.

Proof Sketch: Without loss of generality over the classes, the theorem can be proven by means of reverse induction, where M_t denotes the backward cumulative mistakes. The total number of mistakes at any trial is the sum over the individual mistakes made on the set of k edge disjoint path cover for the PQ graph at that trial. If $\mathbb{G}_t$ is the PQ graph at trial t, the set of k edge disjoint paths is $\{p_t^1, p_t^2, \ldots, p_t^t\}$. For the base case at $t = T$, the number of mistakes made $M_T = 0$, so the inductive hypothesis holds as $M_T = 0 = \sum_{i=1}^{k} M_T^i$ where $M_T^i = 0 \leq \log |p_T^i|$. The goal is to show the inductive hypothesis holds for some $t > 1$, say $t - 1$. If a mistake has not been made between trial t to trial $t - 1$ and the PQ graph stays the same from trial t to trial $t - 1$, the inductive hypothesis holds as the path cover stays the same. However, if the PQ graph has changed since the last trial t and a mistake is made at trial $t - 1$ on queried vertex Z_b, we show the following. By PQ graph construction before and after, we have the state of the PQ graph at trial $t - 1$ as shown in (a) with dotted lines, and the state of the graph at trial t after the PQ graph collapse shown in bold lines in (b). Let the path $\hat{p}_{t-1}^i$ be defined in the PQ graph at trial $t - 1$, such that the path $\hat{p}_{t-1}^i = p_t^i + \{(Z_a, Z_b)\} + r_{t-1}^{\top}$, belongs to the set of k edge disjoint paths in trial $t - 1$. $r_{t-1}^{\perp}$ is the segment of the path from the downstream source vertex $\perp$ to Z_b, while $r_{t-1}^{\top}$ is the upstream path from Z_b to target vertex $\top$. Now if the longest-path had made a mistake predicting the label of Z_b as 0, without loss of generality, mistakes made in $t - 1$: $M_{t-1} = M_t + 1$, then by inductive hypothesis, $M_{t-1} = 1 + \sum_{i=1}^{k} M_t^i$, which implies $M_{t-1} = \sum_{i=1}^{k} M_{t-1}^i$, where there are k edge disjoint paths in the PQ graph at trial $t - 1$. Also, since longest path made a mistake, $\left|r_{t-1}^{\perp}\right| \leq \left|r_{t-1}^{\top}\right|$, which implies, $\left|\hat{p}_{t-1}^i \setminus r_{t-1}^{\top}\right| \leq \left|r_{t-1}^{\top}\right|$. Also, length of the path $\left|\hat{p}_{t-1}^i\right| = \left|\hat{p}_{t-1}^i \setminus r_{t-1}^{\top}\right| + \left|r_{t-1}^{\top}\right|$ which implies that the length of the path $\left|\hat{p}_{t-1}^i\right| \geq 2\left|\hat{p}_{t-1}^i \setminus r_{t-1}^{\top}\right|$. From the inductive hypothesis, this implies $2\left|\hat{p}_{t-1}^i \setminus r_{t-1}^{\top}\right| = 2\left|p_t^i\right| \geq 2 \times 2^{M_t^i} = 2^{1+M_t^i} = 2^{M_{t-1}^i}$. Since this holds for all the paths in uncollapsed PQ graph at trial $t - 1$, this holds for all cases. If the longest path did not make a mistake in trial $t-1$, when the PQ graph changed from trial t, on Z_b, the number of mistakes stay the same and its trivial to show the equality.

3.7.3 Global analysis of prediction at zero temperature

In Figure 3.4 we summarize the prediction protocol for `0-Ising` and `longest-path`. We claim the regularity properties of our strategies in the following theorem.

Theorem 3.9 (Proof in the supplementary materials of Herbster et al. (2015))**.** *The strategies* `0-Ising` *and* `longest-path` *are permutation-invariant, label-monotone, and Markov.*

The technical hurdle here is to prove that label-monotonicity holds over a sequence of PQ-games. For this we need an analog of Proposition 3.5 to describe how the PQ-graph changes when the label-consistent minimum-cut increases. The application of the following theorem along with Theorem 3.4 implies we may bound the mistakes of each uniformly-labelled cluster in potentially three ways.

Theorem 3.10 (Proof in the supplementary materials of Herbster et al. (2015))**.** *Given either the* `0-Ising` *or* `longest-path` *strategy $\mathcal{A}$ the mistakes on uniformly-labelled subgraph $\mathcal{C} \subseteq \mathcal{G}$ are bounded by*

$$\mathcal{B}_{\mathcal{A}}(\mathcal{C};\mathcal{G}) \in \begin{cases} \mathcal{O}(1) & \kappa(\mathcal{C}) > |\partial_e^E(\mathcal{C})| \\ \mathcal{O}\left(|\partial_e^E(\mathcal{C})|(1+|\partial_e^E(\mathcal{C})|-\kappa(\mathcal{C}))\log N(\mathcal{C})\right) & \kappa(\mathcal{C}) \leq |\partial_e^E(\mathcal{C})| \\ \mathcal{O}(|\partial_e^E(\mathcal{C})|\log D(\mathcal{C})) & \mathcal{C} \text{ is a tree} \end{cases} \tag{3.13}$$

with the atomic number $N(\mathcal{C}) := \mathcal{N}_{|\partial_e^E(\mathcal{C})|+1}(\mathcal{C}) \leq |V(\mathcal{C})|$.

First, if the internal connectivity of the cluster is high we will only make a single mistake in that cluster. Second, if the cluster is a tree then we pay the external connectivity of the cluster $|\partial_e^E(\mathcal{C})|$ times the log of the cluster diameter. Finally, in the remaining case we pay quadratically in the external connectivity and logarithmically in the "atomic number" of the cluster. The atomic number captures the fact that even a poorly connected cluster may have sub-regions of high internal connectivity. Here, we provide an intuitive proof sketch for the Theorem 3.10, where the detailed proof is provided in the supplementary materials of the chapter (Herbster et al., 2015).

Proof Sketch The goal is to prove the mistake bounds that capture the connectivity structure of the underlying graph. Since, we assume uniformly labelled clusters as shown in Figure 3.5 with dotted lines, the first step is taken towards bounding the number of mistakes made within the clusters C_i as $M \leq \sum_{i=1}^{c} \mathcal{B}_{\mathcal{A}}(\mathcal{C}_i;\mathcal{G})$, where there are c clusters, $\mathcal{A}$ is the `longest-path` algorithm. Further, once the mistakes are bounded within the clusters, the introduced parameter atomic number $N(\mathcal{C})$, is used to tune the inter-cluster connectivity within the graph relative to the intra-cluster connectivity through κ, which is defined as the number of edge disjoint paths connecting a vertex pair. For example in the Figure 3.5, the atomic number at $\kappa = 2$ connectivity level is $N_2(\mathcal{G}) = 2$; the

atomic number partitions the graph into two sub graphs each with internal connectivity more than equal to 2 as shown by the purple dotted lines. With $\kappa = 3$, $N_3(\mathcal{G}) = 6$; the atomic number partitions the graph into 6 sub graphs. each with internal connectivity more than equal to 3 as shown with the orange dotted lines. Hence, when the bounds in Theorem 3.10 can be divided into cases, when the intra cluster connectivity is more than the inter cluster connectivity, when the intra cluster connectivity is less than the inter cluster connectivity and when the cluster is a tree.

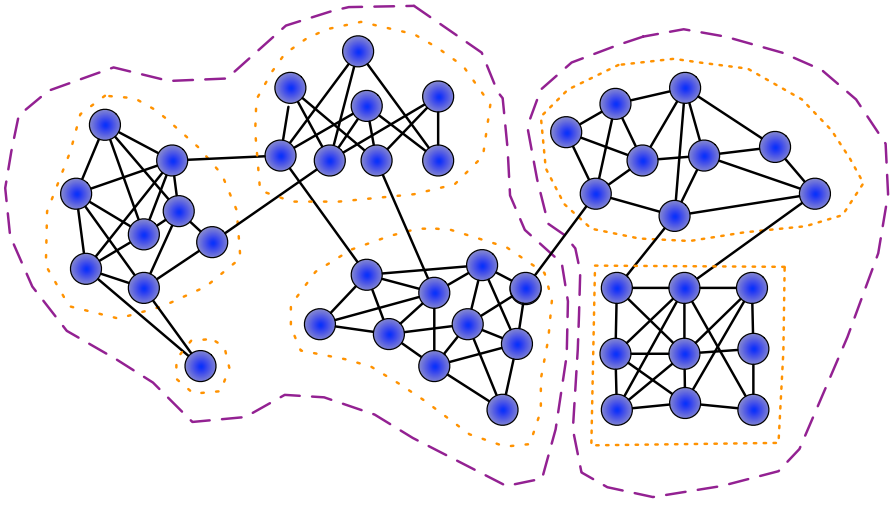


Figure 3.5: Dense cluster structure in sparse graphs tuned by atomic number of the graph $N_(\mathcal{G})$. Image courtesy Mark Herbster.

Computational complexity. If $\mathcal{G}$ is a graph and $\mathcal{S}$ an example sequence with a label-consistent minimum-cut of ϕ then we may implement the `longest-path` strategy so that it has a cumulative computational complexity of $\mathcal{O}(\max(\phi, n)\,|E(\mathcal{G})|)$. This follows because if on a trial the "cut" does not increase we may implement prediction and update in $\mathcal{O}(|E(\mathcal{G})|)$ time. On the other hand if the "cut" increases by ϕ' we pay $\mathcal{O}(\phi'|E(\mathcal{G})|)$ time. So we implement an online "Ford-Fulkerson" algorithm (Ford and Fulkerson, 1956) which starts from the previous "residual" graph to which it then adds the additional ϕ' flow paths with ϕ' steps of size $\mathcal{O}(|E(\mathcal{G})|)$.

Discussion. There are essentially five dominating mistake bounds for the online graph labelling problem: (I) the bound of `treeOpt` (Cesa-Bianchi et al., 2009) on trees, (II) the bound in expectation of `treeOpt` on a random spanning tree sampled from a graph (Cesa-Bianchi et al., 2009), (III) the bound of `p-seminorm interpolation` (Herbster and Lever, 2009) tuned for "sparsity" ($p < 2$), (IV) the bound of `p-seminorm interpolation` as tuned to be equivalent to online label propagation (Zhu et al., 2003) ($p = 2$), (V) this chapter's `longest-path` strategy.

The algorithm `treeOpt` was shown to be optimal on trees. In the proof in the chapter, we show that `longest-path` also obtains the same optimal bound on trees. Algorithm (II) applies to generic graphs and is obtained from (I) by sampling a random spanning

tree (RST). It is not directly comparable to the other algorithms as its bound holds only in *expectation* with respect to the RST.

We use (Herbster and Lever, 2009, Corollary 10) to compare (V) to (III) and (IV). We introduce the following simplifying notation to compare bounds. Let $\mathcal{C}_1, \ldots, \mathcal{C}_c$ denote uniformly-labelled clusters (connected subgraphs) which cover the graph and set $\kappa_r := \kappa(\mathcal{C}_r)$ and $\phi_r := |\partial_e^E(\mathcal{C}_r)|$. We define $D_{r(i)}$ to be the *wide diameter* at connectivity level i of cluster $\mathcal{C}_r$. The wide diameter $D_{r(i)}$ is the minimum value such that for all pairs of vertices $v, w \in \mathcal{C}_r$ there exists i edge-disjoints of paths from v to w of length *at least* $D_{r(i)}$ in $\mathcal{C}_r$ (and if $i > \kappa_r$ then $D_{r(i)} := +\infty$). Thus $D_{r(1)}$ is the diameter of cluster $\mathcal{C}_r$ and $D_{r(1)} \leq D_{r(2)} \leq \cdots$. Let ϕ denote the minimum label-consistent cutsize and observe that if the cardinality of the cover $|\{\mathcal{C}_1, \ldots, \mathcal{C}_c\}|$ is minimized then we have that $2\phi = \sum_{r=1}^{c} \phi_r$.

Thus using (Herbster and Lever, 2009, Corollary 10) we have the following upper bounds of (III): $(\phi/\kappa^*)^2 \log D^* + c$ and (IV): $(\phi/\kappa^*) D^* + c$ where $\kappa^* := \min_r \kappa_r$ and $D^* := \max_r D_r(\kappa^*)$. In comparison we have (V): $[\sum_{r=1}^{c} \max(0, \phi_r - \kappa_r + 1)\phi_r \log N_r] + c$ with atomic numbers $N_r := \mathcal{N}_{\phi_r+1}(\mathcal{C}_r)$. To contrast the bounds, consider a double lollipop labelled-graph: first create a lollipop which is a path of $n/4$ vertices attached to a clique of $n/4$ vertices. Label these vertices 1. Second, clone the lollipop except with labels 0. Finally join the two cliques with $n/8$ edges arbitrarily. For (III) and (IV) the bounds are $\mathcal{O}(n)$ independent of the choice of clusters. Whereas an upper bound for (V) is the exponentially smaller $\mathcal{O}(\log n)$ which is obtained by choosing a four cluster cover consisting of the two paths and the two cliques. This emphasizes the generic problem of (III) and (IV): parameters κ^* and D^* are defined by the worst clusters; whereas (V) is truly a per-cluster bound. We consider the previous "constructed" example to be representative of a generic case where the graph contains clusters of many resistance diameters as well as sparse interconnecting "background" vertices.

On the other hand, there are cases in which (III,IV) improve on (V). For a graph with only small diameter clusters and if the cutsize exceeds the cluster connectivity then (IV) improves on (III,V) given the linear versus quadratic dependence on the cutsize. The log-diameter may be arbitrarily smaller than log-atomic-number ((III) improves on (V)) and also vice-versa. Other subtleties not accounted for in the above comparison include the fact a) the wide diameter is a crude upper bound for resistance diameter (cf. (Herbster and Lever, 2009, Theorem 1)) and b) the clusters of (III,IV) are not required to be uniformly-labelled. Regarding "a)" replacing "wide" with "resistance" does not change the fact the bound now holds with respect to the worst resistance diameter and the example above is still problematic. Regarding "b)" it is a nice property but we do not know how to exploit this to give an example that significantly improves (III) or (IV) over a slightly more detailed analysis of (V). Finally (III,IV) depend on a correct choice of tunable parameter p.

Thus in summary (V) matches the optimal bound of (I) on trees, and can often improve on (III,IV) when a graph is naturally covered by label-consistent clusters of different diameters. However (III,IV) may improve on (V) in a number of cases including when the log-diameter is significantly smaller than log-atomic-number of the clusters.

3.8 Experiments

In this section, we provide an experimental validation of our theoretical results on the `longest-path`, by conducting experiments on multiple standard machine learning and simulated datasets.

3.8.1 Simulation Data

Our synthetic dataset uses a 2D grid like topology. The motivation behind using the grid images for our simulation experiments stems from the naturally occurring graph structure in such 2D grids. Our selection of images vary in smoothness, complexity and patterns of class membership. We use a square image that is constructed using a set of pixels, each with an intensity of 0 or 1. The 0 and 1 intensities are balanced across the pixels i.e. there are equal number of pixels with 0 and 1 intensities. When the graphs are constructed from the images, each pixel in the image corresponds to a vertex in the graph and the intensities correspond to the label or class of the vertex. There are 3600 or 1600 vertices in the graphs generated depending on the original size of the problem. The neighbourhood system in the graph comprises edges connecting pairs of adjacent pixels. The connectivity is typically guided by the pixel locations. In this chapter, we are only interested in undirected and unweighted graphs. The graphs thus generated have a weight of 1 on every edge and there is an edge in either direction. Further, we investigate two types of neighbourhood system, namely torus and non-torus. In the torus graph, each pixel has four neighbours; achieved by wrapping the image from top to bottom and from left to right, thereby creating the twists and rings on the torus. In the non-torus graph, the corner pixels have two neighbours, pixels on edges other than the corner pixels have three neighbours and the remaining pixels have four neighbours. Our simulation experiments run over ten trials. We feed the same graph in every trial. We randomly sample the labelled vertices from the graph such that there are equal number of labels from each class. We vary the training labels available through $6\%, 12\%, 18\%, 24\%, 30\%$ of the graph size. For every set of partially labelled vertices, the algorithms are analysed on their predictive performance over the unlabelled vertices and their performance averaged over ten trials. Here, we report the generalization error or accuracy of the algorithms over the graph size i.e. we also take into account the labelled vertices when reporting the accuracy.

Table 3.1: Datasets used in this 0-temp Ising Model.

Data set	#Instances	#Features	#Classes
USPS	7291	256	10
Isolet	7797	617	26
Webspam	9072	100	2
20Newsgroup	16242	100	4

3.8.2 Datasets

We perform several experiments on real-world graphs to evaluate the performance of our algorithm against the state-of-the-art. The datasets come from different domains, these are: handwritten digit recognition; text categorization; web spam filtering; and noisy, perceptual spoken letter recognition.

The `USPS` handwritten digits from the UCI (Lichman, 2013) machine learning repository, is an optical character recognition dataset comprising 16×16 grayscale images of digits 0 through 9 obtained from scanning handrwritten digits. The pre-processed dataset has each image with 256 real valued features without missing values scaled to $[-1, 1]$. We randomly sample the examples for the graph from the 7291 original training points. Each vertex in the graph thus sampled is a digit. We perform several binary classification tasks of one digit versus the other digit.

We use a noisy perceptual dataset for spoken letter recognition called `Isolet` from the UCI datasets, consisting of 7797 instances with 617 real valued features. A total of 150 subjects spoke each letter of the English alphabets twice resulting in 52 training examples from each speaker. The 150 speakers are split into sets of 30 speakers each, named as `Isolet 1` through to `Isolet 5`. We are only interested in binary classification tasks where we classify the first 13 spoken letters against the last 13 spoken letters.

The `Webspam` dataset is available as part of 2007 the `Webspam` Challenge website (Webspam, 2007) at the University of Paris VI. Pre-processed graphs are provided in the form of a smaller host-graph of 9072 vertices and a larger web graph of $400,000$ vertices. Here, we use the host graph for our experiments, with all its 9072 vertices and $464,959$ edges, where the vertices represent computer hosts. The hosts are connected if there is a link between the web page on one host and the web page on the other host. Each vertex in the graph has a label associated, which indicates whether the host is spam or normal.

The original text categorization dataset called the `20 Newsgroup` dataset, has $200,000$ messages across 20 news categories. We use a subset of the `20 Newsgroup` dataset that was put together by Sam Roweis (Roweis, 2006) that consisted of text documents from four news groups namely (comp.*, rec.*, sci.* and talk.* usenet hierarchies). Each document is represented by 100 binary features that indicates whether the 100 non-stop

words are present or absent in the document. There are 16242 examples in this subset, and we use all of them in our graphs. The classification task that we use is of the nature of one newsgroup versus the rest.

3.8.3 Algorithms

The discriminative benchmarks that we compare our algorithm against are as follows. `labelProp` (Zhu et al., 2003; Belkin and Niyogi, 2004) is a label propagation batch transductive learning algorithm that solves a linear system of equations involving the graph Laplacian. This is equivalent to the harmonic energy minimization where `labelProp` essentially solves a quadratic program $\boldsymbol{u}^T \boldsymbol{L} \boldsymbol{u}$ that uses the spectral kernel of the graph structure or the graph Laplacian $\boldsymbol{L}$ and the training labels to infer the final labelling $\boldsymbol{u}$ of the graph. For our experiments, we have implemented the algorithm ourselves in Matlab.

`treeopt` Shazoo (Cesa-Bianchi et al., 2009) is an optimal label prediction algorithm on any weighted tree. Here, we use the unweighted version of Shazoo as our second benchmark. We have adapted the Shazoo code written in Java that has been provided to us by the original authors. Our input to the algorithm is the graph that we generate from the data and training labels. Shazoo internally builds its random spanning trees on the graph inputted.

3.8.4 Graph Construction

Throughout, we construct the graph from the dataset as a separate step. The graph generation process is uniform across all the datasets, varying slightly on the training label size. All graphs are undirected and unweighted (there is an edge in either direction) with a weight of 1 assigned to the edge. The graph construction process ensures that there are nearly balanced instances from each of the classes. For each dataset after the balanced sampling of the instances representative of each class, a "cost" matrix is computed for storing the distance among all examples (patterns) in the sampled graph. The Euclidean distance metric gets used for computing the distance matrix except in the case of `ISOLET` where the "cosine distance" is used. This is followed by the creation of an unweighted minimum spanning tree (MST) and a "3-NN" graph (via the cost matrix) where the the edge sets of the MST and 3-NN get "unioned" together to form the final graph for each of the datasets. The rationale behind selecting the last step in the graph generation methodology is based on the common empirical observation that 3-NN graphs are often among the most competitive of the unweighted k-NN graphs. The added MST edges ensure that the final graph is connected. Further, MST graphs have been noted to have very good empirical performance as compared to k-NN graphs, see for example (Vitale et al., 2011).

The result of the graph construction process is the creation of relatively sparse graphs that reduce the computational burden for all methods. Additionally, the uniform process of graph creation, reduces the variance by avoiding model selection. Although it is beyond the scope of our limited study, it may be the case that constructed graphs with higher connectivity could potentially lead to higher accuracies.

The parameters that we use in the experiments to vary through are K for the connectivity, L for the number of available labels and N for the size of the graph. In `ISOLET`, we build a 3 nearest neighbour graph from the Euclidean distance matrix constructed using the pairwise distances between the instances (spoken letters) in the dataset as vertices in the graph. In order to ensure that the graph is connected for such low connectivity, we sample a MST for each graph and always maintain the MST edges in the graph. The MST uses the Euclidean distances as weights. The graph thus built, remains the same for all ten trials of the experiment. The training labels are sampled randomly such that the two classes are balanced.

For the `Webspam` experiment, we use the fully pre-processed graph available on the WebSpam Challenge 2007 website (Webspam, 2007). We use the link matrix with a non zero entry for an edge in the hostgraph of 9072 vertices. As of now, we drop the weights to build our adjacency matrix. The graph thus sampled remains the same across all ten trials of the `Webspam` experiment. However, we randomly sample l training labels (label of 1 indicated spam and 0 for normal) at every trial, such that both spam and non-spam labels are balanced and the algorithm is asked to predict on the unlabelled vertices.

We perform the text categorization experiment on the subset of the `20 Newsgroup` as described before. The binary problems that we select are the one vs rest i.e. Comp.* vs. Rec.*, Sci.* and Talk.* classification tasks between newsgroups. Each document is characterized by 100 logical or binary features, indicating if the 100 non-stop words are present or absent. We calculate the cosine distance between pairs of documents and the documents. In our current experiments, we use the MST edges and K = 3. The MST uses the cosine distance metric as weights. The same graph is used across all the trials.

In the USPS experiments, we sample a different graph for each trial. While sampling the vertices of the graph, we ensure to select vertices equally from each class. Primarily, our experiments classifies one digit versus the other. We vary the training labels through almost $1\%, 2\%, 3\%, 6\%$ and 12% with equal number of labels from each class. We use the pairwise Euclidean distance as the weights for the MST construction. All the sampled graphs maintain the MST edges. We vary k values through an exhaustive range of $K = 3, 4, 5, 6, 7, 8, 9, 10, 11, 12$. Here, we report the results of $K = 3, N = 1000$.

We perform our experiments on a single Intel i7 64 bit processor with 4 cores Windows laptop, having 8GB RAM. The extensive set of experiments are carried out on the IRIDIS High Performance Computing Facility, and associated support services at the University of Southampton.

3.9 Preliminary Experimental Results

In this section we present some preliminary experiments that compare the `longest-path` strategy to `treeOpt` and label propagation `LabProp`. The datasets include the four standardized benchmark datasets `USPS 2 vs 3`, `3 vs 8`, `20 newsgroups` and `ISOLET` as well as a constructed dataset `Stripes`. We used our own implementation of `longest-path` and `labelProp`. For the purposes of computational efficiency we ran our experiments in the "batch mode" rather than "online."

We used the benchmark datasets as follows. With the `USPS` datasets we sampled 500 digits from each class. For `ISOLET` we combined "`ISOLET1`" to "`ISOLET5`" giving 3900 in class "0" (letters 'A-M') and 3897 class "1" (letters 'N-Z') examples. While for `20 Newsgroups` we combined "comp.*" and "rec.*" creating class "0" with 8124 examples and combining "sci.*" and "talk.*" creating class "1" with 8118 examples.

In Figure 3.2 we report our results. We give the mean accuracy (computed over *all* labels in the graph) and its standard deviation from ten runs. For each "column," and each run of 10, we sampled uniformly $\ell/2$ labels from each class. For the `USPS` datasets we also randomly sampled and built a new graph on each run. Finally on each run as `treeOpt` expects a tree we further sampled a uniform random spanning tree as per (Cesa-Bianchi et al., 2009) from the built graph on each of the 10 runs.

Our obervations are as follows. `LabelProp` performs systematically well across all datasets. `treeOpt` tends to have the weakest performance. Note, however, that `treeOpt` is very computationally efficient and it is natural to run with an ensemble of trees to improve performance; this is discussed and experimentally confirmed in (Vitale et al., 2011). `Longest-path` is competitive and improves on `labelProp` often. But it has a "failure mode" as seen in the first column for the relatively smaller label sets. We observed that when this occurred we are finding small PQ-graphs corresponding to unbalanced trivial label-consistent minimum-cuts.

We also show results on a constructed dataset to illustrate the potential of the algorithm. `Stripes` is a 60×60 grid graph shown in Figure 3.6 with toroidal boundary connectivity. Thus each vertex has four neighbors. The problem corresponds to a simple geometric concept of "stripes." We induce the two classes by alternately "coloring" each of the 6 vertical stripes of 10×60 vertices. For this dataset the performance of `longest-path` strongly dominates. We provide a visualization of a typical PQ-graph from a `Stripes` in Figure 3.13.

Table 3.2: Experiments of 0-temp Ising Model

		$\ell = 8$	$\ell = 16$	$\ell = 32$	$\ell = 64$	$\ell = 128$
2 vs. 3	`labelProp`	**.980** ± .010	**.982** ± .008	.984 ± .005	**.988** ± .003	**.991** ± .002
	`treeOpt`	.814 ± .055	.885 ± .032	.891 ± .032	.956 ± .013	.959 ± .013
	`longest-path`	.504 ± .001	.940 ± .143	**.987** ± .003	**.988** ± .003	.990 ± .002
		$\ell = 8$	$\ell = 16$	$\ell = 32$	$\ell = 64$	$\ell = 128$
3 vs. 8	`labelProp`	**.956** ± .009	**.953** ± .007	.961 ± .007	.967 ± .004	.971 ± .004
	`treeOpt`	.797 ± .105	.749 ± .095	.878 ± .013	.935 ± .026	.960 ± .023
	`longest-path`	.505 ± .001	.600 ± .184	**.969** ± .006	**.971** ± .004	**.972** ± .003
		$\ell = 32$	$\ell = 128$	$\ell = 512$	$\ell = 1600$	$\ell = 2048$
Isolet	`labelProp`	**.661** ± .039	**.764** ± .024	.820 ± .012	.887 ± .006	.899 ± .004
	`treeOpt`	.658 ± .042	.731 ± .012	**.824** ± .008	.888 ± .007	.906 ± .002
	`longest-path`	.524 ± .033	.726 ± .016	.799 ± .016	**.906** ± .007	**.921** ± .006
		$\ell = 800$	$\ell = 1000$	$\ell = 2000$	$\ell = 4000$	$\ell = 6000$
News	`labelProp`	**.825** ± .005	**.826** ± .007	**.844** ± .004	**.871** ± .002	**.894** ± .003
	`treeOpt`	.753 ± .006	.758 ± .014	.804 ± .001	.847 ± .002	.878 ± .003
	`longest-path`	.549 ± .004	.798 ± .013	.839 ± .005	.867 ± .003	.890 ± .002
		$\ell = 250$	$\ell = 450$	$\ell = 650$	$\ell = 850$	$\ell = 1050$
Stripes	`labelProp`	.915 ± .013	.948 ± .005	.962 ± .004	.972 ± .004	.978 ± .005
	`treeOpt`	.817 ± .012	.879 ± .017	.909 ± .006	.928 ± .003	.936 ± .006
	`longest-path`	**.921** ± .090	**.998** ± .002	**.997** ± .002	**.994** ± .004	**.993** ± .002

3.10 Extensive Experimental Results

3.10.1 Simulated Grid Images

The experiments with simulated data involve using regular grid graphs that are lattice like approximations to torus. On a grid image, the pixel intensities vary between 0 and 1. The adjacent pixels are neighbours of each other in the corresponding graph constructed from the image. Further, we ensure a 4-connected torus while generating the graph, by wrapping around the boundary edges of the image from top to bottom and from left to right, hence forming the rings and twists in the torus as seen in the Figures 3.10 and 3.14. While constructing the simulated image data, our goal has been to progress from easier to difficult problem complexity and non-smooth to smooth data. In each of the non-torus graphs that we construct using the grid images, the following neighbourhood distribution is maintained. The corner cells have two neighbours, the cells on the boundary edges have three neighbours while any other intermediate cell has four neighbours.

The grid image of `Stripes` is a regular 60×60 4-connected lattice approximation to the torus. A stripe is a collection of consecutive columns of similar pixel intensity values within the grid. Stripes with pixel intensity value of 1 are interspersed with stripes with pixel intensity value of 0 forming three stripes of intensity 0 and three of intensity 1. The corresponding 4-connected torus constructed from the `Stripes` grid image is seen in Figures 3.10 and 3.11.

Figure 3.6: Simulated Grid Image `Stripes` with $N = 3600$.

The grid image of `Checker` used here is a 40×40 regular mesh with a checker-board pattern. The graph constructed from this grid has toroidal property. In total there are 16 squares with alternating pixel intensities of 1 and 0. The corresponding 4-connected torus is shown in Figures 3.14 and 3.15.

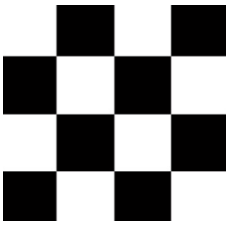


Figure 3.7: Simulated Grid Image `Checker` for $N = 3600$

The grid image of `Squares` is that of concentric squares. A 60×60 lattice grid of squares whose centers coincide and the corresponding sides are parallel to each other. The innermost square of pixel intensity value 1 is inscribed inside alternating circumscribing squares of pixel intensity values 0 and 1.

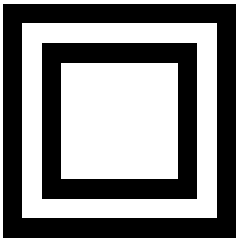


Figure 3.8: Simulated Grid Image `Squares` for $N = 3600$

The grid image of `Circles` is the most smooth mesh with concentric circles. It is a 60×60 grid. In our experiments, we vary the number of circles through $3, 6$ and 7. We construct both torus and non-torus graphs from the image.

3.10.2 Graphs from Simulated Data

In the Figure 3.10, we observe the visualization of the graph as generated from the grid image `Stripes`, using Gephi (Bastian et al., 2009). The graph generation process

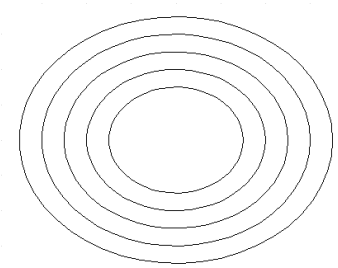


Figure 3.9: Simulated Grid Image `Circles` for $N = 3600$

is the same as before and uses the parameter setting as $K = 4, L = 250, N = 3600$. As in the case of any torus graph that we study here, all the vertices are 4-connected. What is interesting to observe from the visualization is the particular twists in the torus. These turns correspond to the stripes or collection of consecutive columns within the grid image, that have similar pixel intensities. The twists are due to the stripes in the image being wrapped around the boundaries from left to right and top to bottom.

In the Figure 3.11, we see the alternative visualization of the graph generated from the simulated grid image `Stripes`, using GraphViz (Gansner and North, 2000). This particular instance of the graph is from the following parameter setting $K = 4, N = 3600, L = 250$. Once again, this is a 4-connected torus graph with every vertex having four neighbours. This original graph visualization provides insight into the process of collapse into the corresponding PQ graph in 3.12.

Figure 3.12 is the resultant PQ graph corresponding to the original graph for `Stripes`. The PQ graph is the result of running the Algorithm 3.1 on the original graph. From the PQ graph, we observe the vertex at the top as the source and the vertex at the bottom as the target. Every other vertex is a super-vertex. The label on the super-vertices denote the number of vertices that have collapsed onto the super-vertex. The directed edges induce a partial ordering on the graph. The edges are multi-edges and the label on the edges denote the number of edges incident on the pair of super-vertices at its endpoints or the total value of the flow. Figure 3.12 allows for the visualization of the hyper-graph nature of the PQ graph. It is interesting to see that the value of the flow incoming to a super-vertex is equal to the value of the flow exiting the super-vertex. The total number of super-vertices in this particular instance of PQ graph is 130, collapsed from the original 3600 vertices.

Figure 3.14 is the torus visualization of the original graph generated from `Checker`. As the boundaries are wrapped around from top to bottom and left to right, the twists are generated as shown. Every vertex is 4-connected.

Figure 3.15 is the alternative visualization of the same original graph from `Checker`. The parameter settings for this graph are $K = 4, N = 1600, L = 250$. The visualization uses GraphViz to show the lattice grid approximation to the torus.

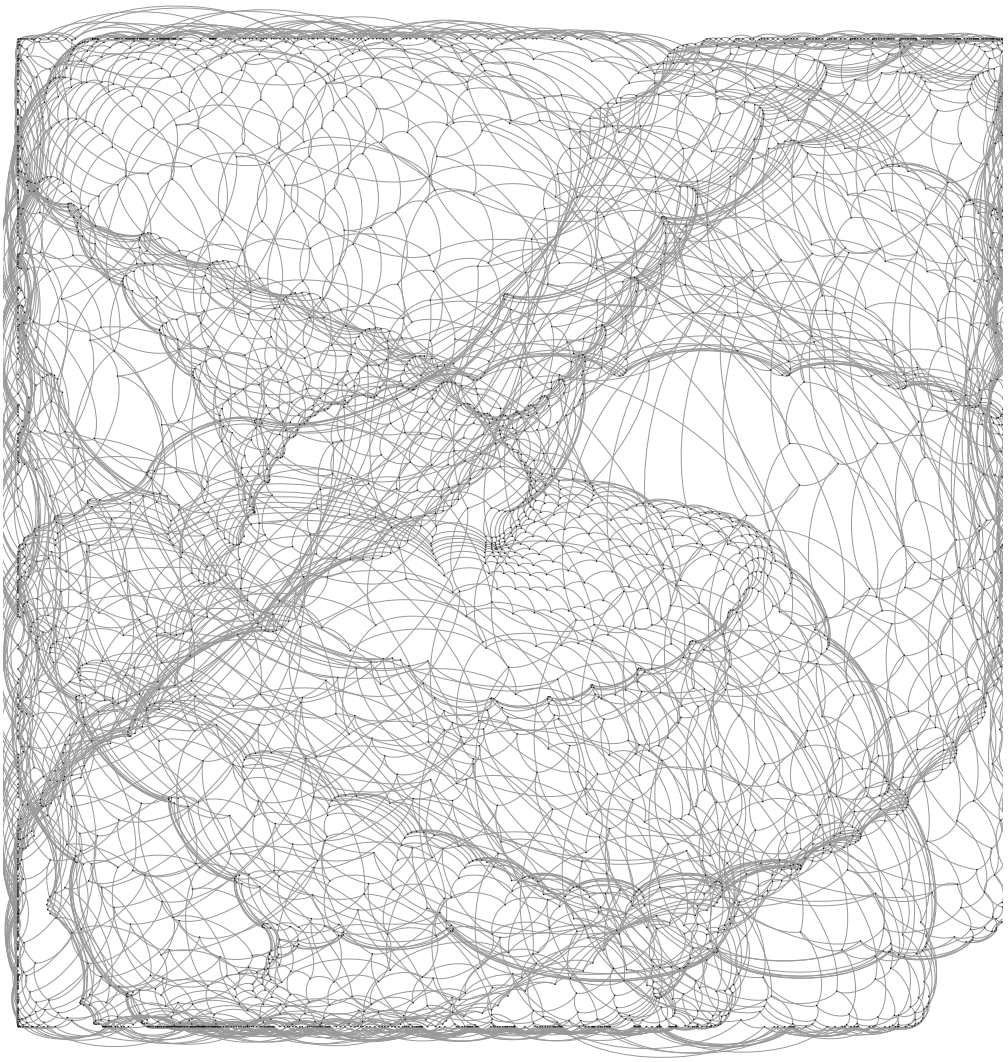


Figure 3.10: Graph of torus `Stripes` with $K = 4$, $N = 3600, L = 250$. Graph visualization on Gephi using Force Atlas 2 algorithm with no overlap, linlog mode and tolerance 0.7.

Figure 3.16 below shows the collapsed PQ graph as observed from running the PQ algorithm on the original graph generated from `Checker` in Figures 3.14 and 3.15. The two central vertices are the source and the target, while the rest are super-vertices. There are a total of 168 super-vertices in the PQ graph.

Figure 3.17 is the original graph constructed from the simulated grid image `Squares` for the parameter settings of $K = 4, N = 3600, L = 250$. The graph is a torus with all four boundary edges wrapped around in order to generate a 4-connected graph.

In Figure 3.18, we observe the PQ graph generated from the original graph of `Squares` as shown in Figure 3.17. The total number of super-vertices in the collapsed PQ graph is 205.

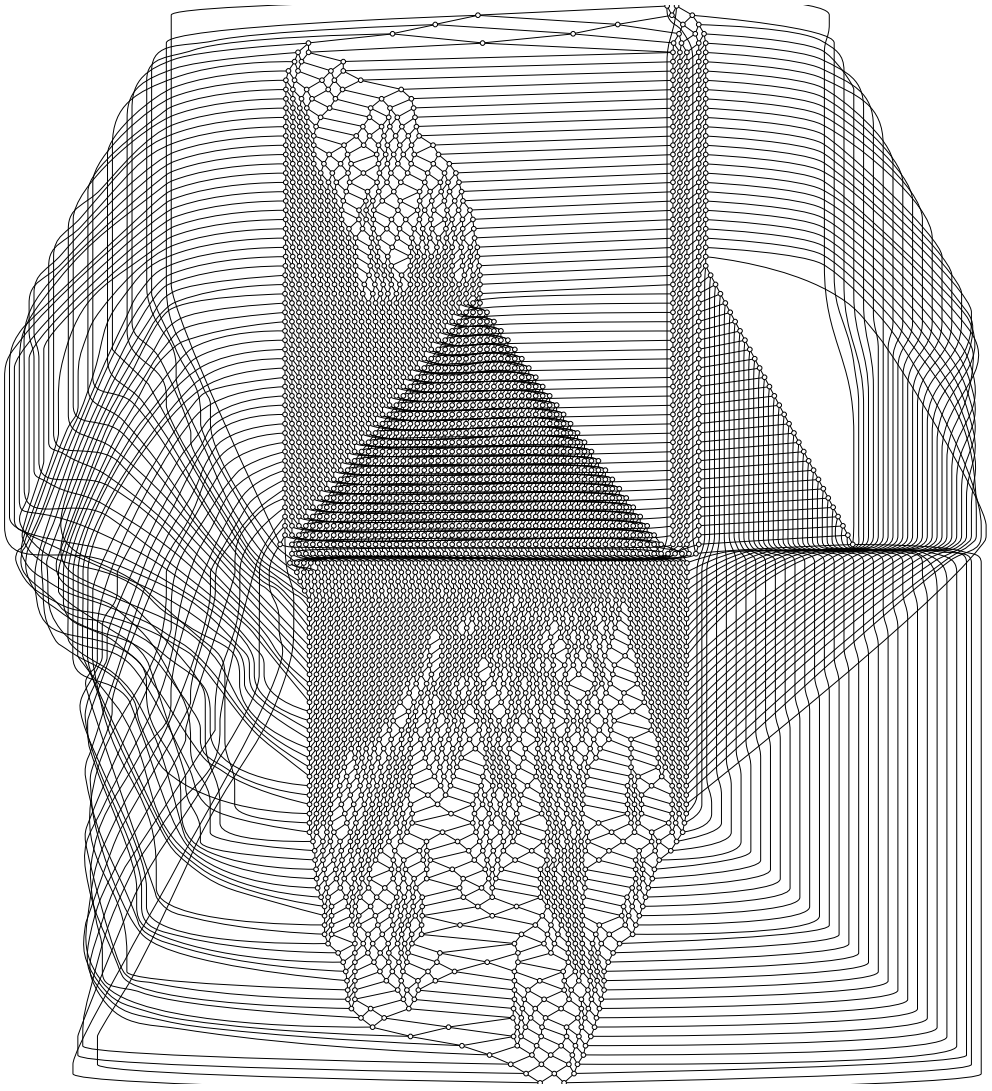


Figure 3.11: Graph of grid image `Stripes` with $K = 4$, $N = 3600, L = 250$. Graph visualization as on GraphViz using .dot specification.

Figure 3.19 is the PQ graph as collapsed from the synthetic data of `Circles`. Unlike in other simulated graphs, the graph of `Circles` from which this PQ graph is derived, is not a torus. The corner vertices thus have two neighbours, the boundary edges vertices have three neighbours and the intermediate vertices have four neighbours. The PQ graph shown here has 126 super-vertices.

3.10.3 Simulated Data Robustness Experiments

The set of experiments in this section is a series of robustness, consistency and scalability tests of `longest-path`, pitted against the standard algorithms. The experiments allow us to have a clear understanding as to where the strengths and weaknesses of our method lie. In all the experiments that follow in this section, we report the mean prediction accuracy of the algorithms calculated over ten trials.

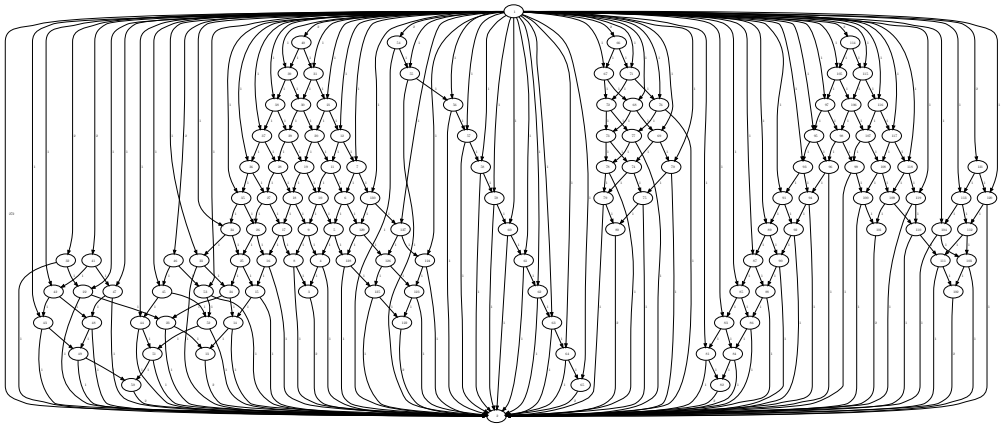


Figure 3.12: PQ Graph generated from the grid image graph of `Stripes` with $K = 4$, $N = 3600, L = 250$ with size 130. Graph visualization as on GraphViz using the .dot graph specification.

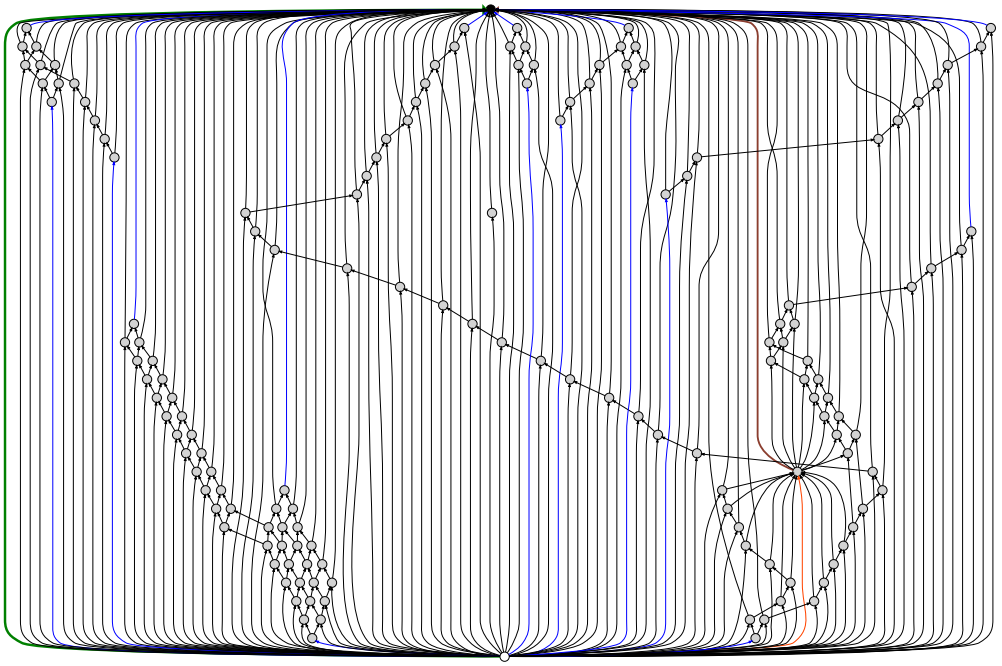


Figure 3.13: PQ Graph generated from the grid image graph of `Stripes` with $K = 4$, $N = 3600, L = 250$ with size 152. Graph visualization as on GraphViz using the .dot graph specification.

In Figure 3.20, we show the result of the experiment of varying the difficulty of the synthetic problem `Circles`. In the experiment, we evaluate the performance of our algorithm `longest-path` with the standard algorithms. We increase the problem size of the toroidal graph structure from 3-circles to 7-circles. 7-circles has 7 concentric rings of 0 intensity pixels interspersed with 1 intensity pixels. The graph construction and label sampling procedure remains the same. However, in this experiment we do not allow for balanced labels, thus having 66% from one class and 34% from the other class. The total number of available labels is $L = 1024$. Our observation is, on the 3-circles, although

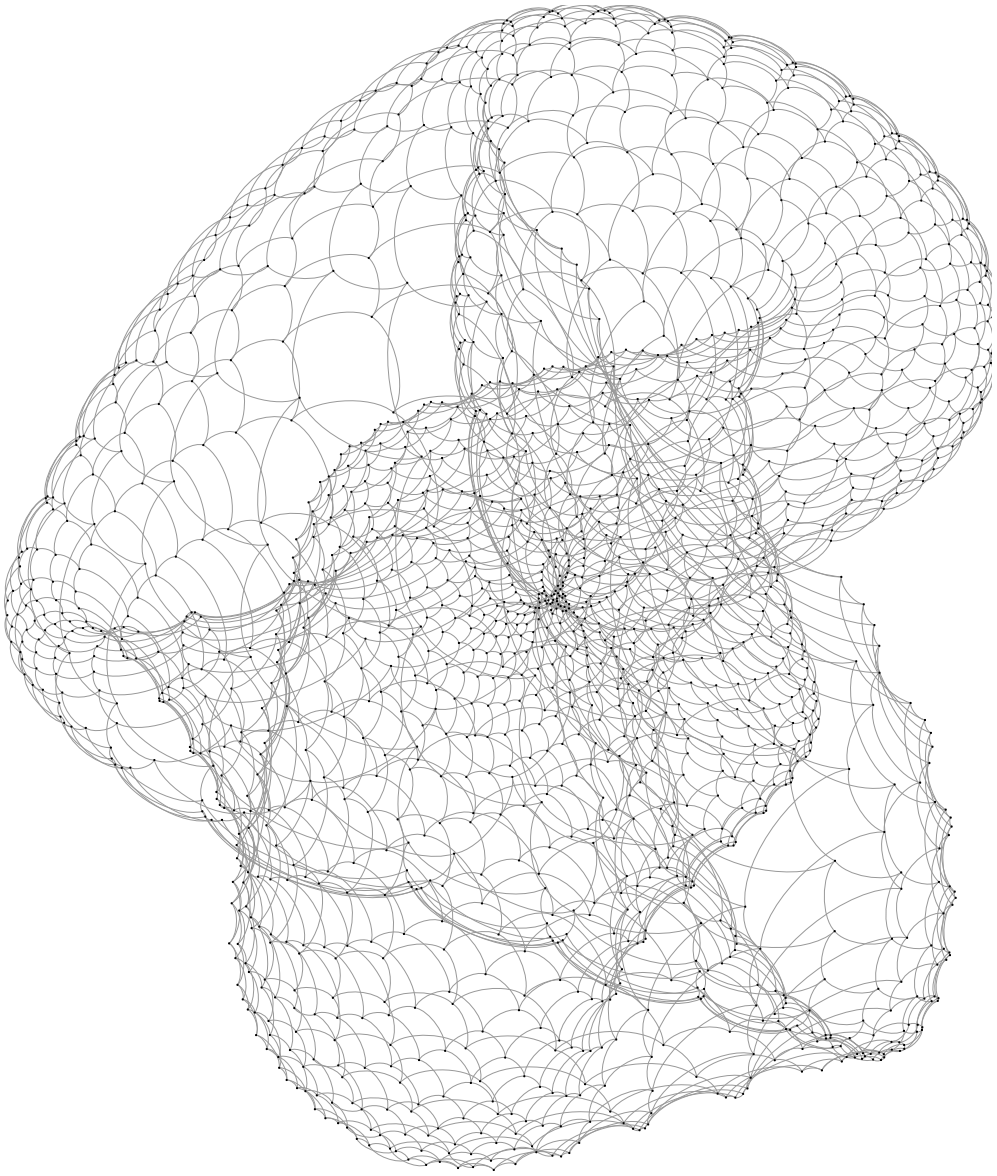


Figure 3.14: Torus graph from `Checker` with $K = 4$, $N = 1600, L = 250$. Graph visualization as on Gephi using Force Atlas 2 algorithm with no overlap, linlog mode and tolerance 0.7.

`longest-path` outperforms `p2`; on the much harder problem of 7-circles, `p2` does much better than `longest-path`. On increasing the level of difficulty in the problem, the smoothness of the model increases, `p2` is designed for taking advantage of the increased connectivity and the dense structure of the graph compared to `longest-path`.

The experiments shown in Figure 3.22 do a performance comparison of the candidate algorithms with varying balance in the class distribution of available labels. Although the graph is the same in both cases: 3-circles. The labels are sampled equally from either class in Figure 3.22 (a), whereas in Figure 3.22 (b), the class distribution is 3:1. The total number of available labels is $L = 1024$. In the balanced class distribution setting, `p2` is the clear winner among the three algorithms. We observe that all methods uniformly

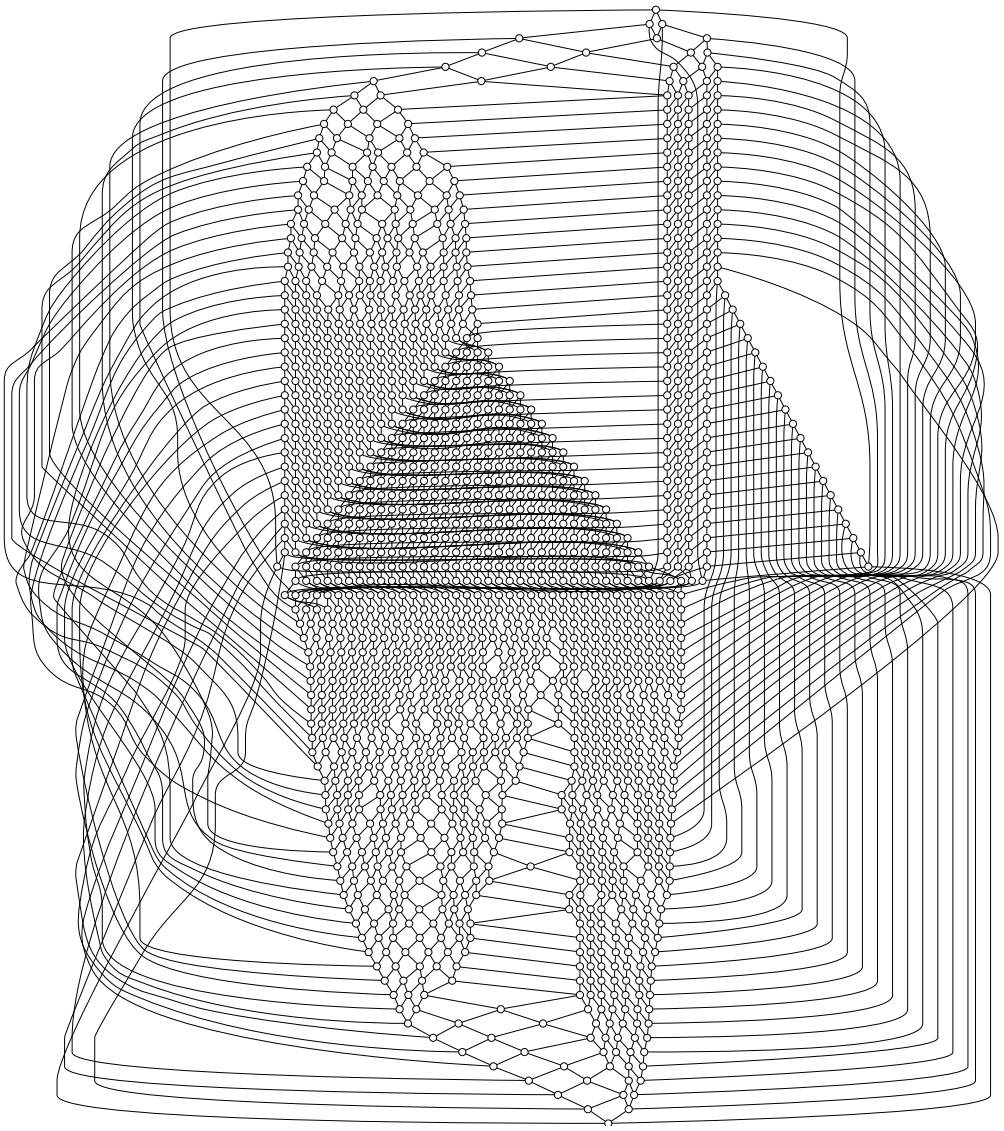


Figure 3.15: Graph of grid image `Checker` with $K = 4$, $N = 1600, L = 250$. Graph visualization as on GraphViz using the .dot graph specification.

perform better in the biased class distribution experiment with `longest-path` surpassing the others. In the unbalanced case, due to the bias, majority voting will always allow the algorithms to perform better.

Figure 3.21 shows the result of one of the vanilla experiments with the synthetic problem `Squares` analogous to `Circles`. `Squares` is an easier problem than `Circles`, where the concentric squares of 0 intensity are less smooth than in the `Circles`. We have a 60×60 grid with 2 concentric squares of 0 intensity interspersed with squares of 1 intensity. The graph construction process is similar with a toroidal structure. In this particular experiment, the available labels are varied in the range $L = 250, 450, 650, 850, 1050$ with balanced class distribution. We observe `longest-path` is able to outperform `p2` at about 18% of the labels available. Even in a non-smooth concentric layout like `Squares`, the region of similarly coloured pixels is inscribed throughout by regions of the opposite

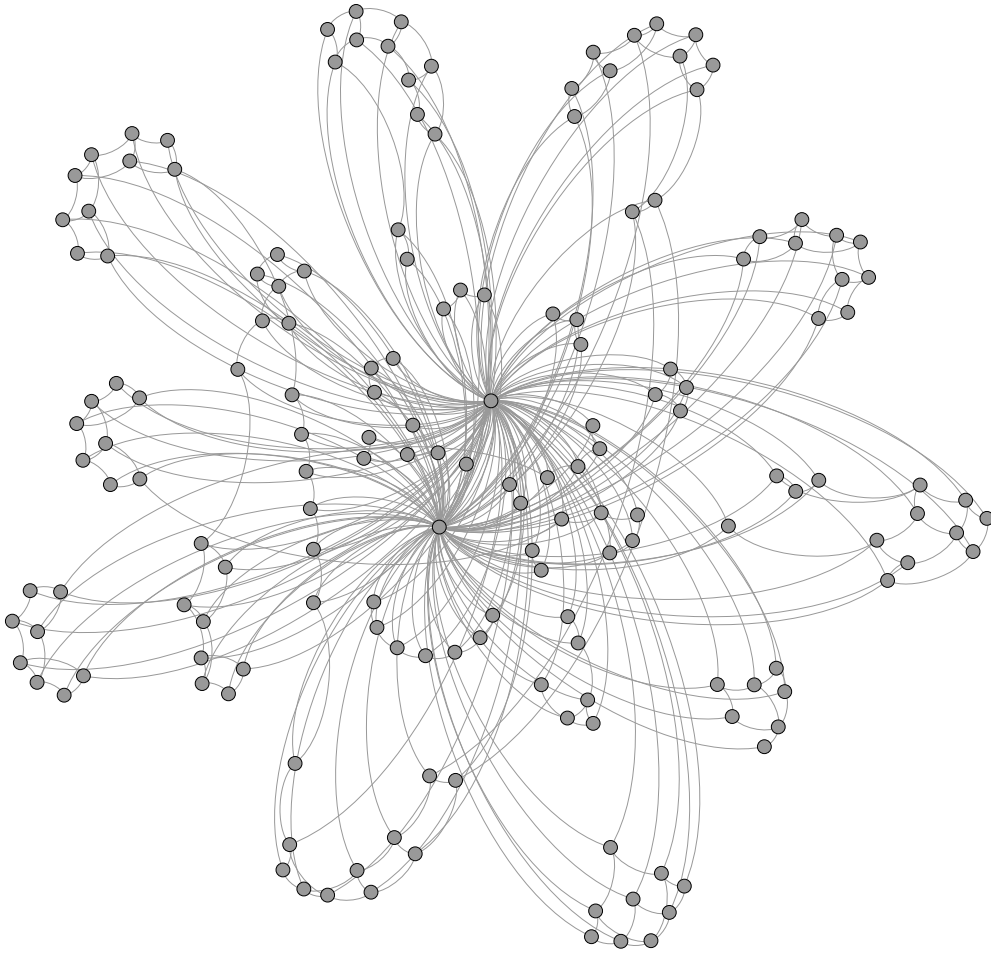


Figure 3.16: PQ graph of `Checker` with $K = 4$, $N = 1600, L = 250$ with 168 super-vertices. Graph visualization as on Gephi using Force Atlas 2 algorithm with no overlap, linlog mode and tolerance 0.7.

colour pixels; leading to dense edges within and across clusters in the corresponding graph. `longest-path` is more applicable when densely clustered regions have sparse connections across the clusters. The algorithm `mindestedge` is one of our sanity test algorithms that we shall discuss more in the Future Works section in Chapter 7.

In Figure 3.23, we compare the torus and non-torus simulated graph problem for the `Checkers` problem. A much harder problem than `Squares`, `Checkers` has interspersed regions of 0 intensity and 1 intensity, as in a checker-board. The difference in the graph construction process is that in the toroid graph, every pixel (vertex) is connected to 4 neighbours. In the non-toroidal graph, the number of neighbours vary between $2, 3$ and 4, with the corner pixels having 2 neighbours, edge pixels having 3 neighbours and the remaining having 4 neighbours. The performance measurement shows that `p2` surpasses `longest-path` in both the settings. The labels are varied through the range $L = 250, 450, 650, 850, 1050$.

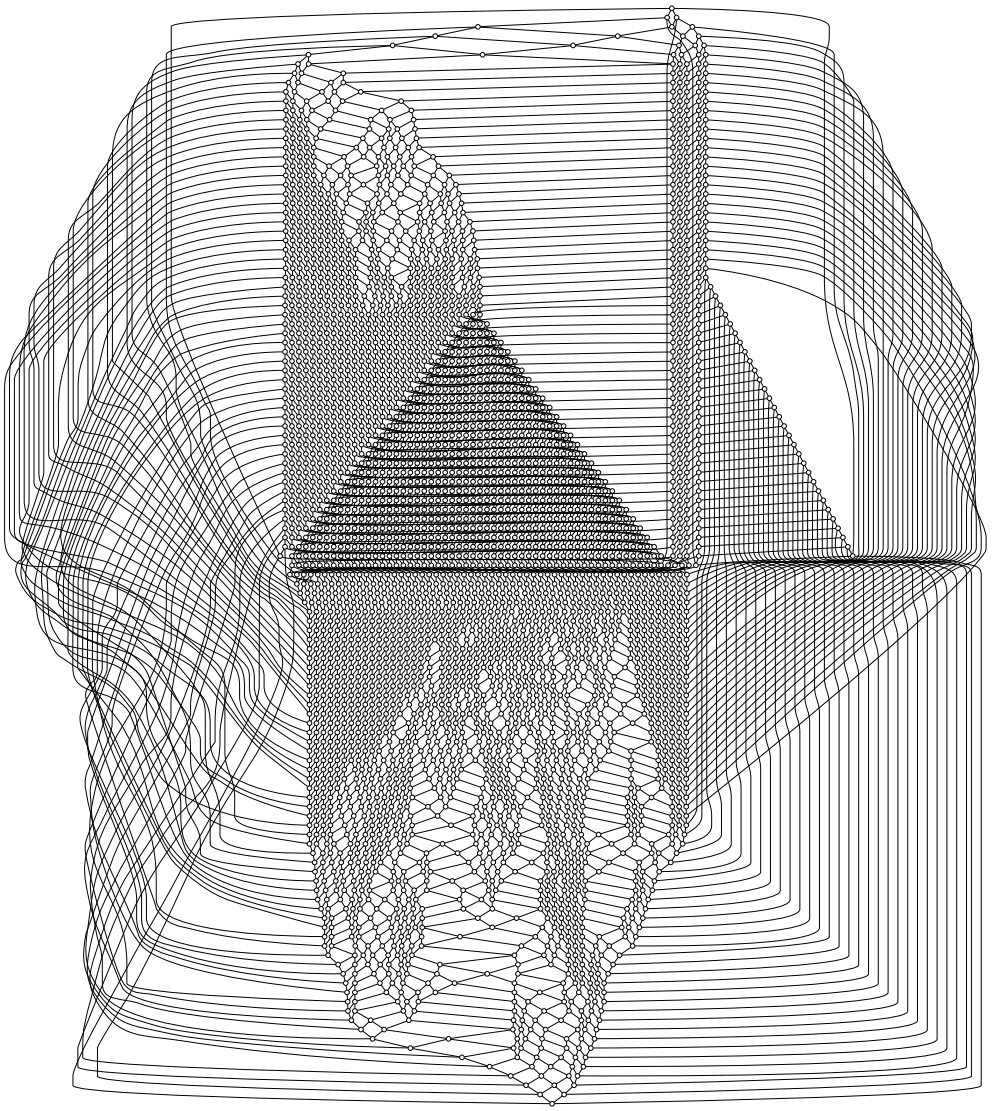


Figure 3.17: Graph of grid image data `Squares` with $K = 4$, $N = 3600, L = 250$. Graph visualization as on GraphViz using the .dot graph specification.

3.10.4 Dataset Robustness Experiments

In Figure 3.24, we show the results of the experiment on the first partition of the dataset `Isolet` that is `Isolet 1`, with varying the connectivity. `Isolet 1` has 1560 instances with 617 features. The classification task is the same as classifying letters 'A'-'M' versus 'N'-'Z'. The instances that are randomly sampled to be included in the graph ensure a balanced class distribution of 780:780. Specifically, we vary the parameter for connectivity between $K = 3$ and $K = 4$. To ensure the graph constructed is connected, we have always maintained the Minimum Spanning Tree (MST) edges in the graph. However, in this experiment we test without the MST edges for $K = 4$. Our results in both (a) and (b) show that although `p2` leads in the initial settings, `longest-path` outperforms `p2` when more than 33% of the labels are available, as in the standard case on this dataset. As `Isolet` is a noisy dataset, the `longest-path` algorithm performs

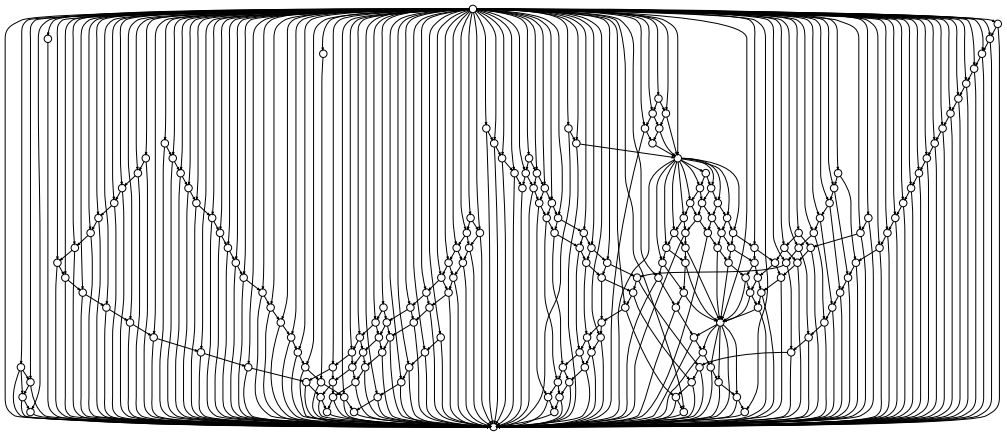


Figure 3.18: PQ graph of grid graph `Squares` with $K = 4$, $N = 3600, L = 250$ with 205 super-vertices. Graph visualization as on GraphViz using the .dot graph specification.

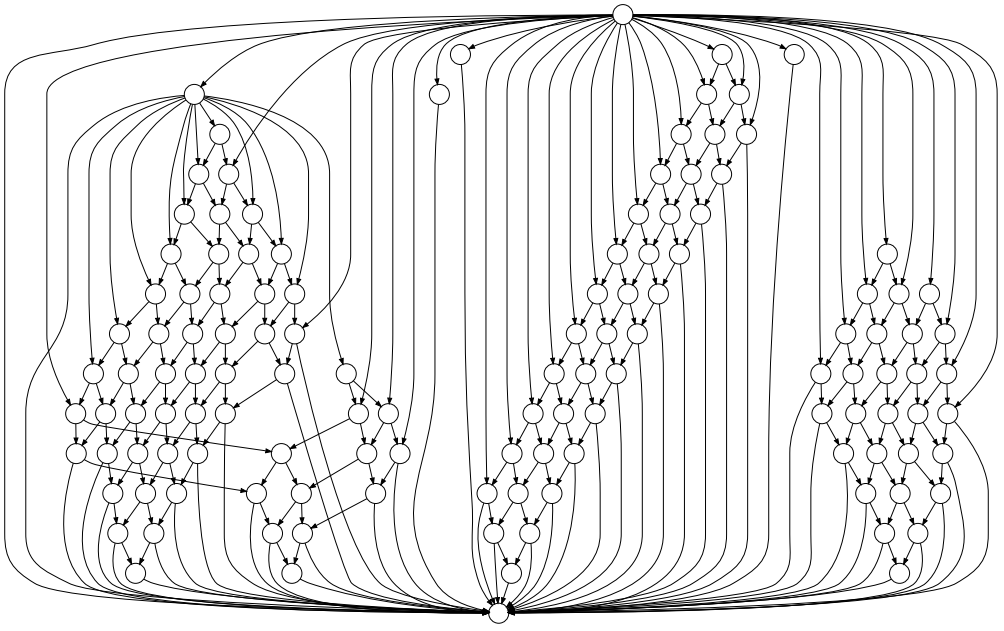


Figure 3.19: PQ graph for non-torus graph from `Circles` with $K = 2, 3, 4$, $N = 3600, L = 128$ with 126 super-vertices. Graph visualization as on GraphViz using the .dot graph specification.

better as it is more robust to noise. Being robust to noise allows the algorithm to function well even with the presence or absence of MST edges.

Figure 3.25 shows the observations on two sets of experiments on the `20 newsgroups` dataset. Figure 3.25 (a), classifies Group '1' against Groups '2', '3' and '4' with unbalanced label sets. Figure 3.25 (b) classifies Groups '1' and '2' against Groups '3' and '4'. The graph construction process is the same as before with this dataset. The cost matrix is computed based on the cosine distance and the MST edges are maintained for ensuring that the graph is connected. The total number of instances included in the graph is 16242. The distribution of the instances into groups is as follows: Group '1':

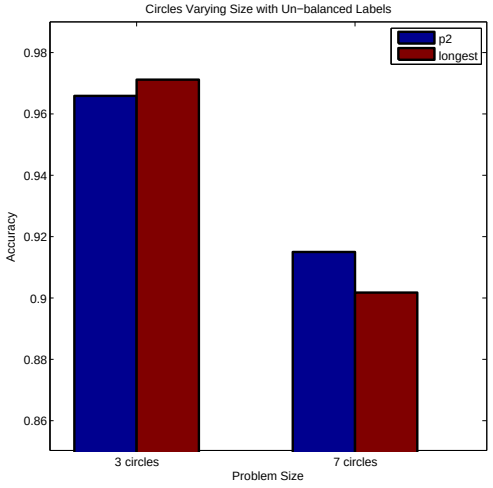


Figure 3.20: `Circles` of varying sizes 3 and 7 with $K = 4$, $N = 3600, L = 1024$.

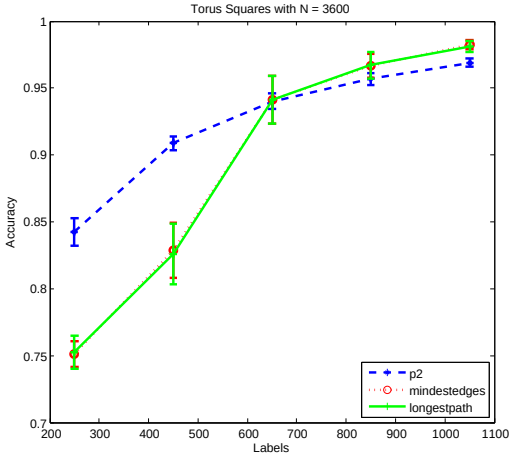
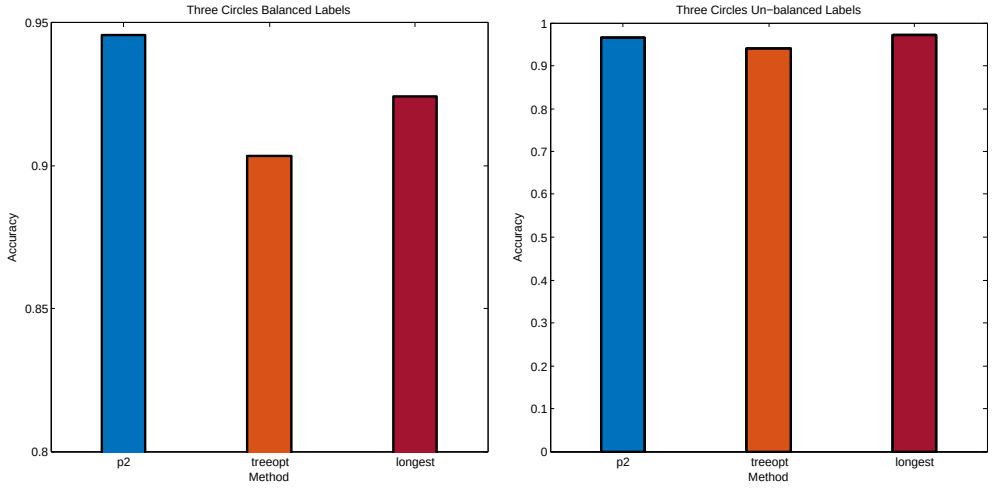


Figure 3.21: Toroidal `Squares` with $L = 250, N = 3600$.

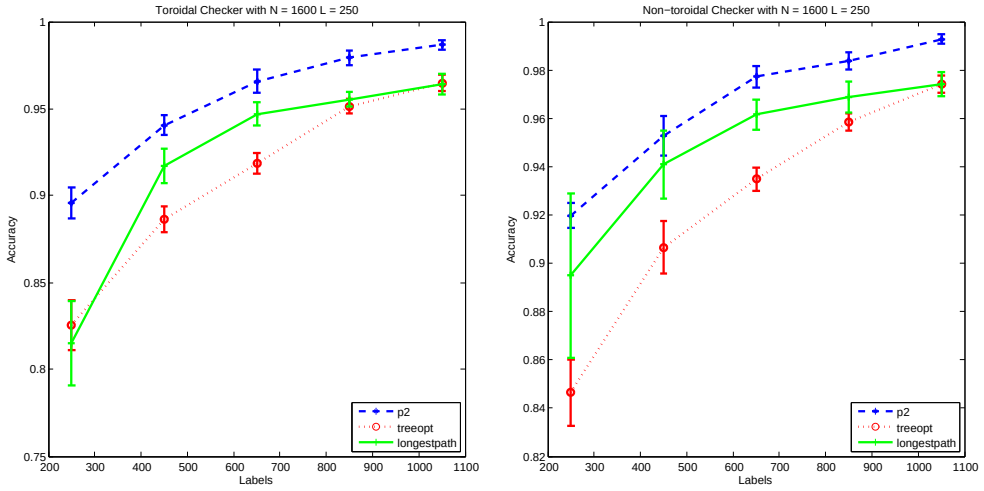
4605, Group '2': 3519, Group '3': 2657 and Group '4': 5461. Total number of examples in Groups '2', '3' and '4' combined are 11637. In (a), the labels are sampled equally from the two classes distributed as 4605:11637 where `longest-path` is the clear winner while in (b) `longest-path` eventually catches up with `p2`. `20 newsgroups` is an extremely sparse dataset. Although, in the case of unbalanced labels, majority vote situation can aid `longest-path`, it is interesting to observe that in the perfectly balanced scenario, `longest-path` is still robust.

Figure 3.26 is the experiment with `WebSpam` for different techniques of sampling the labels and evaluating the algorithms. In both settings, the available label sets are balanced (up to 400 labels in (a)). In (a), we sample the labels from the training set that has 907 instances with a class distribution of 206:701. The evaluation is on the test set of 8165 instances. In (b), we randomly sample from the training set with 907 instances and validation set with 1800 instances with a total of 2797 instances; the class distribution



(a) (b)

Figure 3.22: Balanced Versus unbalanced with labels $L = 1024$ in `Circles` (a) Balanced labels (b) Unbalanced labels.

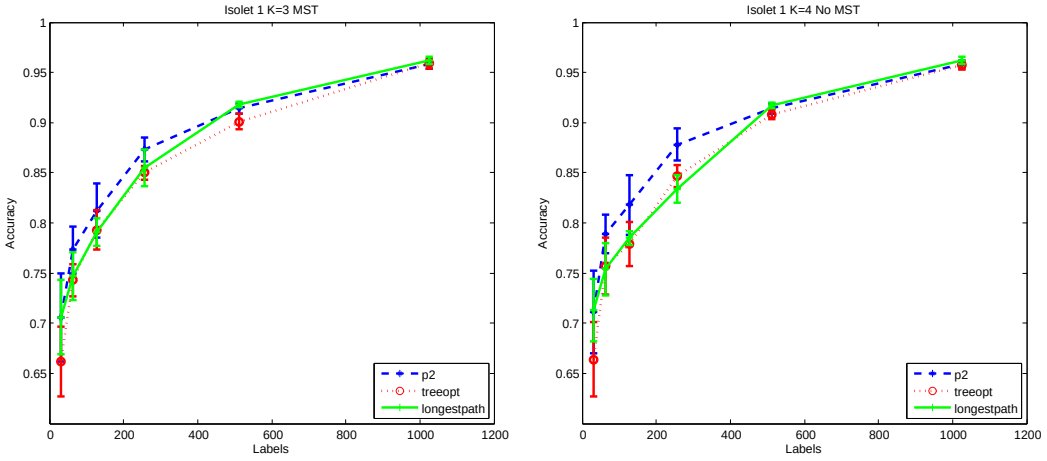


(a) (b)

Figure 3.23: (a) Toroidal `Checker` plot with $N = 1600$ (b) Non Torus `Checker` plot with $N = 1600$.

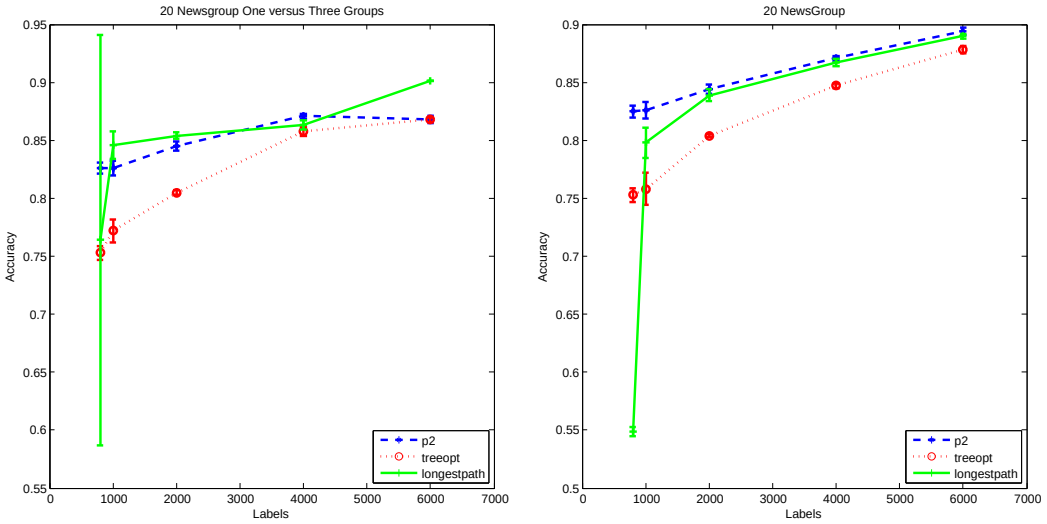
is 594:2113. The evaluation is carried out on the training and validation set over a total of 2707 instances. `Webspam` is a highly sparse dataset, hence `p2` struggles on this dataset. `Treeopt` copes well with the aggregating predictions via majority vote by using a committee of spanning trees. `longest-path` is robust on `Webspam` irrespective of the type of dataset partition used for sampling the labels.

Figure 3.27 describes the results of the experiment on the dataset `Isolet`. The method of graph generation is the same in both cases. The classification task is the same as before with a class distribution of 3900:3897. In (a), the labels are sampled from all the partitions of the dataset `Isolet`-1 through to `Isolet`-5 with balanced class distribution



(a) (b)

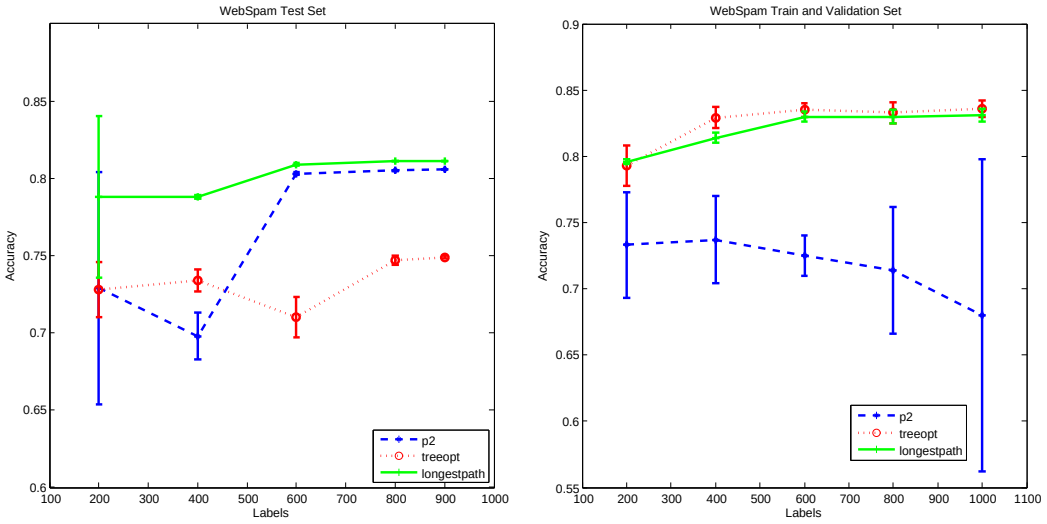
Figure 3.24: Plot for dataset `Isolet` on (a) Isolet 1 with $K = 3$ MST (b) Isolet 1 with $K = 4$ no MST.



(a) (b)

Figure 3.25: Plot for dataset `20 newsgroups` (a) Group '1' vs '2', '3' and '4' groups (b) Groups '1' and '2' Vs. Groups '3' and '4'.

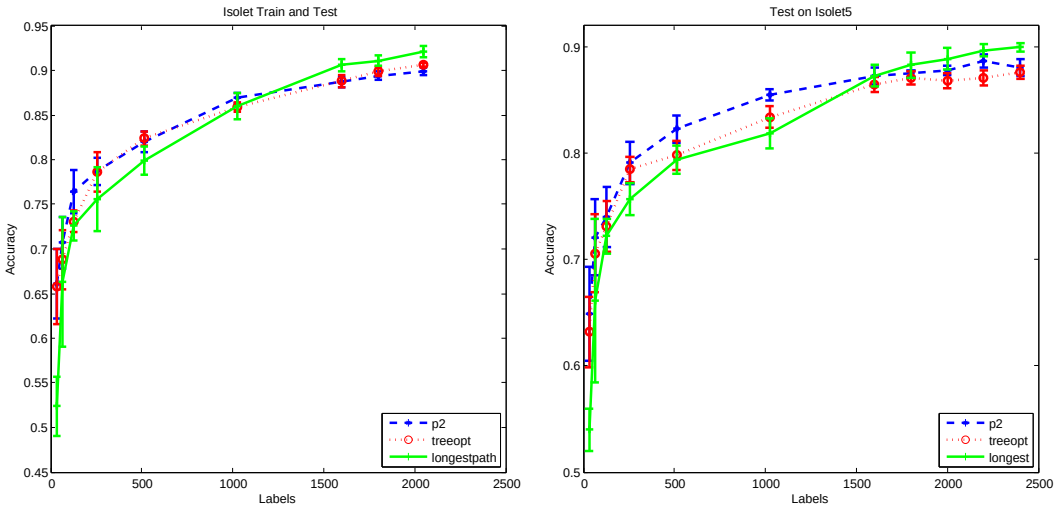
of 3900:3897. There are a total of 7797 instances randomly sampled for the graph. The performance is measured on all of the data in `Isolet`-1 through to `Isolet`-5. In (b), the labels are sampled from the training set only. The training set in `Isolet` constitutes the data in `Isolet 1` through `Isolet 4` with a total of 6238 labels with a class distribution of 3118:3120. The performance is measured on the test set only, which is `Isolet 5` having 1559 unlabelled instances. The observation shows that when the available labels are sampled from the entire dataset, the performance of all the algorithms become uniformly better as seen in (a). When the available labels are only sampled from the training set, label propagation is not very successful for the part



(a) (b)

Figure 3.26: Plot for dataset `Webspam` under varying partitions of the data (a) Labels from training set, evaluation on test set (b) Labels from training and validation sets, evaluation on training and validation sets.

of the dataset under evaluation. `longest-path` though initially under-performs, it is comparable and eventually outperforms both `treeopt` and `p2`. It is robust to changes in the membership of the labels sampled.



(a) (b)

Figure 3.27: Plot for dataset `Isolet` (a) Evaluation on training and test data `Isolet1` through `Isolet5` (b) Performance measured on test data `Isolet5`.

Figure 3.28 describes the experiment with the dataset `Isolet` on its first partition `Isolet 1` having a connectivity $K = 4$. The labels are varied through the range of L=32, 64, 128, 256, 512, 1024. The labels are balanced in terms of class distribution.

The available labels are randomly sampled from the 1560 labelled instances in `Isolet 1` and the performance is measured over all of the unlabelled instances. `treeopt` is competitive with `longest-path` and `p2` here, although `longest-path` leads with labels more than $L = 512$. `Isolet` is a noisy dataset and `longest-path` is robust on this dataset.

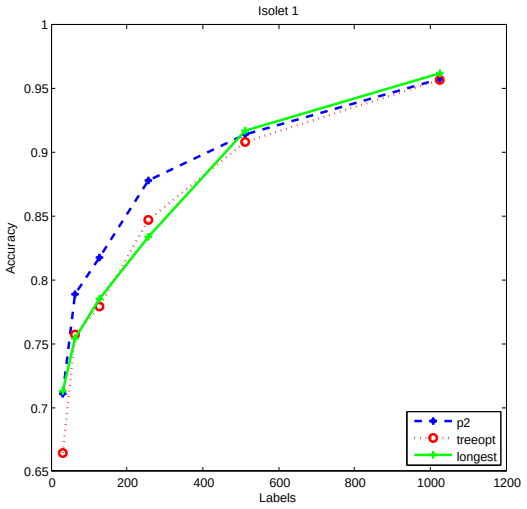


Figure 3.28: Plot for dataset `Isolet1` over $N = 1560$ instances, $K = 4$.

The experiment in Figure 3.29 is a comparison between `treeopt` and `longest-path` over a large number of available labels from the `WebSpam` dataset. The labels are varied in the range L=200, 400, 600, 800, 1000. The evaluation is carried out over the entire dataset. Being the most sparse dataset, `treeopt` beats `longest-path` throughout as it carefully exploits the majority vote from the committee of spanning trees while making the prediction.

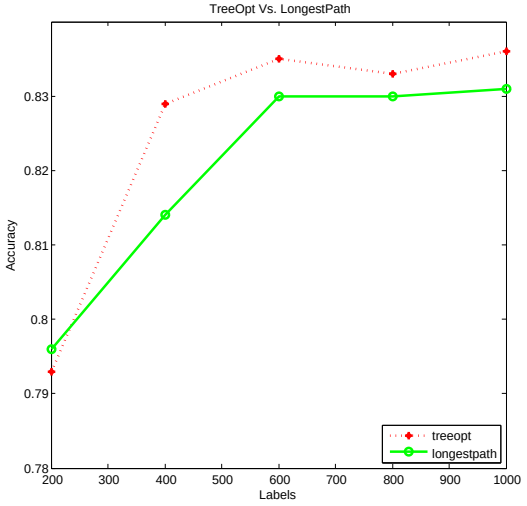


Figure 3.29: Plot for `WebSpam` over large number of available labels.

In the Figure 3.30, we use the `USPS` dataset for testing the robustness of the algorithms with varying sparsity. The number of available labels for this experiment is fixed to

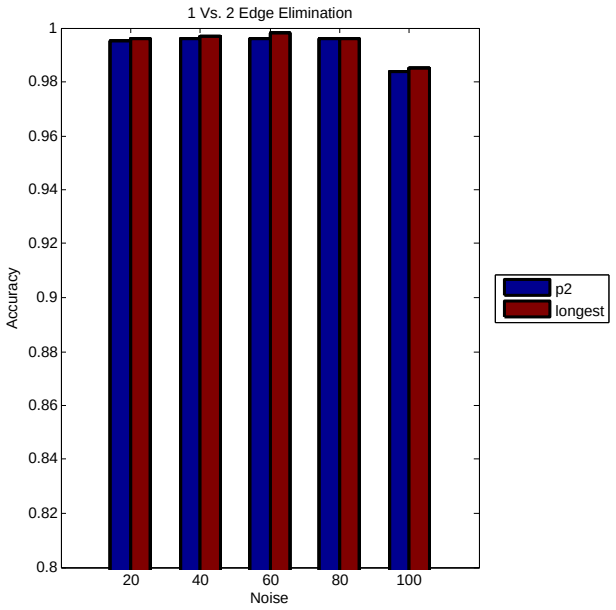
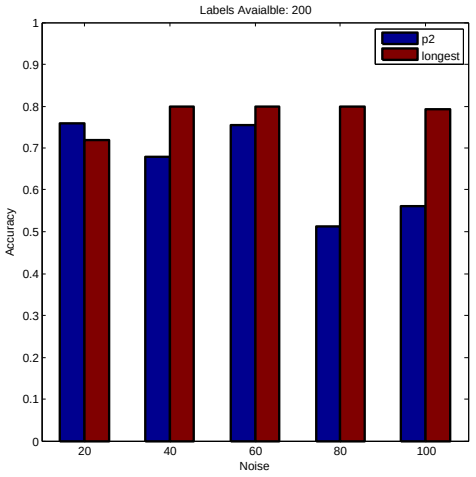


Figure 3.30: `USPS` 1 Vs. 2 Edge Elimination.

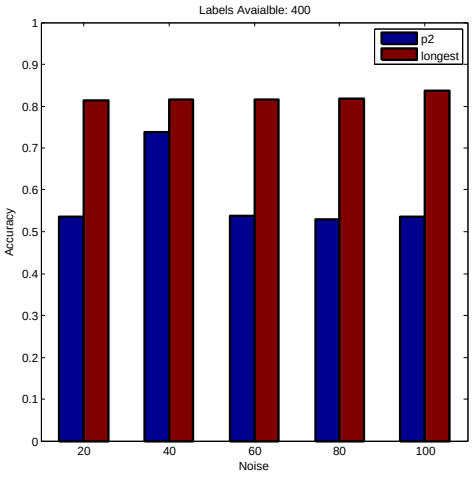
32. We introduce a parameter 'noise' that varies over the range $20, 40, 60, 80, 100$ and a parameter 'bias' expressed as a percentage of noise. The noise is introduced at the graph construction phase. For every edge in the graph constructed from the dataset, a coin is flipped and if the value drawn is greater than the bias, the edge is included, else the edge is eliminated. With a noise level of 100%, all the edges are eliminated except the minimum spanning tree edges that are maintained for having a connected graph. Here, the noise parameter serves to control the sparsity in the graph. `p2` is known to suffer with sparse graphs with fewer edges whereas `longest-path` is quite robust to sparsity and up-to a sufficient level of sparsity in the graph, performs very well; the performance falls only when the minimum number of MST edges at 100 noise level. As edges are being eliminated, connections between the vertices are lost and hence `p2` finds it difficult to propagate the labels.

In Figure 3.31, the similar experiment with `WebSpam` is carried out where the algorithms are tested for robustness with varying sparsity through the means of balanced edge elimination. This is achieved through the addition of the same number of random edges to the graph upon deletion of edges based on the level of bias. For every edge in the graph constructed from the dataset, a coin is flipped and if the value drawn is greater than the bias, the edge is included, else the edge eliminated and a new random edge is added to the graph. Here, the labels are varied in the range L = 200, 400, 600, 800 and 1000. We observe that `longest-path` is extremely robust to noise in comparison to `p2`. The new random edges do not contribute to the `longest-path` prediction adversely as in the case of `p2`.

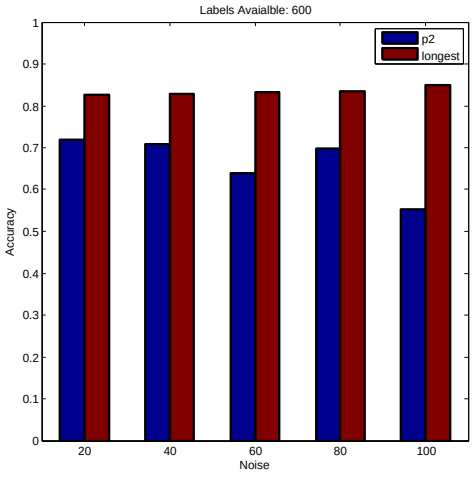
The Table 3.3, summarizes all the `USPS` results under one table. Our implementation of `majvote` is used here as a sanity test to validate `longest-path` prediction. `majvote`



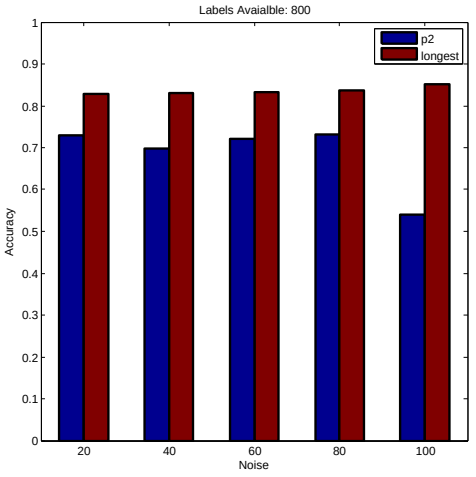
(a)



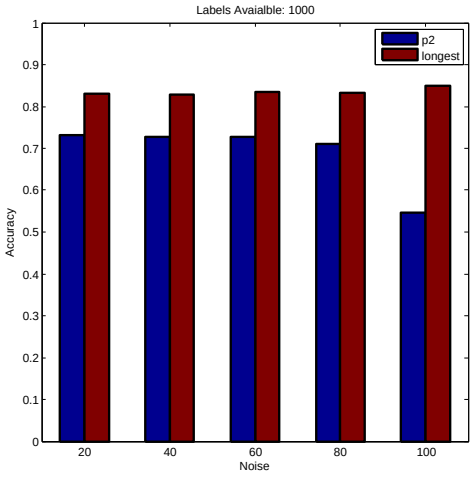
(b)



(c)



(d)



(e)

Figure 3.31: Plots for Balanced Edge Elimination `Webspam` (a) L=200 (b) L=400 (c) L=600 (d) L=800 (e) L=1000

Table 3.3: Performance over N,L,K for `USPS` for 0-temp Ising model

		K = 3				
		N = 1000				
		L = 8	**L = 16**	**L = 32**	**L = 64**	**L = 128**
2 vs 3	**p2**	0.98(0.01)	0.982(0.008)	0.984(0.005)	0.988(0.003)	0.991(0.002)
	treeopt	0.814(0.055)	0.885(0.032)	0.891(0.032)	0.956(0.0133)	0.959(0.01326)
	longpath	0.504(0.001)	0.94(0.143)	0.987(0.003)	0.988(0.003)	0.99(0.002)
	majvote	0.504(0.001)	0.939(0.144)	0.987(0.002)	0.988(0.004)	0.99(0.002)
3 vs 8	**p2**	0.956(0.009)	0.953(0.007)	0.961(0.007)	0.967(0.004)	0.971(0.004)
	treeopt	0.797(0.105)	0.749(0.095)	0.878(0.013)	0.935(0.026)	0.96(0.023)
	longpath	0.505(0.001)	0.6(0.184)	0.969(0.006)	0.971(0.004)	0.972(0.003)
	majvote	0.505(0.001)	0.599(0.182)	0.97(0.006)	0.971(0.004)	0.972(0.003)
4 vs 7	**p2**	0.985(0.006)	0.986(0.003)	0.987(0.003)	0.986(0.003)	0.989(0.003)
	treeopt	0.84(0.115)	0.884(0.064)	0.956(0.027)	0.968(0.013)	0.956(0.024)
	longpath	0.648(0.22)	0.986(0.002)	0.987(0.004)	0.986(0.003)	0.988(0.004)
	majvote	0.648(0.22)	0.986(0.002)	0.987(0.003)	0.985(0.004)	0.988(0.004)
6 vs 9	**p2**	0.998(0.001)	0.999(0.001)	0.998(0.001)	0.998(0.001)	0.999(0.001)
	treeopt	0.945(0.045)	0.955(0.032)	0.992(0.023)	0.994(0.034)	0.994(0.041)
	longpath	0.999(0.001)	0.999(0.002)	0.998(0.002)	0.998(0.002)	0.999(0.001)
	majvote	0.999(0.001)	0.9986(0.001)	0.9983(0.001)	0.9983(0.001)	0.9989(0.0009)

is the approximated Ising prediction that takes a majority vote from the neighbours of the vertex in question for prediction. In principle, `majovote` prediction should be identical to `longest-path`. As before, all the algorithms behave uniformly better with more number of labels.

Table 3.4, summarizes the results of performing random spanning tree (RST) graph generation. The graph comprises the RST edges sampled during the graph generation. The number of RSTs sampled varies through $1, 2, 4, 8$. Although, $K = 9$ is not the ideal connectivity for the graph, here we use a value of 9 to test the robustness of the algorithm. The intuition behind using the randomized RST edges to approximate the graph connectivity is to allow `treeopt` to function at its best. As discussed before `treeopt` can find the cluster structure in RST approximated graphs in expectation. We also evaluate the robustness of `longest-path` in this setting. It appears that losing the graph connectivity (cluster) structure due to the approximation does affect `longest-path` at low labels although it catches up to `treeopt` when the labels increase. The randomization affects all the algorithms; although `longest-path` gets most affected. `treeopt` is competitive with `p2` throughout. `p2` is the clear winner and uniformly performs better with the increase in randomization. This shows the sensitivity of `longest-path` to randomization in the cluster structure. Uniformly labelled clusters, is vital for `longest-path` to perform at its best.

3.10.5 Discussion and Conclusion

The performance measure used throughout the empirical evaluation is the generalized prediction accuracy over the graph size. We summarize the observations of the empirical

Table 3.4: 1 Vs. 2 Random Spanning Trees on 0-temp Ising model

		K = 9			
		N = 1000			
		RST = 1	**RST = 2**	**RST = 4**	**RST = 8**
l=2	**p2**	0.834(0.115)	0.877(0.183)	0.8(0.191)	0.849(0.204)
	treeopt	0.781(0.141)	0.715(0.043)	0.926(0.095)	0.77(0.107)
	longpath	0.835(0.115)	0.501(0)	0.501(0)	0.501(0)
l=4	**p2**	0.788(0.101)	0.965(0.047)	0.954(0.064)	0.988(0.013)
	treeopt	0.941(0.024)	0.839(0.161)	0.82(0.116)	0.896(0.117)
	longpath	0.801(0.103)	0.502(0)	0.502(0)	0.502(0)
l=8	**p2**	0.869(0.064)	0.992(0.003)	0.993(0.004)	0.991(0.004)
	treeopt	0.888(0.059)	0.902(0.044)	0.87(0.073)	0.921(0.045)
	longpath	0.872(0.054)	0.602(0.195)	0.504(0)	0.504(0)
l=10	**p2**	0.893(0.077)	0.992(0.003)	0.994(0.002)	0.993(0.002)
	treeopt	0.872(0.09)	0.853(0.06)	0.908(0.055)	0.904(0.045)
	longpath	0.909(0.058)	0.945(0.146)	0.652(0.225)	0.603(0.197)
l=12	**p2**	0.92(0.045)	0.976(0.051)	0.994(0.001)	0.994(0.002)
	treeopt	0.903(0.026)	0.9(0.056)	0.892(0.027)	0.87(0.083)
	longpath	0.896(0.079)	0.895(0.194)	0.946(0.147)	0.947(0.147)
l=14	**p2**	0.932(0.029)	0.991(0.005)	0.994(0.002)	0.993(0.002)
	treeopt	0.921(0.044)	0.9(0.061)	0.95(0.043)	0.946(0.032)
	longpath	0.93(0.035)	0.994(0.003)	0.945(0.146)	0.948(0.147)

results on our extended set of experiments as follows:

Over large and sparse datasets such as `USPS` and `Webspam`, `longest-path` outperforms `labelProp` consistently. The performance of `labelProp` and `longest-path` on the `Isolet` dataset are competitive, where `longest` path beats `labelProp` when available labels are around 30% of the graph size, when the graph is generated from `Isolet 1`. `Treeopt` beats `labelProp` on the `Webspam` dataset even at 2% of training labels. `longest-path` is very competitive with `Treeopt`, although cannot outperform `Treeopt` in this very sparse dataset. Lower connectivity in the graphs aids `longest-path` throughout. RST experiments with RST = 1 on `USPS` show `Treeopt` competitive in comparison to `longest-path` and `labelProp`. In concentric `Squares`, `longest-path` beats `labelProp` at about 18% of the available labels and above. In concentric `Circles`, `labelProp` beats `longest-path` all throughout. However, `longest-path` becomes competitive with `labelProp` as more labels are available. `labelProp` seems to stagnate in its performance towards the end, with slightly increased performance with the newly available labels. Torus graphs seem to give a minor improvement over non torus graphs on the simulated data. `longest-path` cannot beat `labelProp` in the `Circles` simulated graph variants even in the online setting and on smaller size problems.

The `longest-path` strategy is optimal on trees and on generic graphs it improves on the state of the art in many interesting cases. The strategy's worst-case time to sequentially label the entire graph is quadratic. In future work we would like extend our prediction strategy from predicting with the longest path to prediction that takes into account the connectivity. Two possible approaches are predicting with the minimum edge destruction or minimum vertex destruction. This would take into account the connectivity at the

time of prediction, such that if a mistake is made, maximum number of edges or vertices will be collapsed to the resultant PQ graph or in other words, their labels will be deduced. We attempt to conclude this chapter by providing insight into the question : in what sense are the predictions of `longest-path` deterministic approximate predictions of `0-Ising`? Let the vectors $\boldsymbol{y}, \hat{\boldsymbol{y}}^{\texttt{0I}}, \hat{\boldsymbol{y}}^{\texttt{LP}} \in \{0,1\}^n$ denote the true labelling of the graph, the vectors of the sequential predictions of `0-Ising` and `longest-path` and let $B(\mathcal{S})$ denote a mistake bound of `0-Ising` and `longest-path` on an example sequence $\mathcal{S}$ (see Theorems 3.4 and 3.10). Then we have

$$\|\hat{\boldsymbol{y}}^{\texttt{0I}} - \hat{\boldsymbol{y}}^{\texttt{LP}}\|_1 \leq \|\boldsymbol{y} - \hat{\boldsymbol{y}}^{\texttt{0I}}\|_1 + \|\boldsymbol{y} - \hat{\boldsymbol{y}}^{\texttt{LP}}\|_1 \leq 2B(\mathcal{S})$$

thus the sequence of predictions $\hat{\boldsymbol{y}}^{\texttt{LP}}$ from the deterministic approximation `longestPath` are guaranteed to never differ from the sequence of predictions $\hat{\boldsymbol{y}}^{\texttt{0I}}$ of `0-Ising` except in at most $2B(\mathcal{S})$ trials. We would also like to analyse this further as part of future work.

Chapter 4

Online MeanField Approximation for Graph Labelling

4.1 Introduction

In this chapter, we study the similar problem of sequential classification of unlabelled vertices of a graph, from the perspective of variational approximation. Like in the previous chapter, in the limit of zero temperature, predicting the label of a vertex with the maximum marginal probability is NP-hard.

Here, we use meanfield variational approximation to approximate the posterior distribution over all possible labellings by means of a decoupled distribution, and then predict with the maximum marginal value of the approximated distribution, the label of the queried vertex at low temperatures.

4.2 Background

We assume that the underlying average behaviour of the system under observation is known. Suppose there are a multitude of graphs, and for the whole set, there is some average number of cuts observed. A "cut" as we have seen before, is the number of edges with disagreeing labels. Let the cut of any graph in the set of graphs be defined by,

$$\phi_{\mathcal{G}}(\boldsymbol{u}) = \sum_{(i,j)\in E} \left[\!\left[u_i \neq u_j \right]\!\right] \tag{4.1}$$

where, $\boldsymbol{u}$ is the labelling of the graph, E is the number of edges and $[\![]\!]$ predicate denotes an indicator function which is equal to 1 if the predicate is true and 0 otherwise. If the "cut" observed is given by c^*, on average the expected number of cuts should be equal

to the cut observed.

$$\mathbb{E}\left[\phi_{\mathcal{G}}\left(\boldsymbol{u}\right)\right] = c^{*} \tag{4.2}$$

Now, if the probability of a particular labelling of a graph is given by $\mathbb{P}\left(\boldsymbol{u}\right)$, then the goal is to compute the probability of the labelling with the assumption that the average number of cuts c^{*} is observed. $\mathbb{P}\left(\boldsymbol{u}\right)$ is also the probability of having a particular number of cuts. Alternatively, we have from 4.2.

$$\sum_{\boldsymbol{u}} \mathbb{P}\left(\boldsymbol{u}\right) \phi_{\mathcal{G}}\left(\boldsymbol{u}\right) = c^{*} \tag{4.3}$$

Typically, for inferring the probability distribution over all possible labellings $\mathbb{P}\left(\boldsymbol{u}\right)$, one would look at the entropy of the distribution given by,

$$H\left(\boldsymbol{u}\right) = -\sum_{\boldsymbol{u}} \mathbb{P}\left(\boldsymbol{u}\right) \log\left(\mathbb{P}\left(\boldsymbol{u}\right)\right). \tag{4.4}$$

Maximum entropy of the distribution is the measure of the maximum uncertainty in the distribution. Naturally, the maximum uncertainty in the distribution is when all possible labellings are equally likely. In other words, maximum entropy counts the number of all possible ways of having cuts. The average number of cuts end up being the prior. Taking the Lagrangian of 4.4 and incorporating the constraints in 4.3 and $\sum_{\boldsymbol{u}} \mathbb{P}\left(\boldsymbol{u}\right) = 1$, one is interested in maximizing the Lagrangian, as shown.

$$L = -\sum_{\boldsymbol{u}} \mathbb{P}\left(\boldsymbol{u}\right) \log\left(\mathbb{P}\left(\boldsymbol{u}\right)\right) + \beta\left(\sum_{\boldsymbol{u}} \mathbb{P}\left(\boldsymbol{u}\right) \phi_{\mathcal{G}}\left(\boldsymbol{u}\right) - c^{*}\right) + \lambda\left(\sum_{\boldsymbol{u}} \mathbb{P}\left(\boldsymbol{u}\right) - 1\right). \tag{4.5}$$

Maximizing with respect to the probability of each labelling,

$$\frac{\partial L}{\partial \mathbb{P}\left(\boldsymbol{u}\right)} = -\log \mathbb{P}\left(\boldsymbol{u}\right) - 1 + \beta\phi_{\mathcal{G}}\left(\boldsymbol{u}\right) + \lambda = 0.$$

Rearranging and taking exponential, we have,

$$\mathbb{P}\left(\boldsymbol{u}\right) = \mathrm{e}^{\beta\phi_{\mathcal{G}}(\boldsymbol{u})+\lambda-1}. \tag{4.6}$$

For the parameters, we can derive λ, from the second constraint,

$$\begin{aligned}
\sum_{\boldsymbol{u}} \mathbb{P}\left(\boldsymbol{u}\right) = \sum_{\boldsymbol{u}} \mathrm{e}^{\beta\phi_{\mathcal{G}}(\boldsymbol{u})+\lambda-1} = 1 \\
\Rightarrow \mathrm{e}^{\lambda-1} \sum_{\boldsymbol{u}} \mathrm{e}^{\beta\phi_{\mathcal{G}}(\boldsymbol{u})} = 1 \\
\Rightarrow \mathrm{e}^{\lambda-1} = \frac{1}{\sum_{\boldsymbol{u}} \mathrm{e}^{\beta\phi_{\mathcal{G}}(\boldsymbol{u})}} \\
\Rightarrow \mathbb{P}\left(\boldsymbol{u}\right) = \frac{\mathrm{e}^{\beta\phi_{\mathcal{G}}(\boldsymbol{u})}}{\sum_{\boldsymbol{u}} \mathrm{e}^{\beta\phi_{\mathcal{G}}(\boldsymbol{u})}}.
\end{aligned} \tag{4.7}$$

In 4.7, the expression of sum over all possible labellings is also known as the partition function given by:

$$Z = \sum_{\boldsymbol{u}} \mathrm{e}^{\beta\phi_{\mathcal{G}}(\boldsymbol{u})}. \tag{4.8}$$

Interestingly, β can be derived using 4.7 as:

$$\frac{\partial}{\partial\beta} \log \sum_{\boldsymbol{u}} \mathrm{e}^{\beta\phi_{\mathcal{G}}(\boldsymbol{u})} = \frac{\sum_{\boldsymbol{u}} \phi_{\mathcal{G}}(\boldsymbol{u}) \, \mathrm{e}^{\beta\phi_{\mathcal{G}}(\boldsymbol{u})}}{\sum_{\boldsymbol{u}} \mathrm{e}^{\beta\phi_{\mathcal{G}}(\boldsymbol{u})}} = \sum_{\boldsymbol{u}} \mathbb{P}(\boldsymbol{u}) \, \phi_{\mathcal{G}}(\boldsymbol{u}). \tag{4.9}$$

Given that we know the partition function in Equation 4.8, this leads to the following simplification using 4.2

$$\beta = \frac{c^*}{\log Z} \int \partial\beta = \frac{c^*}{\log Z}\beta + c. \tag{4.10}$$

Therefore, the probability distribution as derived in 4.7, is the model of maximum entropy distribution. β is the Lagrange multiplier that behaves as a crucial parameter to control the uncertainty in the model, making a single labelling on a single graph preferable over others based on the cost variable $\phi(\boldsymbol{u})$; which in this case is the cut. Alternatively, $\beta = \frac{1}{\tau}$, where τ plays the role of that of temperature in a physical system. At the exact zero temperature $\tau = 0$, the probability distribution favours low cost solutions with minimum number of cuts with overwhelming probability. At lowest temperatures or the highest uncertainty setting, it becomes very difficult to make a jump from low to high cost solutions, quite possibly leading to locally optimum low cost solutions. Very low temperatures push down on the landscape towards equally probable minimum cut solutions while high temperatures allow more movement in different directions (including the wrong directions) and equilibrate over equally likely high cost solutions. Interestingly, at lower temperatures but not the lowest, it is unlikely to converge to either very low cost solutions or very high cost solutions, allowing just enough flexibility to move around to avoid getting stuck in the local optimum. Typically, decreasing the temperature very slowly (annealing), is likely to let it converge to low cost global solutions.

In practice, in order to predict the labelling with maximum probability, computing the sum over all possible states in 4.7 is not tractable. In other words, the expected distribution is too big. If there are n data points, C classes, with m known labels such that $m \ll n$, the computation would be exponential in the order of C^{n-m}. Clearly, this is intractable for large graphs. It has been a fairly established method to approximate the true distribution $\mathbb{P}(\boldsymbol{u})$, that avoids the summation over all possible labelling by introducing some clever factorized distributions. This is exactly what we are trying to achieve by revisiting this well known technique of approximating the true distribution by inducing a meanfield, in the light of semi supervised learning and sequential classification.

4.3 Motivation

Let us consider an example to motivate our approach for mean field approximation. We refer to this example at multiple places in the chapter. An electoral constituency decides to conduct a survey of voting interests of the people of the constituency. The constituency has representatives from three political parties that is Labour, Conservative and Liberals. There are about $30,000$ registered voters in the constituency. The survey committee decides to assign a level of uncertainty to the voting interest of the person; a probability to vote Conservative, Liberal or Labour. The probability distribution for the voting interests is a joint probability distribution over all the individual voting interests of the people in the constituency over all possible combinations. For the expected probability of voting interests of the community to be close to the true interest, the joint distribution needs to be computed which is of the exponential order of computing 3^{30000} probabilities. Clearly, this is an infeasible option for the survey committee. One can think of an approximation technique that could approximate the true probability distribution.

A way to think about this is to assume that individual person in the constituency votes independently of any other person's voting affiliation except for the individual's close group of friends and relatives. Using the social network of the community, the committee could assume a person's vote is only influenced by close links on her network. Thereby, the monumental task of trying to compute the summation over all possible combinations, is reduced in complexity to computing the product of the distribution over a set of factors (group of friends).

4.4 Related Work

Semi-supervised learning is a well established approach of learning from a few examples. On a graph that is built from the given labelled and unlabelled data points, where each datum is represented as a vertex, two vertices share the same label if they are connected by an edge. The standard graph labelling semi-supervised algorithms in the literature use the graph Laplacian in order to optimize the labelling consistent with the labels seen so far (Zhu et al., 2003; Zhu and Ghahramani, 2002; Herbster et al., 2005; Herbster, 2008). Graph Laplacian based methods suffer from the limitations in the light of many unlabelled data (Luxburg et al., 2010). Our work uses a similar regularity measure for smoothness of the labelling that instead induces an Ising model distribution over the vertices of the graph. Our work is different from the approximation method in label propagation (Zhu et al., 2003; Zhu and Ghahramani, 2002) as we do not drop the higher order terms in the approximation. Also, our energy equation 4.13, is the complement of theirs. Treeopt algorithm (Vitale et al., 2011; Cesa-Bianchi et al., 2009) is optimal over a tree without graph connectivity utilization. In-spite of being a decade

old, to the best of our knowledge the algorithms (Zhu et al., 2003; Zhu and Ghahramani, 2002; Vitale et al., 2011; Cesa-Bianchi et al., 2009) are still the state-of-the-art in online graph/tree labelling semi-supervised setting. We differ from the techniques discussed in Chapter 3, in the sense that our method is also applicable for multi-class classification.

4.5 Mean Field Approximation

We consider a unit weighted graph $\mathcal{G} = (V, E, w)$ where $w(e)$ is the weight of edge $e \in E$ (we assume that $w(e) = 0$ if $e \notin E$). For convenience, we denote the weight of edge (i, j) by $w(i, j) = w_{ij}$. We consider the scenario when we are given labels for a subset of the vertices, $\mathcal{L} \subset V$. We denote the label of vertex i by S_i, which can take a value from the class set $\mathcal{C}$. Our task is to assign labels to the unobserved vertices $\mathcal{U} = V \setminus \mathcal{L}$. The weights are taken to be a measure of similarity so that vertices that share an edge are more likely to have the same label than other vertices. Because of this, our labelling not only depends on the labelled vertices, but also on the structure of the graph (heavily connected components of the graph are likely to have the same label). In this sense this can be viewed as a semi-supervised learning problem (we have a few labelled examples, but we are also learning from unlabelled data). We use a probabilistic model to encode our uncertainty about the labels of the vertices in $\mathcal{U}$, where we assume,

$$\mathbb{P}(\boldsymbol{S}) = \frac{\mathrm{e}^{-\beta E(\boldsymbol{S},\boldsymbol{S}^o)}}{Z}, \qquad Z = \sum_{\boldsymbol{S}} \mathrm{e}^{-\beta E(\boldsymbol{S},\boldsymbol{S}^o)} \tag{4.11}$$

where $\boldsymbol{S}$ denotes the labels at the unobserved vertices, $\boldsymbol{S}^o$ the labels at the observed vertices. β is a parameter (the inverse temperature) encoding the degree of uncertainty, the sum is over the labels of the unobserved vertices,

$$\sum_{\boldsymbol{S}} \cdots = \left(\prod_{i \in \mathcal{U}} \sum_{S_i \in \mathcal{C}} \right) \cdots \tag{4.12}$$

and $E(\boldsymbol{S}, \boldsymbol{S}^o)$ is a minimum cut energy function given by 4.13. It is important to note that in label propagation (Zhu et al., 2003), the authors use the complement of this function. Here, the available label information is believed to be true with absolute certainty. This is equivalent to clamping the vertices, while the labels of the unlabelled vertices are approximated. The main idea of the energy function is in inducing an energy landscape over the vertices of the graph such that the minimum energy configuration is the minimum cut over all edges.

$$E(\boldsymbol{S}, \boldsymbol{S}^o) = \frac{1}{2} \sum_{(i,j) \in \mathcal{E}} w_{ij} \left[\!\!\left[S_i \neq S_j \right]\!\!\right]. \tag{4.13}$$

The factor of two can be avoided if the edges are only counted once. In other words, if an edge (i, j) is present in our edge set E; (j, i) is not present in E.

4.5.1 Approximating the Posterior

Note, the probability function in 4.11 favours labellings that minimise the number of edges whose vertices have different labels. Unfortunately, for large vertex sets, computation of the probability is intractable. The expected value of labelling on unobserved vertices, when marginalized over available labels, is still an exponentially big space. We therefore resort to using a variational approximation of the marginal of the posterior $\mathbb{P}(\boldsymbol{S})$ where we minimise the variation free energy given by,

$$\Phi(\boldsymbol{\theta}) = \sum_{\boldsymbol{S}} \mathbb{Q}(\boldsymbol{S}|\boldsymbol{\theta}) \log\left(\frac{\mathbb{Q}(\boldsymbol{S}|\boldsymbol{\theta})}{\exp(-\beta\, E(\boldsymbol{S}, \boldsymbol{S}^o))}\right) \tag{4.14}$$

where $\mathbb{Q}(\boldsymbol{S}|\boldsymbol{\theta})$ is taken as a separable probability distribution that factors or decouples into smaller distributions. For each vertex, an influence from its neighbourhood is measured by means of an intermediate mean field. In small regular networks, lattice, acyclic graphs and trees, the approximation is (most often) exact as one can sum over all possible labellings,

$$\mathbb{Q}(\boldsymbol{S}|\boldsymbol{\theta}) = \prod_{i\in\mathcal{U}} \sum_{\mu\in\mathcal{C}} \theta_i^{\mu} \left[\!\!\left[S_i = \mu \right]\!\!\right] \tag{4.15}$$

with $\theta_i^{\mu} \geq 0$ and for all vertices $\sum_{\mu\in\mathcal{C}} \theta_i^{\mu} = 1$. The parameters θ_i^{μ} can be interpreted as the marginal probability of the label for unlabelled vertex i to be in class μ. In our example of the election constituency, θ_i^{μ} is the best guess of the probability of individual i voting μ, where μ is either Conservative, Labour or Liberal, that only depends of the choices of the neighbours or friends of i. The decoupled distribution in 4.14, has very nice property of efficient computation, by means of utilizing the factorized distribution $\mathbb{Q}(\boldsymbol{S}|\boldsymbol{\theta})$ in 4.15 over all the unlabelled vertices. 4.15 can also be written as:

$$\mathbb{Q}(\boldsymbol{S}|\boldsymbol{\theta}) = \prod_{i\in\mathcal{U}} \sum_{\mu\in\mathcal{C}} \theta_i^{\mu} \left[\!\!\left[S_i = \mu \right]\!\!\right] = \prod_{i\in\mathcal{U}} \mathbb{Q}_i(S_i|\theta_i). \tag{4.16}$$

We can interpret 4.16, as the probability of label at vertex i. In our example, this would mean probability of either voting Conservative, Labour or Liberal as before.

4.5.2 Minimizing the KL divergence

Multiplying numerator and the denominator of right hand side in 4.14 by Z,

$$\Phi(\boldsymbol{\theta}) = \sum_{\boldsymbol{S}} \mathbb{Q}(\boldsymbol{S}|\boldsymbol{\theta}) \log\left(\frac{\mathbb{Q}(\boldsymbol{S}|\boldsymbol{\theta})}{Z\exp(-\beta\, E(\boldsymbol{S}, \boldsymbol{S}^o)/Z)}\right). \tag{4.17}$$

Since Z does not depend on $\boldsymbol{S}$, here it ends up behaving as a constant. We can then rewrite the above as,

$$\begin{aligned}\Phi(\boldsymbol{\theta}) &= \sum_{\boldsymbol{S}} \mathbb{Q}\left(\boldsymbol{S}|\boldsymbol{\theta}\right)\left(\log\left(\frac{\mathbb{Q}\left(\boldsymbol{S}|\boldsymbol{\theta}\right)}{\mathbb{P}\left(\boldsymbol{S}\right)}\right) - \log(Z)\right) \\ &= \mathrm{KL}\left(\mathbb{Q}\left(\boldsymbol{S}|\boldsymbol{\theta}\right)\middle\|\mathbb{P}\left(\boldsymbol{S}\right)\right) - \log(Z)\end{aligned} \tag{4.18}$$

where $\mathbb{P}\left(\boldsymbol{S}\right)$ and Z are defined in 4.11 and $\mathrm{KL}\left(\mathbb{Q}\left(\boldsymbol{S}|\boldsymbol{\theta}\right)\middle\|\mathbb{P}\left(\boldsymbol{S}\right)\right)$ is the Kullback-Leibler (KL) divergence given by

$$\mathrm{KL}\left(\mathbb{Q}\left(\boldsymbol{S}|\boldsymbol{\theta}\right)\middle\|\mathbb{P}\left(\boldsymbol{S}|\boldsymbol{S}^{o}\right)\right) = \sum_{\boldsymbol{S}} \mathbb{Q}\left(\boldsymbol{S}|\boldsymbol{\theta}\right)\log\left(\frac{\mathbb{Q}\left(\boldsymbol{S}|\boldsymbol{\theta}\right)}{\mathbb{P}\left(\boldsymbol{S}|\boldsymbol{S}^{o}\right)}\right). \tag{4.19}$$

We use the inequality $\log(x) \geq (x-1)$ in Theorem 4.1 to prove the non-negativity of the KL divergence (Kullback and Leibler, 1951)

Theorem 4.1. *(Kullback and Leibler, 1951) The KL-divergence between two probability distributions can be viewed as a measurement of their difference. It is non-negative and reaches it minimum value of zero when the two distributions are identical at least, the distributions can only differ on sets of measure zero.*

Proof.

$$\sum_{\boldsymbol{S}} \mathbb{Q}\left(\boldsymbol{S}|\boldsymbol{\theta}\right)\log\left(\frac{\mathbb{P}\left(\boldsymbol{S}|\boldsymbol{S}^{o}\right)}{\mathbb{Q}\left(\boldsymbol{S}|\boldsymbol{\theta}\right)}\right) \geq \sum_{\boldsymbol{S}} \mathbb{Q}\left(\boldsymbol{S}|\boldsymbol{\theta}\right)\left(\frac{\mathbb{P}\left(\boldsymbol{S}|\boldsymbol{S}^{o}\right)}{\mathbb{Q}\left(\boldsymbol{S}|\boldsymbol{\theta}\right)} - 1\right) \tag{4.20}$$

$$\geq \sum_{\boldsymbol{S}} \mathbb{P}\left(\boldsymbol{S}|\boldsymbol{S}^{o}\right) - \sum_{\boldsymbol{S}} \mathbb{Q}\left(\boldsymbol{S}|\boldsymbol{\theta}\right) = 0 \tag{4.21}$$

□

Since $\log(Z)$ in Equation 4.11 does not depend on the variational parameters, $\boldsymbol{\theta}$, minimising $\Phi(\boldsymbol{\theta})$ is equivalent to minimising the KL-divergence. The KL-divergence in Equation 4.19 is minimized when $\mathbb{Q}\left(\boldsymbol{S}|\boldsymbol{\theta}\right) = \mathbb{P}\left(\boldsymbol{S}|\boldsymbol{S}^{o}\right)$, then $\log 1 = 0$. Thus, in minimising the variational free energy we are choosing the parameters $\boldsymbol{\theta}$ so that $\mathbb{Q}\left(\boldsymbol{S}|\boldsymbol{\theta}\right)$ is as close as possible (as measured by the KL-divergence) to $\mathbb{P}\left(\boldsymbol{S}\right)$. Furthermore as the KL-divergence is non-negative we obtain a bound that $-\log(Z) \leq \Phi(\boldsymbol{\theta})$ (in classical physics $-\beta\log(Z)$ is known as the free energy).

4.5.3 Minimizing the Entropy

We can also rewrite the variational free energy from 4.17,

$$\Phi(\boldsymbol{\theta}) = \sum_{\boldsymbol{S}} \mathbb{Q}\left(\boldsymbol{S}|\boldsymbol{\theta}\right)\log(\mathbb{Q}\left(\boldsymbol{S}|\boldsymbol{\theta}\right)) + \beta\sum_{\boldsymbol{S}} \mathbb{Q}\left(\boldsymbol{S}|\boldsymbol{\theta}\right)E(\boldsymbol{S},\boldsymbol{S}^{o}) \tag{4.22}$$

$$\implies \Phi(\boldsymbol{\theta}) = -H(\mathbb{Q}) + \beta\, U(\mathbb{Q}) \tag{4.23}$$

where $H(\mathbb{Q})$ is the entropy of $\mathbb{Q}\,(\boldsymbol{S}|\boldsymbol{\theta})$

$$\begin{aligned}
H(\mathbb{Q}) &= -\sum_{\boldsymbol{S}} \mathbb{Q}\,(\boldsymbol{S}|\boldsymbol{\theta})\ \log(\mathbb{Q}\,(\boldsymbol{S}|\boldsymbol{\theta})) \\
&= -\sum_{\boldsymbol{S}} \prod_{i\in\mathcal{U}} \mathbb{Q}_i(S_i|\theta_i) \log \prod_{i\in\mathcal{U}} \mathbb{Q}_i(S_i|\theta_i) \\
&= -\sum_{\boldsymbol{S}} \prod_{i\in\mathcal{U}} \mathbb{Q}_i(S_i|\theta_i) \sum_{i\in\mathcal{U}} \log \mathbb{Q}_i(S_i|\theta_i) \\
&= -\sum_{\boldsymbol{S}} \sum_{i\in\mathcal{U}} \mathbb{Q}_i(S_i|\theta_i) \log \mathbb{Q}_i(S_i|\theta_i) \\
&= -\sum_{i\in\mathcal{U}} \sum_{\kappa\in\mathcal{C}} \sum_{\mu\in\mathcal{C}} \theta_i^{\mu} \left[\!\!\left[\kappa = \mu\right]\!\!\right] \log \sum_{\nu\in\mathcal{C}} \theta_i^{\nu} \left[\!\!\left[\kappa = \nu\right]\!\!\right] \\
&= -\sum_{i\in\mathcal{U}} \sum_{\mu\in\mathcal{C}} \theta_i^{\mu}\ \log(\theta_i^{\mu})\,.
\end{aligned} \tag{4.24}$$

The simplification in 4.24, is achieved by applying 4.16, 4.15 and the distributive law. In 4.24, i sums over all the unlabelled vertices while μ sums over all the classes; in our example, the summation is over the finite constituency of all people and the three parties. It is interesting to note that without the approximation, an exponential number of 3^{30000} computations were required. In 4.24, the computation is reduced a to linear 3×30000 computations. $U(\mathbb{Q})$ is the "mean energy" or "expected energy" with respect to the probability distribution $\mathbb{Q}\,(\boldsymbol{S}|\boldsymbol{\theta})$

$$\begin{aligned}
U(\mathbb{Q}) &= \sum_{\boldsymbol{S}} \mathbb{Q}\,(\boldsymbol{S}|\boldsymbol{\theta})\ E(\boldsymbol{S}, \boldsymbol{S}^o) \\
&= \sum_{\boldsymbol{S}} \mathbb{Q}\,(\boldsymbol{S}|\boldsymbol{\theta})\ \frac{1}{2} \sum_{(i,j)\in\mathcal{E}} w_{ij} \left[\!\!\left[S_i \neq S_j\right]\!\!\right] \\
&= \frac{1}{2} \sum_{i\in\mathcal{U}} \sum_{j\in\mathcal{U}} w_{ij} \sum_{\mu,\nu\in\mathcal{C}} \theta_i^{\mu}\theta_j^{\nu} \left[\!\!\left[\mu \neq \nu\right]\!\!\right] \\
&\quad + \sum_{i\in\mathcal{U}} \sum_{j\in\mathcal{L}} w_{ij} \sum_{\mu\in\mathcal{C}} \theta_i^{\mu} \left[\!\!\left[\mu \neq S_j\right]\!\!\right] \\
&\quad + \frac{1}{2} \sum_{i\in\mathcal{L}} \sum_{j\in\mathcal{L}} w_{ij} \left[\!\!\left[S_i \neq S_j\right]\!\!\right].
\end{aligned} \tag{4.25}$$

The simplification in 4.25 is achieved by applying 4.13. In 4.25 and 4.24, the entropy is minimized when few labels are known while the mean energy is minimized when there are few disagreeing labels. In order to find the minimum of the non-linear variational free energy in 4.23, subject to $\sum_{\mu\in\mathcal{C}} \theta_i^{\mu} = 1$, at each vertex we minimise the Lagrangian,

$$L(\boldsymbol{\theta}) = \Phi(\boldsymbol{\theta}) + \sum_{i\in\mathcal{U}} \lambda_i (\sum_{\mu\in\mathcal{C}} \theta_i^{\mu} - 1) \tag{4.26}$$

where the λ_i's are a set of Lagrange multipliers that are chosen to ensure the constraints are satisfied. The "mean-field equations" which are satisfied at the minima of the variational free energy are given by

$$\frac{\partial L(\boldsymbol{\theta})}{\partial \theta_i^\mu} = \log(\theta_i^\mu) + 1 + \beta \sum_{j \in \mathcal{U}} w_{ij} \sum_{\substack{\nu \in \mathcal{C} \\ \nu \neq \mu}} \theta_j^\nu + \beta \sum_{j \in \mathcal{L}} w_{ij} \left[\!\!\left[\mu \neq S_j \right]\!\!\right] + \lambda_i = 0. \tag{4.27}$$

These equations are not in general solvable in closed form. Our goal is to maximize with respect to λ_i and minimize with respect to θ; while essentially looking for a saddle point solution. We choose λ_i in order to satisfy the constraint $\sum_\mu \theta_i^\mu = 1$ and by rearranging the terms in 4.27, we have

$$\begin{aligned} \sum_{\mu \in \mathcal{C}} \theta_i^\mu = \sum_{\mu \in \mathcal{C}} \mathrm{e}^{\lambda_i + 1 + 2\beta \sum_{\substack{j \in \mathbb{N}_i \\ j \in \mathcal{U}}} w_{ij} \sum_{\substack{\nu \neq \mu \\ \nu \in \mathcal{C}}} \theta_j^\nu} = 1 \\ \implies \mathrm{e}^{\lambda_i + 1} \sum_{\mu \in \mathcal{C}} \mathrm{e}^{2\beta \sum_{\substack{j \in \mathbb{N}_i \\ j \in \mathcal{U}}} w_{ij} \sum_{\substack{\nu \neq \mu \\ \nu \in \mathcal{C}}} \theta_j^\nu} = 1 \\ \implies \mathrm{e}^{\lambda_i + 1} = \frac{1}{\sum_{\mu \in \mathcal{C}} \mathrm{e}^{2\beta \sum_{\substack{j \in \mathbb{N}_i \\ j \in \mathcal{U}}} w_{ij} \sum_{\substack{\nu \neq \mu \\ \nu \in \mathcal{C}}} \theta_j^\nu}}. \end{aligned} \tag{4.28}$$

λ_i thus chosen, gives the normalisation term in 4.29. θ_i^μ is always positive, with $\sum_{\substack{\nu \in \mathcal{C} \\ \nu \neq \mu}} \theta_j^\nu$ values propagated from neighbours of i where $j \in \mathbb{N}_i$. Instead we can attempt to solve these equations iteratively by setting,

$$\theta_i^\mu(t+1) = \frac{\mathrm{e}^{-\beta \tilde{E}_i^\mu(\boldsymbol{\theta}(t), \boldsymbol{S}^o)}}{\sum_{\nu \in \mathcal{C}} \mathrm{e}^{-\beta \tilde{E}_i^\nu(\boldsymbol{\theta}(t), \boldsymbol{S}^o)}} \tag{4.29}$$

where,

$$\tilde{E}_i^\mu(\boldsymbol{\theta}, \boldsymbol{S}^o) = \sum_{j \in \mathcal{U}} w_{ij} \sum_{\substack{\nu \in \mathcal{C} \\ \nu \neq \mu}} \theta_j^\nu + \sum_{j \in \mathcal{L}} w_{ij} \left[\!\!\left[\mu \neq S_j \right]\!\!\right] \tag{4.30}$$

which is a self-consistent soft-max solution. For the available labels $\boldsymbol{S}^o$, we set their corresponding θ_i values that satisfy 4.29. The θ_i^μ values are updated iteratively. There can be many local solutions to the mean-field equations so that the result will depend on the initial conditions. To prevent finding very poor solutions we can anneal the temperature (start from a low value of β and increase it to the required value) at each iteration. [1]

[1] We may further wish to randomly choose the order of the variables we are updating to reduce the bias caused by the order of updating (alternatively we can update all the variables at once).

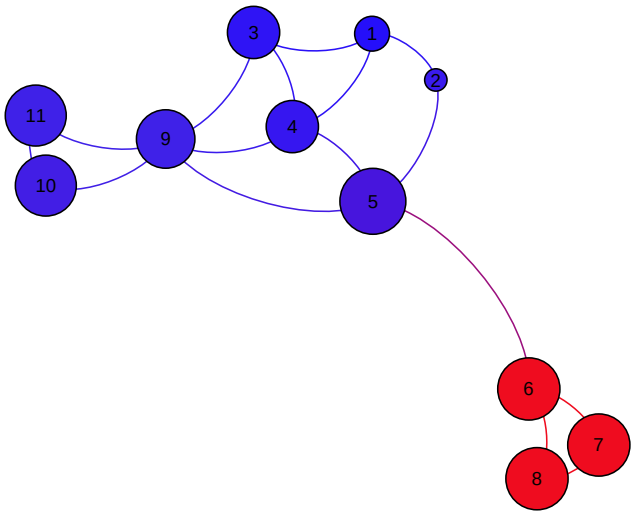


Figure 4.1: Toy Graph with Vertices 1 and 4 labelled 1 while vertex 8 labelled -1. The graph shows the ground truth labelling. The minimum cut size is 1, with the edge between vertices 5 and 6 as the cut edge. The strongly connected clusters are $9-10-11$, $6-7-8$ and $1-2-3-4-5$.

4.5.4 Online Meanfield Game

As in any online learning algorithm, a sequential game is played between the learner and the adversary or the environment. The learner's goal is to minimize its mistaken predictions (incorrect class prediction) while the adversary's goal is the opposite. Nature selects a data-point (vertex) at every trial, learner predicts the class, nature then returns the true label to verify if the learner made a mistake. The adversary is regular in nature; else with an adaptive adversarial opponent, the learner will always make a mistake. The only restriction is the adversary cannot return a label that increases the minimum-cut. For the prediction of the class label, we allow to slowly anneal the value of β until stabilized. Once stabilized, we predict the label of the query point by the class label that maximizes θ for that vertex. Once the true label is revealed, we update the partial labelling vector $\boldsymbol{S}^o$, increment the mistake count (if a mistake has been made) and repeat the process . Since we assume the adversary cannot increase the minimum cut, it cannot force a mistake on the learner; the cost of the feedback from the environment is trivial. In the Figure 4.2, the online mean field approximation method is provided. We perform experiments using our algorithm and report results in Section 4.6. The experiments are performed in batches; each batch with a set number of labels available with labels increasing across the batches. This is essentially the same as running an online algorithm.

4.5.4.1 Algorithm

Parameters: Graph $\mathcal{G}$ of size N, uncertainty parameter β;

Input: Available labels: $\boldsymbol{S}^o \in L$, classes: $\mathcal{C}$, unlabelled vertices: $\boldsymbol{S} \in \mathcal{U}$

Example sequence: $\mathcal{Z} = \langle (i_1, \mathcal{C}_{i_1}), (i_2, \mathcal{C}_{i_2}), (i_3, \mathcal{C}_{i_3}), \ldots, (i_\ell, \mathcal{C}_{i_\ell}) \rangle, i_t \in \mathcal{U}, \mathcal{C}_{i_t} \in \mathcal{C}$

Initialization: θ_1 is the initial distribution over $\mathcal{G}$ such that,

$\theta_1 = (\frac{1}{0.5}, \frac{1}{0.5}, \ldots, \frac{1}{0.5})^{\mathcal{U} \times \mathcal{C}}$, $\beta = 0.01$

for $t = 1, \ldots, |\mathcal{U}|$ **do**

Receive: $i_t \in \{1, \cdots, \mathcal{U}\}$

Stabilize: **while** β not stabilized for all $i \in \mathcal{U}$ **do**

$$\theta_i^{\mu}(t) = \frac{\mathrm{e}^{-\beta \tilde{E}_i^{\mu}(\boldsymbol{\theta}(t-1), \boldsymbol{S}^o)}}{\sum\limits_{\nu \in \mathcal{C}} \mathrm{e}^{-\beta \tilde{E}_i^{\nu}(\boldsymbol{\theta}(t-1), \boldsymbol{S}^o)}} \text{ for all } \mu, \nu \in \mathcal{C} \text{ using 4.29}$$

Anneal: β slowly

end

Predict: Label $\hat{y}_t(i_t | \mathcal{Z}_{t-1}) := \mathrm{argmax}_{\mu \in \mathcal{C}}\, \theta_{i_t}^{\mu}(S_{i_t} = \mu | S_{i_1} = \mu_1, \ldots, S_{i_{t-1}} = \mu_{t-1})$

Receive: True Label y_t;

Compute: Mistake $M = |\{t : \hat{y}_t \neq y_t\}|$

Update: $S_{i_t}^o = y_t$

end

Figure 4.2: Online `meanField` Approximation Algorithm.

4.6 Experiments

We perform empirical tests to evaluate our method on the standard machine learning datasets from UCI (Lichman, 2013). We use our own implementation of the `meanField` method and the competitor algorithm `labelProp` (Zhu et al., 2003; Zhu and Ghahramani, 2002), while we adapt the code for `treeOpt` given to us by the authors in (Vitale et al., 2011). In general, for our experiments we use an uniform way of sampling instances and building the graphs from the datasets. For datasets `ISOLET` (UCI), `webSpam` (Webspam, 2007) and a subset `20 newsGroups` (Roweis, 2006), we randomly sample instances from the entire dataset. We also construct synthetic dataset to test our methods.

4.6.1 Datasets

The sampling of instances and labels ensure that both the classes are equally represented. Each instance is represented as a vertex in the graph. An Euclidean distance matrix

is constructed using the pairwise distances between the instances. In the case of `20 newsGroups`, instead of using the Euclidean distance matrix, we use the cosine distance matrix for binary valued instances. We build a 3-NN nearest neighbour graph using the distance matrix built as the previous step. For ensuring that the graph thus constructed is connected, we always sample a minimum spanning tree (randomly) using the Euclidean distance or cosine distance as weights. We ensure that the MST edges are maintained. Having MST edges also allows for sparsity in the graph. All trials receive the same graph. We choose the connectivity of $K = 3$ for our experiments, as in the literature, empirical evidences show competitive performance for 3-NN connectivity. We use quad-core processor notebooks (@2.30 GHz each) with 8GB and 16GB RAM. We also use the Iridis 4 HPC cluster at University of Southampton, UK.

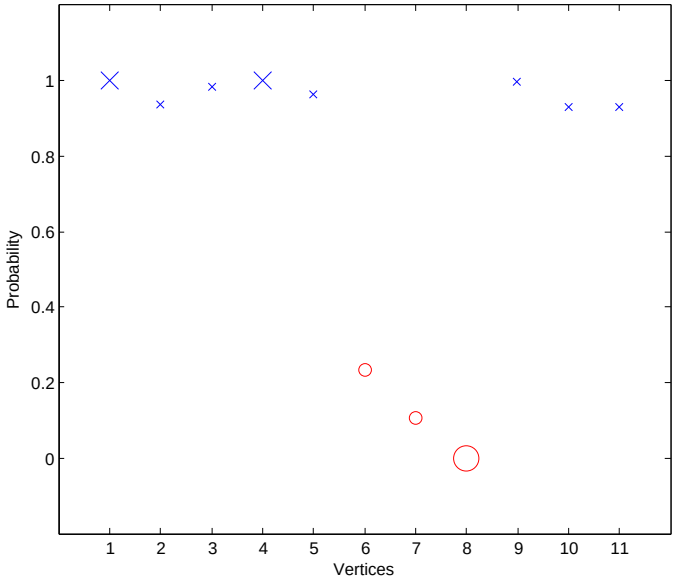


Figure 4.3: Toy Graph result of the marginal probability of the labelling for class 1 over the vertices. The strongly connected clusters tend to be uniformly labelled. The crosses represent class 1 while the circles represent class -1. The magnified crosses represent the labelled vertices for class 1 with probability 1, magnified circle for labelled vertex in class 1 with probability 0.

The synthetic dataset `Squares` is a 60x60 image from which a grid graph with 3600 vertices is constructed, where each pixel is represented as a vertex. We ensure that the graph has toroidal boundary properties such that each vertex is surrounded by four neighbours based on pixel locations. The results of the experiments for each dataset are discussed in Table 4.1 through to Table 4.4. The performance of the algorithms is measured as the generalized accuracy of the prediction, that is number of correctly classified instances over all the instances; higher the better. For all datasets, results are averaged over 10 trials except for `20 newsGroups` which used 5 trials. The randomly sampled labels l are balanced. For all the experiments, the choice of β is annealed

to a value of 2.4 before prediction. Due to computational feasibility, we perform the experiments in the batch setting rather than online.

4.6.2 Toy Dataset

In Figure 4.1, we show our toy graph. The vertices 1 and 4 are labelled 1 while vertex 8 is labelled -1. The Figure depicts the ground truth labelling. We perform experiments using the approximation technique of our algorithm to predict the labelling, on the toy data and report two sets of results. In Figure 4.3, we report the marginal probabilities of the labels of the vertices after the θ values are stabilized. The blue crosses belong to one class while the red circles belong to the other. The magnified blue crosses at vertices 1 and 4 are available labels while the magnified red circle at vertex 8 is another available label of the opposite class. We see the approximated θ value of all other unlabelled vertices; here the θ values for class 1 are shown. In Figure 4.4, we see the results of the change in marginal probability of the label of a vertex with respect to the annealed β values. Each sub-plot is for a particular vertex showing the class its label converges to. For vertices 1, 4 and 8, the class is already available. For the rest, the class label is approximated and the colour that the class label corresponds to, for a particular β is shown in the plot.

Table 4.1: Results on `Squares` for classification of 0 intensity pixels against 1 intensity pixels. `labelProp` and `meanField` are extremely competitive, with `labelProp` eventually outperforms `meanField`.

	$\ell = 250$	$\ell = 450$	$\ell = 650$	$\ell = 850$	$\ell = 1050$
`labelProp`	$.842 \pm .010$	$\mathbf{.908} \pm .005$	$\mathbf{.940} \pm .005$	$\mathbf{.956} \pm .004$	$\mathbf{.969} \pm .003$
`meanField`	$\mathbf{.846} \pm .014$	$.894 \pm .006$	$.925 \pm .006$	$.942 \pm .005$	$.955 \pm .005$

Table 4.2: Results on `20 newsGroups` which is a sparse dataset. Label distribution is 8124:8118 on classifying (comp.* and rec.* Vs. sci.* and talk.*) newsgroups. `meanField` beats `labelProp` with enough information. `treeOpt` underperforms until the labels are sufficiently large for comparable performance.

	$\ell = 800$	$\ell = 1000$	$\ell = 2000$	$\ell = 4000$	$\ell = 6000$
`labelProp`	$\mathbf{.826} \pm .005$	$\mathbf{.826} \pm .006$	$\mathbf{.845} \pm .004$	$\mathbf{.871} \pm .002$	$.868 \pm .003$
`treeOpt`	$.753 \pm .006$	$.772 \pm .01$	$.805 \pm .002$	$.858 \pm .004$	$.868 \pm .002$
`meanField`	$.800 \pm .006$	$.805 \pm .009$	$.833 \pm .003$	$.865 \pm .002$	$\mathbf{.890} \pm .002$

Table 4.3: Results on `webSpam` which is a sparse dataset of a computer hosts network. Classifying spam Vs. non-spam hosts, we see that `meanField` eventually surpasses `labelProp`, with `treeOpt` being competitive.

	$\ell = 200$	$\ell = 400$	$\ell = 600$	$\ell = 800$	$\ell = 900$
`labelProp`	$.\mathbf{729} \pm .075$	$.698 \pm .015$	$.\mathbf{803} \pm .001$	$.\mathbf{805} \pm .0001$	$.806 \pm .0001$
`treeOpt`	$.728 \pm .018$	$.\mathbf{734} \pm .007$	$.71 \pm .013$	$.747 \pm .003$	$.749 \pm .001$
`meanField`	$.677 \pm .036$	$.645 \pm 0.002$	$.770 \pm .008$	$.800 \pm .003$	$.\mathbf{808} \pm .0004$

Table 4.4: Results on `ISOLET`-1 through 5 for all instances. `meanField` beats `labelProp` in most of the settings. `ISOLET` has a 3900:3897 ratio between the labels while classifying the first 13 letters against the next 13 letters. Files `ISOLET 1` through `ISOLET 5` are used for the construction of the graph.

	$\ell = 32$	$\ell = 64$	$\ell = 128$
`labelProp`	$.661 \pm .039$	$.\mathbf{707} \pm .029$	$.764 \pm .024$
`treeOpt`	$.658 \pm .042$	$.688 \pm .033$	$.731 \pm .012$
`meanField`	$.\mathbf{700} \pm .022$	$.700 \pm .022$	$.\mathbf{791} \pm .014$
	$\ell = 256$	$\ell = 512$	$\ell = 1024$
`labelProp`	$.787 \pm .015$	$.82 \pm .012$	$.\mathbf{869} \pm .006$
`treeOpt`	$.786 \pm .022$	$.824 \pm .008$	$.859 \pm .005$
`meanField`	$.\mathbf{813} \pm .017$	$.\mathbf{837} \pm .011$	$.860 \pm .006$

4.6.3 Discussion and Conclusion

The main focus of our work is in minimizing the total number of mistakes and not the computational efficiency of the method. The inverse temperature β controls the amount of uncertainty in our model. We could choose it through cross-validation rather than through empirical evaluation. Alternatively, we can take a Bayesian interpretation in which we take the joint probability of the unobserved spins $\boldsymbol{S}$ and the observed spins $\boldsymbol{S}^o$ to be

$$\mathbb{P}(\boldsymbol{S}, \boldsymbol{S}^o) = \frac{\mathrm{e}^{-\beta E(\boldsymbol{S},\boldsymbol{S}^o)}}{Z'}, \qquad Z' = \sum_{\boldsymbol{S},\boldsymbol{S}^o} \mathrm{e}^{-\beta E(\boldsymbol{S},\boldsymbol{S}^o)}$$

where Z' is the partition function under the assumption that no labels are observed. The probability of the observed spins is

$$\mathbb{P}(\boldsymbol{S}^o) = \sum_{\boldsymbol{S}} \frac{\mathrm{e}^{-\beta E(\boldsymbol{S},\boldsymbol{S}^o)}}{Z'} = \frac{Z}{Z'}.$$

We have seen that the variational free energy $\Phi(\boldsymbol{\theta}^*)$ (where $\boldsymbol{\theta}^*$ is our solution to the mean-field equation) acts as an approximation for $-\log(Z)$ (the evidence). We can similarly introduce a variational free energy to compute $-\log(Z')$ (we repeat the calculation except with no observed spins). Choosing the value of β which maximises the

difference $\log(Z) - \log(Z')$ would provide an approximation to the best value of β (i.e. it is the value which maximises the probability of the data). However, for very large problems this may not be feasible and we may just have to use some guess for β based on experimentation. Alternatively, β can be considered as the free parameter, that can be learnt as a hyperparameter from the prior. A nice feature of this framework is that we obtain the marginal distribution for the labels. This can be used in any decisions theoretic framework.

The quality of the mean-field approximation is hard to determine a priori. In practice, the mean field approximation methods often converge although theoretically, they might not. In regular graphs, the mean field approximation is exact. The separable probability distribution $\mathbb{Q}(\boldsymbol{S}|\boldsymbol{\theta})$ will not capture the strong dependencies between many of the labels. These are not entirely ignored in the approximation since they come in through the $U(\mathbb{Q})$ term. The approximation technique is not guaranteed to converge. However, in practice, most often the methods converge in polynomial time. One can investigate the quality of the approximation by considering a small system where we can compute the sum over all labellings exactly.

This concludes our discussion on the work in this chapter about online learning meanfield approximation technique for graph labelling in a semi-supervised setting. This leads us to possible future work. If we incorporate the Halving algorithm and other online graph prediction techniques (Littlestone, 1988; Herbster et al., 2005), where we predict such that maximum number of hypotheses are eliminated in case of a mistake, we are sure to see `meanField` challenging `labelProp` more often. Also, if we relax the adversary such that in can increase the minimum cut, then that could easily be the beginning of another online learning game, the idea then should be to minimize mistakes over all such online minimum cut games.

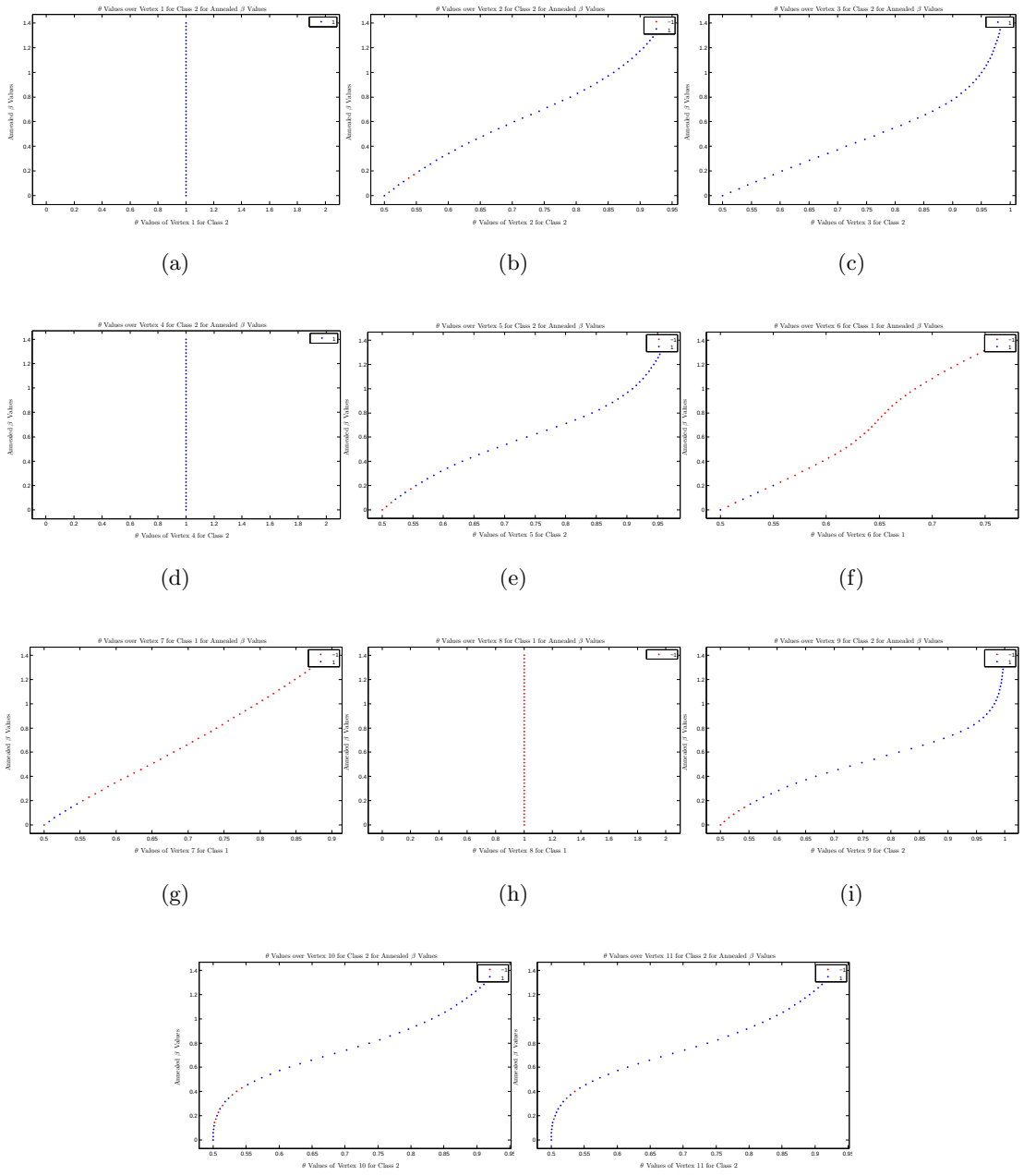


Figure 4.4: Toy Graph plots for marginal probability of the labelling of the vertices versus the θ values. In all the figures, class 1 and class 2 denote the class label, that the θ for a particular vertex is converging towards, with maximum probability. In the plots, the y axis represents the annealed β values that is the same in all cases. x axis is the varying θ values for vertices as follows: (a) available label of vertex 1 (b) approximated label of vertex 2 (c) approximated label of vertex 3 (d) available label of vertex 4 (e) approximate label of vertex 5 (f) approximate label of vertex 6 (g) approximate label of vertex 7 (h) available label of vertex 8 (i) approximate label of vertex 9 (j) approximate label of vertex 10 (k) approximate label of vertex 11.

Chapter 5

Ising Bandits with Side Information

5.1 Introduction

This chapter marks the beginning of the part of the thesis that deals with varying feedback settings, multiple objective optimization and transition to the basic set-up of online learning procedures. This chapter in particular, tackles a problem of sequential action selection under limited (bandit) feedback with side information, where actions are inter-related by structural relationships on a graph. The desired goal of optimal action selection is achieved by exploiting the knowledge of the graphical structure of the data in an online setting. We do so by associating a complexity with the labelling of the graph. As we studied in the previous chapters, this complexity is called the "cut" or "energy" of the labelling on a Markov random field with discrete states (Ising model).

The ultimate objective of the algorithm that we develop is to perform a graph labelling procedure in order to minimize the energy while being consistent with the side information seen so far. This helps in deciding the intrinsic state of the queried vertex at every round before selecting the action to play. Selecting the appropriate action given the current state of the graph, directs the overall goal towards minimizing the regret of our sequential action selection (bandit) algorithm within the online graph labelling that occurs over the entire sequence.

5.2 Background

5.2.1 Semi-supervised Graph Classifier Complexity

We revisit the background in semi supervised online labelling over graphs that we have already discussed in the preceding chapters. The standard approach in semi supervised learning is to construct the graph from the unlabelled and labelled data such that each datum is denoted as a vertex. Traditionally, the norm induced by the graph Laplacian is used to predict the labelling. Typically, either the norm induced by the Laplacian is directly minimized (interpolated) with respect to constraints or it is used as a regulariser. Both methods help build classifiers on graphs in order to learn sparse labels in $\mathbb{R}^n$ by incorporating a measure of complexity also called "cut" or energy. For a graph $\mathcal{G} = (V, E)$, where the set of vertices $V = \{v_1, \ldots, v_n\}$ are connected by edges in E. Let a weight of A_{ij} be associated with every edge $(i, j) \in E$, such that $\mathbf{A}$ is the $n \times n$ symmetric adjacency matrix, then the Laplacian $\mathbf{L}$ of the graph is given by $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D}$ is the degree matrix with its diagonal values given by $D_{ii} = \sum_j A_{ij}$. We re-state Definition 1 from (Herbster et al., 2009) that relates the quadratic form of the Laplacian with the complexity of the "cut-size" for completeness.

Definition 5.1 (Herbster et al. (2009))**.** If the labelling of the graph $\mathcal{G}$ is given by $\boldsymbol{u} \in \mathbb{R}^n$, the "cut size" of $\mathbf{u}$ is given by

$$\psi_{\mathcal{G}}(\boldsymbol{u}) = \frac{1}{4}\boldsymbol{u}^T \boldsymbol{L}\boldsymbol{u} = \frac{1}{4}\sum_{(i,j)\in E} A_{ij}(u_i - u_j)^2 \ . \tag{5.1}$$

When $\boldsymbol{u} \in \{0, 1\}^n$, the "cut" is on the edge (i, j) where $u_i \neq u_j$, then $\psi_{\mathcal{G}}(\boldsymbol{u})$ is the number of "cut" edges. $\psi_{\mathcal{G}}(\boldsymbol{u})$ is also known an the energy function $E(\boldsymbol{u})$.

The smoothness functional of $\boldsymbol{u}^T \boldsymbol{L}\boldsymbol{u}$ is generalized in the work of semi-norm interpolation (Herbster and Lever, 2009) where the Laplacian $p-$seminorm is defined on $\boldsymbol{u} \in \mathbb{R}^n$ as:

$$||\boldsymbol{u}||_{\mathcal{G},p} \simeq \psi_{\mathcal{G}}(\boldsymbol{u}) = \left(\sum_{(i,j)\in E} A_{ij}\,|u_i - u_j|^p\right)^{\frac{1}{p}} \ . \tag{5.2}$$

When $p = 2$, this is equivalent to the harmonic energy minimization technique in (Zhu et al., 2003). Alternatively, this technique is also called the Laplacian interpolated regularization (Belkin et al., 2004). In Herbster et al. (2009), the online version of the $p = 2$ case is studied in the context of the already available labels. If $\mathcal{G}$ is a partially labelled graph as in our problem, such that $|V| = N$, and the partial labels $l \leq N$, with the labels given by $\boldsymbol{y}_l \in \{1, -1\}^l$ on the l vertices, then the minimum semi-norm interpolation gives the labelling:

$$\boldsymbol{y} = \underset{\boldsymbol{u}}{\operatorname{argmin}}\{\boldsymbol{u}^T \boldsymbol{L}\boldsymbol{u} : \boldsymbol{u} \in \mathbb{R}^n, u_r = y_r, r = 1, \ldots, l\} \ .$$

The prediction is made by using $\hat{y}_i = \text{sgn}(y_i)$ (Herbster and Lever, 2009). The rationale behind minimizing the cut enables the neighbouring vertices to have similarly valued labels. With $p \to 1$, the prediction problem is reduced to predicting using the label consistent minimum cut.

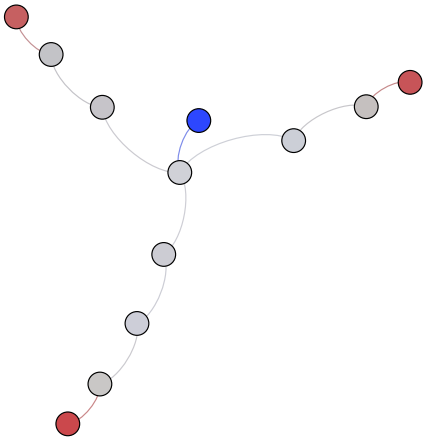
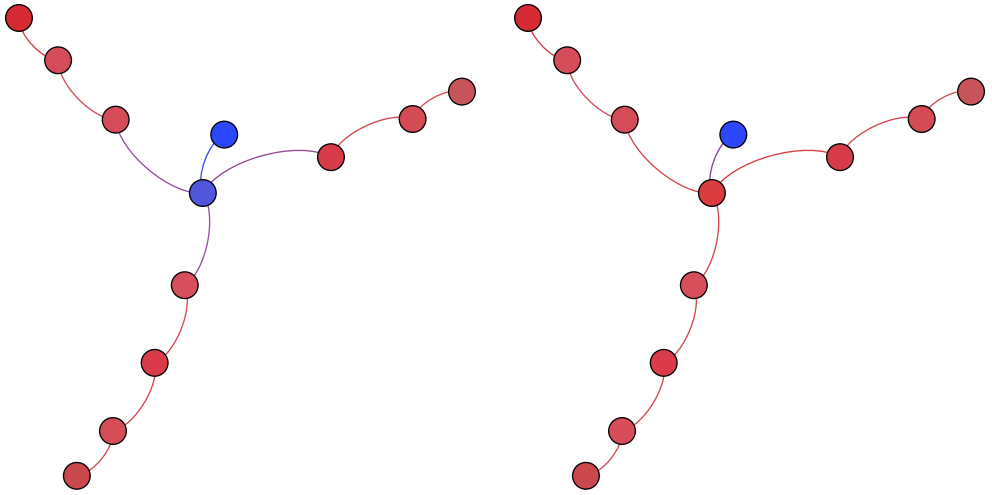


Figure 5.1: 3-armed `Octopus Graph` with four nodes labelled: head labelled blue and three labelled red on the arms.



(a) (b)

Figure 5.2: Optimization results on `Octopus Graph` with label propagation and maximum flow (a) Quadratic harmonic energy minimization with $p \to 2$ (Zhu et al., 2003; Herbster et al., 2009)(b) Linear programming for maximum flow with $p \to 1$ in algorithm 5.3.

5.3 Motivation

Often, interesting applications are tied to problems with rich underlying structure. For example, consider the system of online advertising; serving advertisements on web pages in an incremental fashion. The web pages can be represented as vertices in the graph with the links as edges. At given time t, the system receives a request to serve an advertisement on a randomly selected web-page. Further, at the same time, the system receives side information about the state of the web-page: for simplicity we assume the side information to be a rating of 0 or 1. As a consequence, the advertisement pool from which the advertisement is to be served supposedly changes with the change in the state of the graph. However, this contextual change in the action pool is kept hidden from the system. The goal for the system is to guess what could be the current state of the advertisements given the side information it received along with the past information and serve the most appropriate advertisement on the queried webpage. Once the chosen advertisement is served, the feedback is received which the system then incorporates in its learning algorithm in order to serve the next request.

At a deeper level of understanding, the side information can be interpreted as the label of the vertex. There are few available labels at the start; the rest are only incrementally revealed. So some of the ratings of the webpages are already known, while most of the webpages on which the advertisement needs to be served, the rating is not known until at the time of query. When a vertex is queried (request for an ad placement is made), an action needs to be picked (advertisement needs to be served) from a set of actions. The algorithm should be able to internally predict what the state of the queried vertex is (how the state of the graph changes) and then select the appropriate action from the action pool that (potentially) changes with the predicted label of the queried vertex.

5.4 Related Work

Broadly speaking, there are two central themes that run through this chapter unified under the common framework of online learning, namely, action selection using bandit feedback and semi-supervised graph labelling. The closest related work that addresses the intersection of these two themes is the work by Claudio et al. (Gentile and Orabona, 2014). They use bandit feedback to address a multi-class ranking problem. The algorithm outputs a partially ordered subset of classes and receives only bandit feedback (partial information) among the selected classes it observes without any supervised ranking feedback. In contrast, we play the bandit game of sequential action selection, using side information as the class label of the current context. Our feedback for the action selected is still partial (only loss for the selected action is observed). Further, our bandits have a structure associated with the Ising model distribution over the vertices at low temperature. The work of Amin et al. (2012), addresses the graphical models for

bandit problems to demonstrate the rich interactions between the two worlds, along the similar lines of what we are trying to achieve. Bearing a strong resemblance to our work, they address the similar context-action space. However, in their setting, there is a strong coupling between the context-action space; the algorithm needs to fulfil the entire joint assignment before receiving any feedback. In contrast, our concept-action space is decoupled, labels are revealed gradually determining the current active concept for the learner to choose the action and receive the feedback instantaneously. In their problem formulation under the Ising graph setting, the algorithm tries to pick the action (the label of the concept) that is NP hard. In contrast, we focus on the low temperature setting, where our actions lie on the edges, and are not the labels of the vertices. The computation of the marginal at the vertices is guided by the labels seen so far and the minimal cut. We approximate the labelling of the entire graph rather than predicting the spin configuration of a single vertex using the "cut" as the regularizer that dominates the action selection. The contextual bandits work on online clustering of bandits (Gentile et al., 2014), deals with finding groups or clusters of bandits in the graphs. They have a stochastic assumption of a linear function for reward generation. Similarity is revealed by the parameter vector that is inferred over time. In contrast, we use the similarity over edges to determine the "cut" which in-turns guides the action selection process in adversarial settings. Their work extends to running a contextual bandit for every node, whereas ours is a single bandit algorithm, where the context information is captured in the "cut". The work of Di Castro et al. (2011) on edge bandits is similar in the sense that the bandits lie on the edges. However, instead of direct rewards of action selection, rewards are a difference in the values of the vertices. Further, this is the stochastic setting instead of the adversarial one. In Spectral bandits (Valko et al., 2014), the actions are the nodes, while there is a smooth Laplacian graph function for the rewards. We discuss later the limitations of Laplacian based methods for graph labelling. Further, they do not consider the Ising model that we study.

As we have seen before in Chapter 2, the seminal work of semi supervised graph labelling prediction can be found in (Blum and Chawla, 2001), where minimum label-separating cut is used for prediction. Laplacian based methods that result in neighbouring nodes connected by an edge to share similar values are widely studied in the semi-supervised and manifold learning problems (Zhu and Ghahramani, 2002; Zhu et al., 2003; Herbster, 2008; Herbster and Lever, 2009; Belkin and Niyogi, 2004). Typically, this information is captured by the semi-norm induced by the Laplacian of the graph. Essentially, the smoothness of the labelling is ensured by the "cut". The "cut" is the number of edges with disagreeing labels. Then, the norm induced by the Laplacian can be considered as the regularizer. However, there are limitations in these methods with increasing unlabelled data (Alamgir and von Luxburg, 2011; Nadler et al., 2009). Here, once agin, we also use "cut" as the regularization measure over an Ising model distribution of the values over the vertices of the graph at low temperatures for sequential action selection.

On finding the minimum cut labelling, we consequently find the active partition given the current context and then sample the actions from the relevant partition.

5.5 Ising Model at Low Temperature

As discussed above, the labelling of the whole graph is obtained by optimizing the objective function constrained on the given labels. From label propagation (Zhu et al., 2003), we saw when $p = 2$, the harmonic energy function $E(\boldsymbol{u})$ minimized in (5.1) is quadratic in nature. The technique in (5.1), chooses the label as a function $u : V \to \mathbb{R}$ and a probability distribution on the function $\boldsymbol{u}$ given by a Gaussian field $P(\boldsymbol{u}) = \frac{\exp^{-\beta E(\boldsymbol{u})}}{Z}$, where Z is the partition function and β in the inverse temperature or the uncertainty in the model as we have seen in Chapter 4. There are multiple limitations of the quadratic energy minimization technique. This model is not applicable for $p \to 1$ in the limit. Not only is the computation slow, the mistake bounds obtained are not the best. Further, in our problem, we relax the values of the labels such that $u : V \to [0, 1]$. With $p \to 1$, the energy function is equivalent to the the one that finds the minimum cut. Further, when $p \to 1$ using (5.2) results in the minimization of a non-strongly convex function per trial that is not differentiable. Further, it is known that the Laplacian based methods are limited with the abundance of unlabelled data (Nadler et al., 2009). Hence, we are interested in the Markov random field that is applicable here when $p \to 1$, with discrete states also known as the Ising model. At low temperatures, the Ising probability distribution over the labellings of a graph $\mathcal{G}$ is defined by:

$$P_T^{\mathcal{G}}(\boldsymbol{u}) \propto \exp\left(-\frac{1}{T}\psi_G(\boldsymbol{u})\right) \quad . \tag{5.3}$$

where T is the temperature, $\boldsymbol{u}$ is the labelling over the vertices of $\mathcal{G}$ and $\psi_{\mathcal{G}}(\boldsymbol{u})$ is the complexity of the labelling or the "cut-size". The probabilistic Ising model encodes the uncertainty about the labels of the vertices and at low temperatures favours labellings that minimise the number of edges whose vertices have different labels as shown in (5.2) with $p = 1$. If the vertex labels pairs seen so far is given by Z_t of vertex label pairs $(j_1, y_1), \ldots, (j_t, y_t)$ such that $(j, y) \in V(\mathcal{G}) \times \{0, 1\}$, then the marginal probability of the label of the vertex v being y conditioned on Z_t is given by: $P_T^{\mathcal{G}}(u_v = y|Z_t) = P_T^{\mathcal{G}}(u_v = y|u_{j_1} = y_1, \ldots, u_{j_t} = y_t)$. At low temperatures and in the limit of zero temperature $T \to 0$, the marginal favours the labelling that is consistent with the labelling seen so far and the minimum cut. Such label conditioning or label consistency in the context of graph labelling has been extensively studied (Herbster et al., 2005; Herbster and Lever, 2009; Herbster, 2008). In this chapter, we are only interested in the low temperature setting of the Ising model as the environment in which the player functions. However, at low temperatures, the minimum cut is still not unique.

Figure 5.1, shows an interesting graph from the literature used by Herbster et al. (2009), that is called the d-octopus graph formed of path graphs (tentacles) of length d, where there are $d+1$ vertices on each path graph. The paths adjoin a common vertex that is connected to a single head vertex. It is a good example of simple path graphs linked to form an interesting pattern; it is much easier to analyse the semi supervised graph labelling on a path graph. The goal is to run the vanilla harmonic energy minimization or the quadratic semi-norm interpolation with $p \to 2$ from Section 5.2.1, comparing it to optimized linear maximum flow algorithm for the case when $p \to 1$. The results in Figure 5.2, is the labelling resulting from the optimization of the quadratic formulation of $p \to 2$ and linear $p \to 1$ respectively in (a) and (b). With $p \to 2$, the minimum energy labelling obtained has a "cut" of 3 whereas with $p \to 1$, a much lower energy configuration is obtained giving a cut of 1 for the same setting.

5.6 Multi-Armed Bandit Problem (MAB)

In Chapter 2, we provided a detailed background on online learning. MAB is the framework of online learning with minimum feedback. This is the worst case setting and a much harder problem to solve. As with any sequential prediction game, the MAB is played between the learner and the environment and proceeds in a series of rounds $t = 1, \ldots, n$. At every time instance t, the forecaster chooses an action I_t from the set of actions or arms $a_t \in \mathcal{A}$, where $\mathcal{A}$ is the action set with K actions. When sampling an arm, the learner suffers a loss l_t that the adversary chooses in a randomized way. The forecaster receives the loss for the selected action only in the bandit setting. The objective of the forecaster is to minimize the regret given by the difference between the incurred cumulative loss on the sequence played and the optimal cumulative loss with respect to the best possible action in hindsight. The decision making process depends on the history of actions sampled and losses received up until time $t-1$. The notion of regret is expressed as expected (average) regret and pseudo regret, where pseudo regret is the weaker notion because of the comparison with the optimal action in expectation. For the adversarial case, it is given by:

$$\overline{R_n} = \mathbb{E}\sum_{t=1}^{n} l_{I_t,t} - \min_{i=1,\ldots,K} \mathbb{E}\sum_{t=1}^{n} l_{i,t} \ . \tag{5.4}$$

The expectation is with respect to the forecaster's internal randomization and possibly the adversary's randomization. In this work, we consider the adversarial bandit setting with side information (information at queried vertex). Note that unlike in the standard MAB problems where there is no structure defined over the actions, in our setup of the problem, we not only have a structure over the action set but also potentially utilize the associated structural side information that makes the problem more realistic. One more deviation from the standard MAB framework is that at every round, the adversary

randomly selects a vertex as the current concept; the value of the concept queried is unknown until after the trial and action selection. Further, our adversary is restricted in that the complexity or "cut-size" of the model of the environment in that it cannot increase the "cut-size" across trials. The intuition being, the number of times the learner makes a mistake (predicts the queried state wrong) or does not choose the optimal action, is bounded by the number of times the "cut" changes for the minimum.

In practice, the problems in the context of semi-supervised learning are not hard whereas the bandit algorithms are much harder problems. Here, the hardness of the bandit feedback is relaxed in the form of the side information. The side information behaves as the context and provide latent information about the state of the graph; some actions are more relevant to the current context than others. As far as the feedback on the action selected is concerned, the feedback is still bandit (only reward of the selection action revealed). The intuition is the bandit algorithm functions within a contextual graph labelling algorithm. Since, we do not make any assumptions on how the rewards are being generated; the rewards do not come from any distribution, the adversary is still adversarial. Although, our side labels come from an underlying Ising distribution over the vertices; they do not directly contribute to the reward of the action selected.

Another important point to note here is that we assume that the adversary is regular. This means that the goal of the algorithm in trying to minimize the number of mistakes is not going to get sabotaged by the adversary trying the opposite (adaptive adversary). This allows for the learner to have hope and potentially succeed. The only restriction on the adversary is that, the minimum cut cannot be increased.

5.6.1 Preliminaries

We consider an undirected graph $\mathcal{G} = (V(\mathcal{G}), E(\mathcal{G}))$ where the elements of E are called edges that form an unordered pair between the unique elements of V that are called vertices. We assume a unit weight on every edge. The number of vertices in the graph are denoted by N. The vertices of the graph are associated with partially unknown concept values or labels s_i that are gradually revealed, while the bandits lie on the edges in $E(\mathcal{G})$ to form the action set $\mathcal{A}$ with cardinality $|K|$. We assume a κ connected graph, where the maximum value of κ is such that each vertex has at least κ neighbours. Vertices i and j are neighbours if there is an edge or action connecting them. Note, the number of rounds $n \leq |K|$. In our case, n is equal to number of vertices queried by the environment with unknown labels. A vertex is randomly selected by the environment at every round t, where the queried vertex is given by x_i where $i \in \mathbb{N}_N$. The queried vertex could represent the request to place an advert on the product website the user currently visits. More specifically, the connections in our graph not only capture the explicit connections between vertices given by locality, but our bandits or edges also capture the

implicit connections between the values of vertices that are possibly differently labelled. The labels are relaxed such that the label for the i-th vertex is denoted by $s_i = \{-1, 1\}$.

At the start we are given the labels of a small subset of observed vertices, $s^o \in \mathcal{S}^o \subset V(\mathcal{G})$. The labels of the unlabelled vertices $s^u \in \mathcal{S}^u \subset V(\mathcal{G}) \setminus \mathcal{L}$, with $S = S^o \cup S^u$ are revealed sequentially at the end of each round as side information. We assume that there are at least two vertices labelled at the start, one in each category. The learning algorithm plays the online bandit game where the adversary at each trial reveals the loss of the selected action and the label of a randomly selected vertex. The goal of the learner is to be able to predict the label of the randomly selected vertex and then sample the appropriate action given the prediction.

5.7 Maximum Flow Computation

Given a partially labelled graph, the Ising model associates a probability with every labelling that is a consistent completion of the partial labellings. Now, if "cut" defines the "energy" of the labelling, then the $low-temperature$ Ising is a simplified landscape made up of all such minimum cut (energy) labellings. In a way, the Ising model induces an "energy landscape" over the labellings via the "cut." For a n-vertex graph, the energy levels sit inside the n-dimensional hypercube. One can minimize the energy while being consistent with the observations seen so far to achieve the desired goal.

As a first step in the learning process, the learner has to detect the underlying hidden partition in the graph, given the available labels with respect to the currently queried vertex. It can do so by using efficient graph partitioning methods. However, given the partial labelling, the partition detected should respect or be consistent with the labels seen so far. One way to address this is by using optimization methods that satisfy the label consistency through constraints. Alternatively, there are very efficient linear time exact methods that can solve this in practice. One such method is "Ford-Fulkerson" algorithm (Ford and Fulkerson, 1956). If one can characterize the labelled vertices in such a way that designates a single source, single sink network, running (Ford and Fulkerson, 1956) in an online fashion for every round using the side information can be used to efficiently detect the partition. Here, we choose to use a simplified linear programming relaxation to the classic Linear Programming (LP) maximal flow problem (5.5). Although, the LP formulation used here can be solved in polynomial time, there is nothing restricting us in using the linear time modified "Ford-Fulkerson" algorithm to achieve the same goal. The objective here is to enable the learner to make better predictions and hence lower its regret quicker by detecting the partition early, rather than to illustrate the computational efficiency of the method.

It is known by Menger's theorem of linear programming duality, that maximum flow and the minimum cut are related given a source and a target vertex. Let us introduce

the maximum flow or label consistent minimum cut in the graph using the following notation $c^* = \min\{\mathbf{S} \in \{-1,1\}^N : \psi_{\mathcal{G}}(\mathbf{S}|\mathcal{H})\}$ consistent with the trial sequence $\mathcal{H}$ seen so far.

$$E(\mathbf{S}) = \underset{\mathbf{S}\in\{-1,1\}^N}{\operatorname{argmin}} \sum_{(i,j)\in E(\mathcal{G})} |s_i - s_j| \leq c^* \ . \tag{5.5}$$

In general, linear programming relaxations are much easier to analyse. Interested readers are referred to the article (Trevisan, 2011), where LP relaxations are discussed. We use a linear programming relaxation of the above objective as shown in the algorithm outlined in Figure 5.3 that has auxiliary variables introduced such that there is one variable for every vertex v and one variable for every edge f_{ij}. Since we have an undirected graph, we assume a directed edge in each direction, for every undirected edge. Hence we have two flow variables per edge in the graph. Essentially, the free variables in the optimization are the unlabelled vertices s_i^u, s_j^u and the flows across every direction f_{ij}. The total flow across all the edges will be our maximum flow for this low temperature Ising model. The formulation in Figure 5.3 below is what the learner follows to find the minimum cut $\psi_{\mathcal{G}}$. The output from the computation is a directed graph with the value of flow at every edge and the labelling of the vertices consistent with the labels seen so far; $w_{(i,j)}$ is the cost variable of the LP. The sum of the flows is the maximum flow in the Ising model at low temperatures. We fix one of the labelled vertices as a source and one as target, each with different labels. We assume a unit capacity on every edge. The constraints in Figure 5.3 ensure the capacity constraint $f_{(ij)}$ and conservation constraint $s_i - s_j$are adhered to i.e. the flow in any vertex v other than the source and target, is equal to flow out from v. The largest amount of flow that can pass through any edge is at most 1, as we have unit capacity on every edge. We know that the cost of the maximum flow is equal to the capacity of the minimum cut. The minimum cut obtained as a solution to the optimization problem is an integer.

5.7.1 Playing Ising Bandits

Figure 5.4 describes the main algorithm for Ising bandits. It is important to note that `ComputeMaxFlow` can only guide the player towards the active partition with respect to the current context (queried vertex) by detecting the partition early on. $\mathcal{P}$ is a subgraph of $\mathcal{G}$, $\mathcal{P} \subseteq \mathcal{G}$ iff $V(\mathcal{P}) \subseteq V(\mathcal{G})$ and $E(\mathcal{P}) = \{(i,j) : i,j \in V(\mathcal{P}), (i,j) \in E(\mathcal{G})\}$. `SelectPartition` samples the Ising bandits from the best partition with respect to the active concept if the minimum cut changes from previous round. $E(\mathcal{R}), E(\mathcal{J})$ are the partitions of the action set at trial t. Since $\mathbf{S}'$ provides the labelling, it is easy to see which bandits fall in which partition with respect to x_t . The probability distribution r_t over $E(\mathcal{R})$, and j_t over $E(\mathcal{J})$ sum to p_t. Note that if the cut remains the same, the player keeps playing the same partition until the cut changes. This has an important implication. Since we assume that the adversary cannot increase the cut at any trial,

ComputeMaxFlow (Input parameters *target vertex:* $s_{\sqcup}$; *source vertex:* $s_{\sqcap}$; *trial sequence:* $\mathcal{H} = (x_k, s_k)_{k=1}^{t}$; *graph:* $\mathcal{G}$)

$$minimize \sum_{(i,j)\in E(\mathcal{G})} w(i,j)f(i,j)$$

subject to:

$$f_{(i,j)} \geq 0 \tag{5.6}$$

$$s_i - s_j \leq f_{(ij)} \tag{5.7}$$

$$s_i \geq -1 \tag{5.8}$$

$$s_i \leq 1$$

Return: *min-cut:* c^*; *flows:* f; *consistent partition:* $\mathbf{S}^{'}$

Figure 5.3: Computing the Max-flow.

Parameters: Graph: $\mathcal{G}$; $\eta \in \mathbb{R}^+$

Input: Trial Sequence: $\mathcal{H} = \langle (x_1, -1), (x_2, 1), (x_3, s_3), \ldots, (x_t, s_t) \rangle$

Initialization: p_1 is the initial distribution over $\mathcal{A}$ such that, $p_1 = (\frac{1}{|K|}, \frac{1}{|K|}, \ldots, \frac{1}{|K|})$, Initial cut-size $c = \infty$; active partition distribution $r_1 = p_1$

for $t = 1, \ldots, n$ **do**

Receive: $x_t \in \mathbb{N}_N$

$(c^*, f, \mathbf{S}^{'}) = \texttt{ComputeMaxFlow}(s_{\sqcup}, s_{\sqcap}, \mathcal{H}, \mathcal{G})$

if $(c \neq c^*)$ **then** % if cut has changed

$(E(\mathcal{R}), E(\mathcal{J}), r_t, j_t) = \texttt{SelectPartition}(x_t, p_t, \mathbf{S}^{'}, \mathcal{A})$

Assign: q_t be the distribution over Ising bandits w.r.t x_t, such that, $\sum_{i=1}^{|E(\mathcal{R})|} q_{i,t} = r_t$. For any t, $p_t = r_t \cup j_t$

Play: I_t from q_t

Receive: Loss z_t; side information s_t

Compute: Estimated loss $\tilde{z}_{i,t} = \frac{z_{i,t}}{q_{i,t}} \mathbb{1}_{I_t=i}$

Cumulative estimated loss: $\tilde{Z}_{i,t} = \tilde{Z}_{i,t} + \tilde{z_{i,t}}$

Update: $q_{i,t+1} = \frac{q_{i,t} \exp(\eta \tilde{Z}_{i,t})}{\sum_{j=1}^{|E(\mathcal{R})|} \exp(\eta \tilde{Z}_{j,t})}$

end

Figure 5.4: Ising Bandits Algorithm.

the cut can only decrease or stay the same. For the rounds it stays the same, the regret that the player suffers is well bounded by the number of times the cut changes. In the best case, the algorithm behaves as a typical Multi-armed bandit (MAB) and in the worst case when the partition changes at every round, the algorithm plays the modified `Ising Bandits`. The algorithm parameter η is the standard MAB value $\eta = \sqrt{\frac{\log |K|}{3n}}$.

5.8 Experiments

The experimental evaluation compares three competitor algorithms with our algorithm `IsingBandits`. The three algorithms are `LabProp` (Zhu and Ghahramani, 2002; Zhu et al., 2003), `Exp3` (Auer et al., 1995) and `Exp4` (Auer et al., 1995). `Exp3` and `Exp4` are from the same family of algorithms for bandits in the adversarial setting. `Exp4` is the contextual bandit setting, the close competitor to `Ising` from the contextual perspective. The experts or contexts in `Exp4` for our problem setting are a number of possible labellings. Note that the number of experts selected for prediction have a bearing on the performance of the algorithm. In our experiments, we fixed the number of experts to 10. In reality, even at low temperatures for the model we consider, the set of all possible labellings is exponential in size. `LabProp` (Zhu and Ghahramani, 2002; Zhu et al., 2003) is the implementation where the state-of-the-art graph Laplacian based labelling procedure is used to optimize the labelling consistent with the labels seen so far. For all of the above algorithms, we use our own implementation in MATLAB. Since online experiments are extremely time consuming while processing one data point at a time, we have averaged each set of experiments over five trials but for `ISOLET`, where we average over ten trials. The datasets that we use are the standardized UCI (Lichman, 2013) datasets as before, namely the `USPS` and the `ISOLET` datasets. All datasets are nearly balanced in our experiments to demonstrate the fairness of the class distribution and for avoiding any majority vote cases where the class with the majority vote wins.

5.8.1 Dataset Description

We will reiterate over the datasets for the sake of completion. The summary of datasets used is captured in Table 5.1. The `USPS` handwritten digits is an optical character recognition dataset comprising 16x16 grayscale images of “0” through “9” obtained from scanning handwritten digits. The pre-processed dataset has each image with 256 real valued features without missing values scaled to [-1,1]. We randomly sample the examples for the graph from the 7291 original training points. Each vertex in the graph thus sampled is a digit. We perform experiments for several binary classification tasks. The graph generation process involves sampling instances of one digit vs. the other digit to form our underlying graph with edges or connections between the two digits forming our action set.

Table 5.1: Datasets used in Ising Bandits.

Data set	#Instances	#Features	#Classes
USPS	7291	256	10
Isolet	7797	617	26

We use a noisy perceptual dataset for spoken letter recognition called `ISOLET`, consisting of 7797 instances with 617 real valued features. A total of 150 subjects spoke each letter of the English alphabets twice resulting in 52 training examples from each speaker. The 150 speakers are split into 5 groups of 30 speakers each, named as Isolet 1 through to Isolet 5. For the purpose of the experiments here, we build the graph from Isolet 1 comprising 1560 examples from 30 speakers, with each letter being spoken twice. As before, we are interested in binary classification graphs where we sample the first 13 and the last 13 spoken letters as two separate underlying concepts in the graph, the connections between which form the action set.

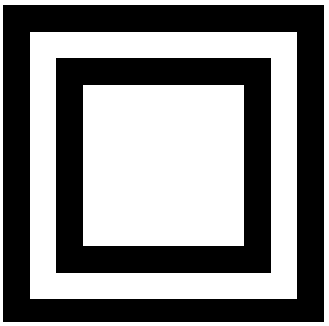


Figure 5.5: Squares image.

5.8.2 Synthetic Dataset

The synthetic data uses a 2D grid like topology. Figure 5.5 shows the image used to construct the graph in our experiments. Our interest in using this image for our simulation experiment, stems from the natural occurring graph structure in such 2D grids. The image style of `Squares` is chosen based on our interest in smooth and wide regions of similar labels interspersed with dissimilar labelled boundary regions. A square image constructed using a set of pixels is used, where each pixel has an intensity of either 0 or 1. The 0 and 1 intensities are balanced across the pixels i.e. there are equal number of pixels with 0 and 1 intensities. Each pixel in the image corresponds to a vertex in the graph and the intensities correspond to the label or class of the vertex. Here, our graph has 3600 vertices. The neighbourhood system in the graph comprises edges connecting pairs of neighbouring similar pixels. The connectivity is typically guided by whether the pixels are of comparable intensities, if the pixels are structurally close to each other or both. Here, we are only interested in the physical pixel locations that are

used to determine connectivity i.e. pixels closer to each other on the grid are connected. The connections eventually form our bandits action set. In this chapter, we are only interested in undirected and unweighted graphs. Our grid graph thus generated has a weight of 1 on every edge and there is an edge in either direction. Further, we investigate the type of neighbourhood system, called torus. In the torus grid, each pixel has four neighbours; achieved by connecting the top with the bottom edge pixels and the left with the right edge pixels. The graph used is the same across trials. We randomly sample the available labelled vertices from the graph such that there are equal number of labels from each concept class.

5.8.3 Graph Generation from Datasets

We design our experiments to test the action selection algorithm under a number of different criteria of graph creation: balanced labels, varying degree of connectedness, varying sizes of initial labels and noise. The parameters that are varied across the experiments are graph size indicated by N, labels available as L, connectivity K, noise levels nse.

In the set of experiments with `ISOLET`, we chose to build the graph from the first 30 speakers in Isolet1 that forms a graph of 1560 vertices of 52 spoken letters (each letter spoken twice) by 30 speakers. The concept classes that are sampled are the first 13 letters of English alphabets as one concept vs. the next 13 letters as the other concept. We build a 3 nearest neighbour graph from the Euclidean distance matrix constructed using the pairwise distances between the examples (spoken letters). In order to ensure that the graph is connected for such low connectivity, we sample a minimum spanning tree (MST) for each graph and always maintain the MST edges in the graph. The MST uses the Euclidean distances as weights. The same underlying graph is used across trials. The edges or connections form the bandits. The available side information is sampled randomly such that the two classes are balanced over the entire graph size.

In the USPS experiments, we randomly sample a different graph for each trial. While sampling the vertices of the graph, we ensure to select vertices equally from each concept class. We use a variety of concept classes 1 vs. 2, 2 vs. 3 and 4 vs. 7. We use the pairwise Euclidean distance as the weights for the MST construction. All the sampled graphs maintain the MST edges. In all the experiments on the datasets, the unweighted minimum spanning tree (MST) and $K = 3$ or 3 -NN graph had their edge sets' "unioned" to create the resultant graph. The motivating reason being that most empirical experiments had shown competitive performance of algorithms at $K = 3$, while the MST guaranteed connectivity in the graph. Further, MST based graphs are sparse in general, enabling computational efficient completion of the experiments. All the experiments were carried out on a quad-core processor notebook (@2.30 GHz each) with 8GB RAM and 16 GB RAM.

5.8.4 Evaluation Criteria

We measure the performance of the algorithms by means of the instantaneous regret or per-round regret of the learning algorithm as compared with the optimal algorithm (lower the better). Instantaneous regret at trial t is the cumulative regret $\overline{R_n}$ in Equation 5.4 divided by t. The instantaneous regret should sublinearly reduce to zero and in the evaluation of the algorithm it is measured against time. In our case, time indicates each unlabelled vertex queried in an iterative fashion by the environment, until all unlabelled vertices have been queried. Ideally, the more vertices that have been queried and side information obtained, the less is the instantaneous regret of the algorithms. In all the experiments, the concept class distribution in the underlying graph is balanced.

5.8.5 Results

In the synthetic dataset of concentric squares experiment, shown in Figure 5.6, `Ising` always outperforms `Exp3`, `Exp4` and `LabProp`. `LabProp` and `Ising` are very competitive over uninformed competitors of `Exp3`, `Exp4`. `Exp3`, `Exp4` do not use the available side information to sample their actions. Note, the overlapping squares is a difficult dataset where closely connected clusters of similar labels `white with intensity 1` are surrounded by clusters of opposite labels `black with intensity 0`, around its boundary. Although `LabProp` is good at exploiting connectivity, here we see that `Ising` captures the opposing boundary side information better than `LabProp`. The dataset experiments be-

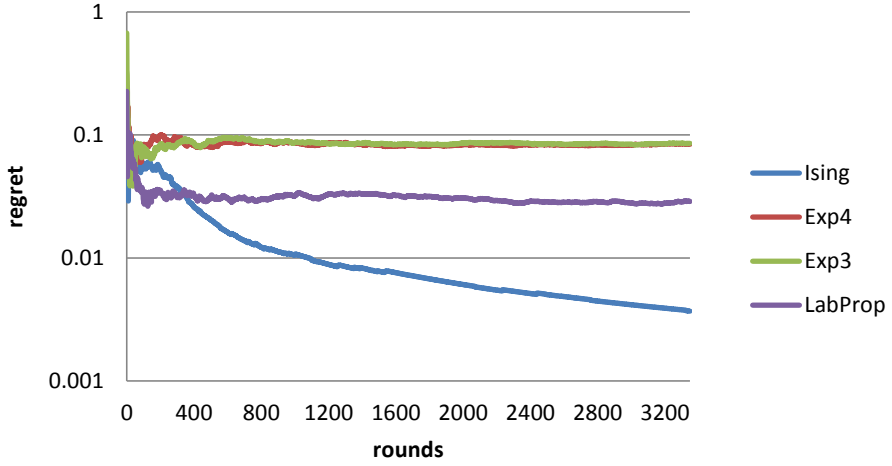


Figure 5.6: Results on torus graph generated from `Squares` image with equal number of neighbours $K = 4$, $N = 3600, L = 250$.

gin with the `USPS 2 Vs.3` experiment with connectivity $K = 3$, available labels $L = 8$, and number of data points $N = 1000$. In Figure 5.7 below, algorithms `Ising` and `LabProp` are very competitive when the side information obtained is about the size of more than half of the dataset. When the side information is very limited at the beginning of the game, `LabProp` outperforms `Ising`. In Figure 5.9 below, we test the behaviour

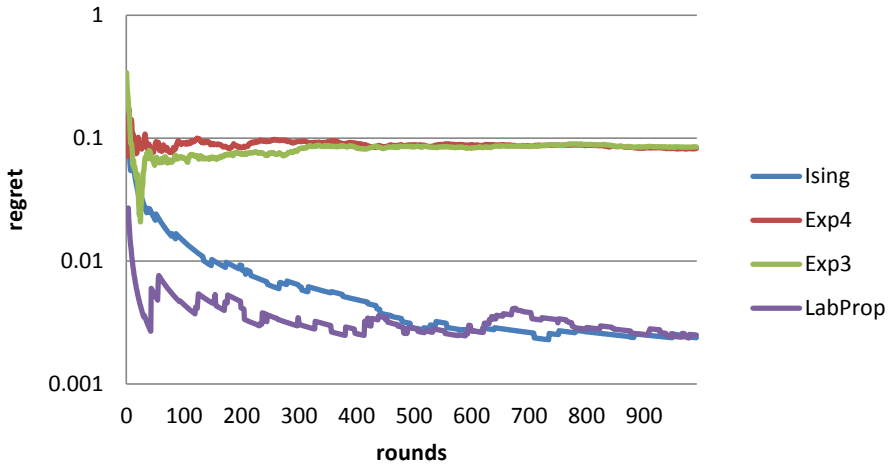


Figure 5.7: `USPS 2 Vs.3 with` $K = 3, N = 1000, L = 8$.

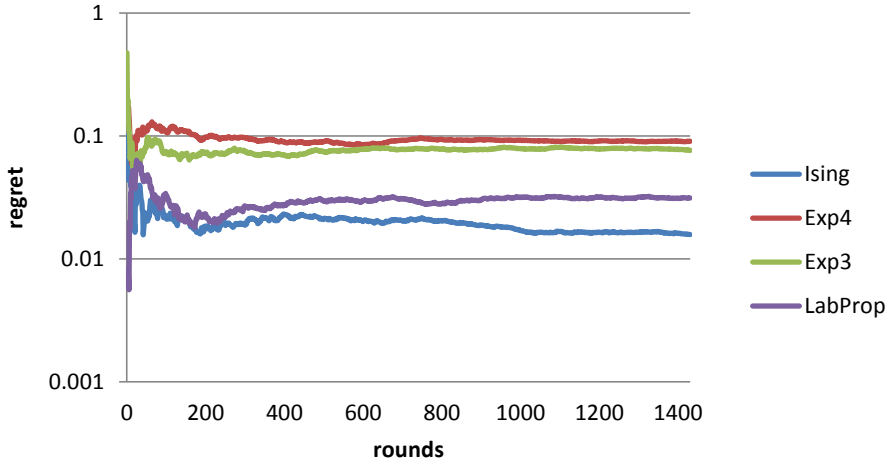
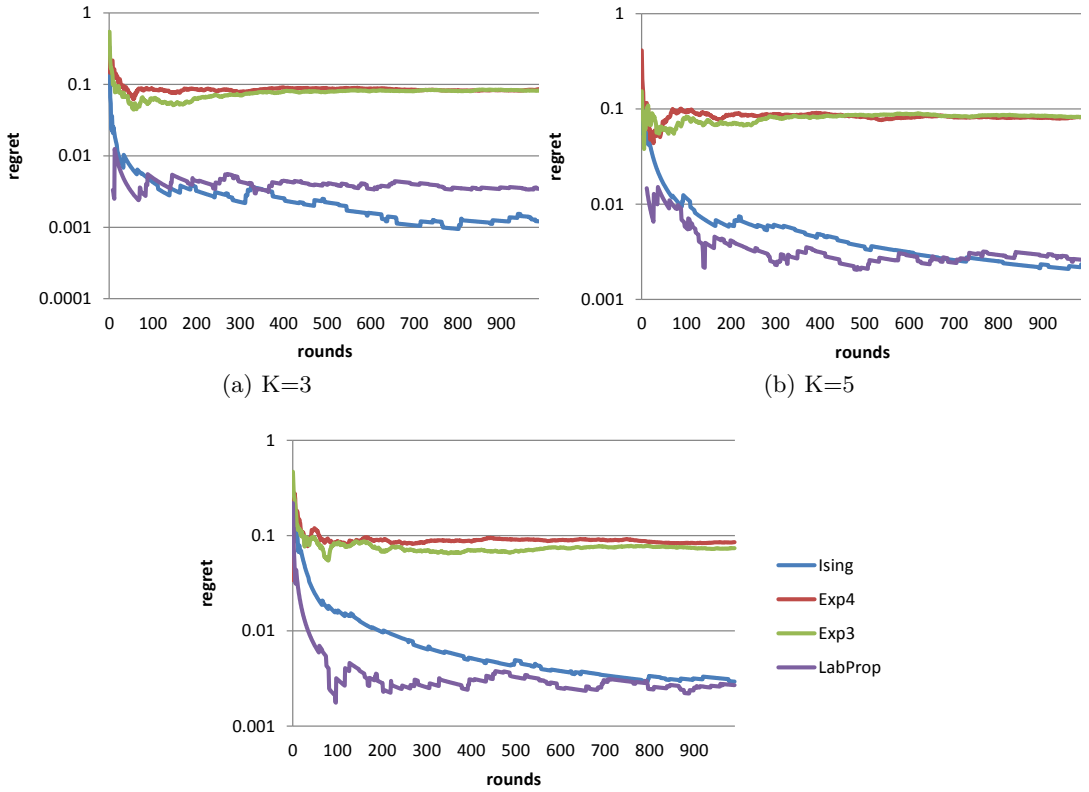


Figure 5.8: Experiments on `ISOLET` with $K = 3, N = 1560, L = 128$.

of the algorithms on `USPS 4 Vs.7` with varying degree of connectivity. We vary the parameter K over a range to check how well the cluster size affects the performance. It is known from labelling over graph literature that with increasing K the behaviour deteriorates. Here, we see `Ising` outperforms `LabProp` for lower values of K, while `LabProp` wins for higher K. In our experiments over the dataset `ISOLET`, we sample the graph from `ISOLET 1`. In Figure 5.8, we observe that with $K = 3$ and $L = 128$, `Ising` outperforms `LabProp` throughout. The overall regret achieved in `ISOLET` is higher than the regret achieved in `USPS` as `ISOLET` is a noisy dataset.

The following set of experiments in Figure 5.10 test the robustness of our methods in presence of balanced noise. Our noise parameter *nse* is varied over the range $s = 10, 20, 30, 40$. When noise is say x percent, we randomly eliminate the actions or edges in the graph (from existing connections) for which the noise is less than x percent, and add a balanced equal number of new actions (connections) to the graph. We see that the performance of `Ising` is the most robust across various noise levels. `LabProp` suffers with noise as it is heavily dependant on connectivity, and under performs in contrast



(a) K=3 (b) K=5

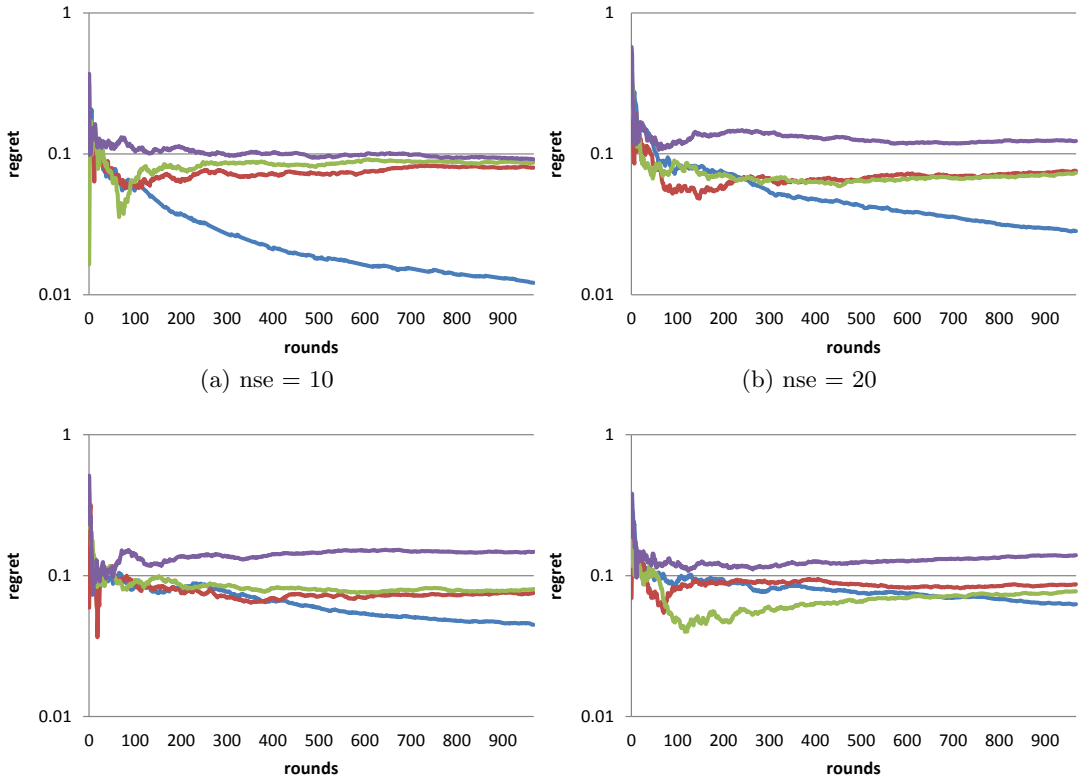
(c) K=7

Figure 5.9: `USPS 4 Vs.7`, with varying connectivity $K = 3, K = 5, K = 7$ on randomly sampled graphs with $N = 1000, L = 8$. The color coding is uniform over all the graphs and as indicated in (c) above.

to `Exp4` and `Exp3`. On the contrary, `Ising` uses the connectivity for side information, with its action selection unaffected with the introduction of noise. When the noise level increases, the performance of all the algorithms decrease uniformly.

5.9 Discussion and Conclusion

There are real life scenarios where a core minimal subset of connections in a network is responsible for partitioning the graph. Such a core group could be a focus of targeted advertising or content-recommendation, as that can have maximum influence on the network with a potential to go viral. Typically, there is a lot of available information in such settings that is potentially usable for detecting the changing partitioning set. In this chapter, we addressed such advertising and content recommendation challenges by casting the problem as an online Ising graph model of bandits with side information. We used the notion of "cut-size" as a regularity measure in the model to identify the partition and play the bandits game. The best case behaviour of the algorithm when there is a single partition is equivalent to the standard adversarial MAB. We showed a polynomial algorithm where the label consistent "cut-size" can guide the sampling



(a) nse = 10 (b) nse = 20

(c) nse = 30 (d) nse = 40

Figure 5.10: `USPS 1 Vs. 2` Robustness Experiments with noise levels 10%, 20%, 30% and 40%. Blue is Ising, red is Exp4, green is Exp3 and purple is LabProp.

procedure. Further, we motivated a linear time exact algorithm for computing the max flow that also respects the label consistency. An interesting effect of the algorithm is that as long as the *cut-size* does not change, the learner keeps playing the same partition on the active action set (size smaller than the actual action set). The regret is then bounded by the number of times the cut changes during the entire game. This can be proven analytically, which we would like to pursue as future work.

Chapter 6

Online Experts for Multiple Objectives

6.1 Introduction

This chapter extends the work in the second part of the thesis in analysing online algorithms with varying feedback. In this chapter, we focus on a specific real-life application of efficient power consumption on low powered devices. Since online learning are adaptive techniques, that can learn from the data on the fly; they are the most likely choice for low powered environments.

Here, we study the full feedback setting of that of the experts unlike the limited feedback setting of the bandits in the previous chapter. Full feedback is an easier problem in online learning where the losses suffered by all of the options in the previous round are known before playing the option in the current round.

6.2 Background

Power is an expensive resource in embedded devices like mobile phones, that exhibit increased computational capacity. There are rising user demands for being able to execute high performing applications on those devices without decreasing the battery lifetime of the device. This presents a challenging multiple objective trade-off problem, that of maximum power or battery life savings with minimum performance or application runtime delay (Dhiman and Rosing, 2009). With increasing number of cores and hyper-threading capability in such devices, more battery usage and application performance settings are available (Esmaeilzadeh et al., 2012, 2011). With such a wide range of settings that the system can be configured to, it is difficult to determine the optimal power-performance trade-off configuration that would enable the right balance between

the objectives. Hyper-threading on cores complicates the power-performance challenges by means of making it extremely difficult to analyse the power-performance situation due to the parallel execution of applications with shared locks (Cochran et al., 2011; Meisner et al., 2011). Further, this problem is also exhibited in large scale-data centers with increased computational demands. These data centers have a hard power constraint which allows for operation of a certain number of server units. If the power constraint is exceeded, the circuit breaks and causes interruption (Cochran et al., 2011). Typically, the best power-runtime trade-off setting also depends on the workload (a workload indicates the applications executing together on the device or server at any point in time) and this setting may vary during execution of the workload (Cochran et al., 2011). Figure 6.1 below shows the optimal runtime points for various application workload settings under a range of possible power budgets as discussed in (Cochran et al., 2011). The figure shows how each representative application has its own optimal set of power-performance settings that lies on the red line also known as Pareto frontier. For each point on this frontier for each representative application, there is no other point that achieves lower energy and shorter performance runtime. These points are the non-dominated set of optimal settings that can be configured on the device when the particular application is under execution for maximal energy savings and minimum runtime delay. Each representative application of the workload has its own Pareto frontier shown in red. Since the power-performance objectives are conflicting in nature, improving one objective is only possible by compromising the other objective: Pareto nature. Thus the Pareto frontier or the optimal Pareto front contains the optimal runtime values as a function of the power usage values.

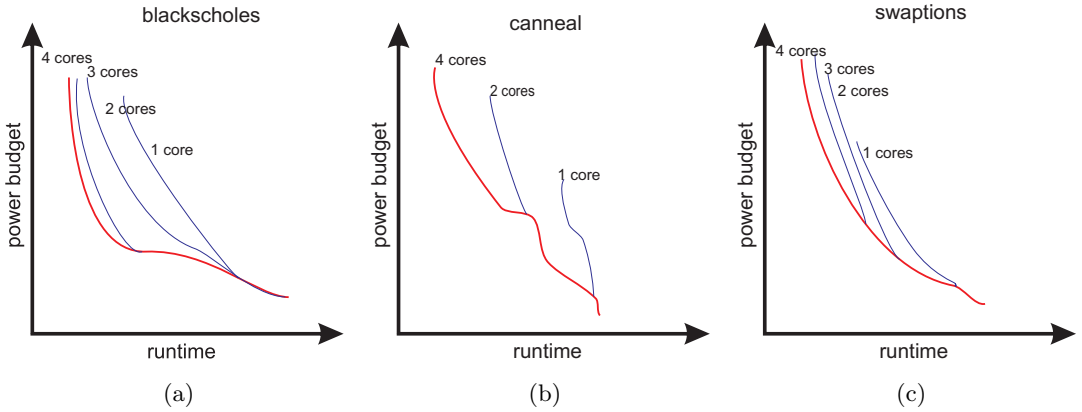


Figure 6.1: Pareto frontiers for runtime minimization under power budget. Red line is the Pareto frontier of optimal settings for different power budgets for representative applications (Cochran et al., 2011).

Here, we consider the problem of learning and predicting the optimal Pareto front of trade-off values for a sequence of workloads executing on the device or server. We believe that once the optimal Pareto front is identified for a sequence of workloads, this automatically finds the optimal Pareto front for each workload. Since these values depend on the

workloads that vary at runtime and that the environment is primarily non-deterministic, the learning happens in an on-line fashion. Also, as the the exhaustive set of power-performance settings, that the system can be possibly configured to are pre-determined, it makes sense to investigate the on-line learning with experts framework (Cesa-Bianchi and Lugosi, 2006) in this context, where the learner has no control on the individual expert's prediction.

6.3 Related Work

Multiple objective optimization is a well known field. Here, more than one objective are optimized while satisfying multiple constraints (Fliege and Svaiter, 2000). Methods used to solve these problems include the normal-boundary intersection method, normal constraint method, multiple runs of single-objective optimization for sampling the trade-off surfaces, or use of additional constraints (Singhee, 2011). Usually, the additional constraints these methods use, make the optimization problem more time consuming and hard or difficult to solve (Singhee, 2011). This makes such methods unsuitable for the application areas of low-powered embedded devices and power restricted data centers. Evolutionary algorithms have been used to deal with multi-criteria optimization problems where the fitness function evaluates non-domination by other solutions. Typically, these methods use stochastic optimization to approximate the Pareto set or the Pareto front (Bokrantz and Forsgren, 2011; Hu et al., 2003; Zitzler et al., 2004). However, these algorithms have very high computational costs due to lack of strong search directions and maintains many candidate solutions for the purpose of convergence. In our problem on embedded mobile devices, we cannot use these approaches again due to the high computational costs associated with them. Adaptive light-weight machine learning methods are good candidates for such problems. On related work in the on-line machine learning community, the tracking of the best expert problem (Herbster and Warmuth, 1998) deals with cases where the underlying true distribution changes with each subsequence or partition of data seen sequentially, for a limited number of times. Our work is different in that we track the Pareto optimal set of experts; experts that lie on the Pareto front for a sequence of observed workloads; their best expert is from the set of experts tracked over the whole sequence which is not necessarily Pareto optimal. Further, we do not know in advance how many switches or jumps as mentioned in their work are necessary to converge to the best expert for every partition seen. The recent work on on-line multi task learning (Lugosi et al., 2009) does not address conflicting objectives that the learner needs to address, rather it focuses on a tuple of actions that the decision maker selects. The other related work on the subset of experts in changing environments by Hazan et. al (Hazan and Seshadhri, 2009)- PAC subset selection of bandits (Kalyanakrishnan et al., 2012) and learning experts by Eban et. al (Eban et al.,

2012) are different from ours in that they work under different settings and assumptions of the environment.

6.4 Contributions

Our main contribution is in the formulation of the tasks of learning multiple objectives and predicting Pareto optimal trade-offs as an on-line learning with experts problem. Consequently, we show how the multiplicative weight update step of the learning algorithm can be significantly improved by exploiting Pareto descent directions that is obtained through a convex combination of the potential function of the regret of the learner with respect to each objective. We devise an importance vector $\boldsymbol{\alpha}$ that is obtained by sampling from a convex cone pointed at the origin of the vector space by exploiting the simultaneous linear inequality of the Blackwell condition (Blackwell, 1956, 1954). The convex combination of $\boldsymbol{\alpha}$ and the potential function of the regret of the learner with respect to each objective, enables the learner to converge to the optimal Pareto front while minimizing its regret with respect to each objective. In the following section, we introduce the definitions that are referred throughout the chapter.

6.5 Preliminaries

From the knowledge of multi-criteria optimization problems (Fliege and Svaiter, 2000; Harada and Kobayashi, 2006; Singhee, 2011) when a point is far from the local optimum, search directions can be found that simultaneously optimize the multiple objectives. However, as the point gets closer to the local optimum, the search direction cannot optimize both objectives together; the solutions thus obtained are Pareto optimal (Brown and Smith, 2003) and they are said to lie on the Pareto front. The Pareto optimal objective vector dominates all other objective vectors in the feasible space of solutions (Hu et al., 2003; Zitzler et al., 2004).

Definition 6.1. Dominance: An objective vector $\mathbf{y}_i$ is said to dominate another objective vector $\mathbf{y}_j$, ($\mathbf{y}_i \succ \mathbf{y}_j$), if no component of $\mathbf{y}_i$ is worse than the corresponding component in $\mathbf{y}_j$ and at least one component is better.

Definition 6.2. Non-Dominated and Pareto Optimal Sets: In the set of feasible solutions Y, if the set of solutions Y' is said to dominate every other solution in Y, then the set Y' is the non-dominated set or the Pareto-optimal set.

We reiterate the definition of descent directions and Pareto descent directions from the literature (Harada and Kobayashi, 2006) as follows:

Definition 6.3. Descent Directions: A descent direction is a direction that falls in the same half-space as the negative gradient of the function to minimize.

Definition 6.4. Pareto Descent Direction: The descent directions in which no other descent directions are superior in improving all objectives together are known as Pareto Descent directions. No Pareto descent direction is better than another Pareto descent direction and all Pareto descent directions are better than all descent directions.

Let T denote finite number of rounds and L denote finite number of experts designated as $\xi_1, \xi_2, \ldots, \xi_L$. The shorthand "expert i" is used to refer to expert ξ_i. Let the number of objectives that the learner has to address in the system be denoted by M. At every round t, the learner is given an input and is asked to predict the appropriate trade-offs for each of the M objectives using the experts predictions that the learner has access to. The prediction of the learner as well the true outcome at round t are both objective vectors of the type $\hat{\mathbf{y}}_t = (y_1, y_2, \ldots, y_M) \in \mathcal{M}, \mathcal{M} \in R^M$. The unknown sequence $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_T$ of elements form the true *outcome space* $\mathcal{Y}$. The learner's predictions $\hat{\mathbf{p}_1}, \hat{\mathbf{p}_2}, \ldots, \hat{\mathbf{p}_T}$ belong to decision space $\mathcal{D}$, which is assumed to be a convex subset of a vector space. The experts predictions at round t are given by the set $\{\mathbf{f}_{i,t} : i \in \xi\}$ where, $\mathbf{f}_{i,t} \in \mathcal{D}$. We assume that the experts' predictions belong to a feasible, objective space of vectors that lie on an exhaustive spread of convex Pareto curves. The learner has no knowledge of these underlying Pareto curves where the predictions of the experts (objective vectors) lie and its goal is to converge to the best possible sequence of experts that lie on the Pareto optimal front or true Pareto front. The learner assigns weights or beliefs in the experts given by the vector $\mathbf{w}_{i,t}^m = (w_1, \ldots, w_M) \in \mathbb{R}^M$ for the expert i; each component of the weight vector is the belief the learner has in the expert towards addressing each objective m. The notation of superscript "m" in any symbol is used to indicate which objective is being addressed by the relevant operator. Conversely, the subscript "m" indicates that the operator is involved in the summation over all objectives. The loss function used to measure the learner's loss is a convex loss function $\mathbf{l}$. The performance of the learner after T rounds is measured by the notion of regret given by:

$$\mathbf{R}_T^m = \sum_{t=1}^{T} (\mathbf{l}^m(\hat{\boldsymbol{p}}_t, \mathbf{y}_t) - \mathbf{l}^m(\mathbf{f}_{i,t}, \mathbf{y}_t)). \tag{6.1}$$

$\mathbf{l}$ is used to denote loss in any particular round and the cumulative loss is denoted by $\hat{\mathbf{L}_T}^m = \sum_{t=1}^{T} \mathbf{l}^m(\hat{\mathbf{p}}_t, \mathbf{y}_t)$, where $\hat{\mathbf{L}_t}$ is the learner's cumulative loss in round t and $\mathbf{L}_{i,t}$ is the expert i's cumulative loss in round t. The learning rate is denoted by η. All throughout, vector inequalities are to be understood component-wise. We deal with two objectives in our case as described in the learning algorithm below. The figure below shows the problem formulation set-up. The objective of the learning algorithm is to make predictions from the Pareto frontier as indicated in the figure.

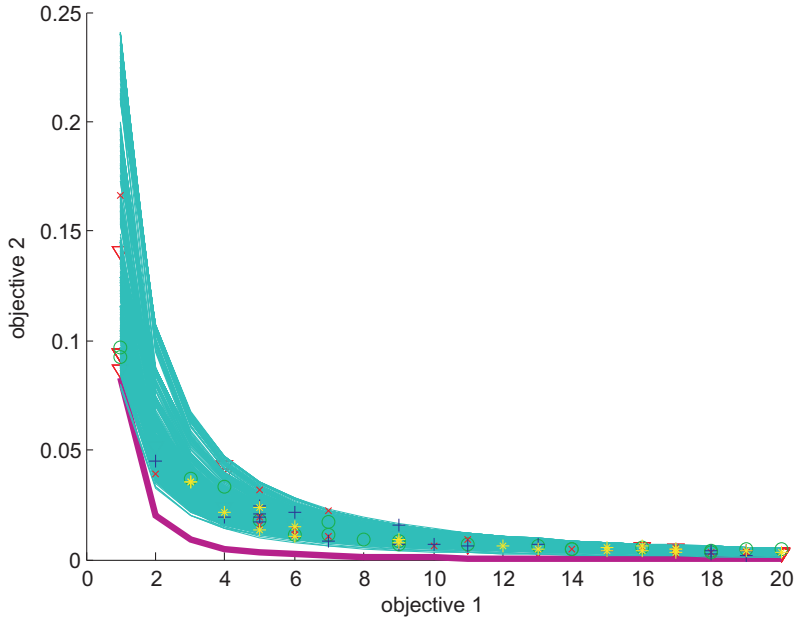


Figure 6.2: Pareto Descent Problem Formulation. The Pareto curves in Cyan are the curves spanning the objective space from which the experts predictions are drawn and unknown to the learner. The Pareto curve in Magenta is the optimal Pareto frontier that has optimal trade-off values for objective 1 and objective 2. All other symbols of +, *, o, x, ∇ are the predictions of various experts. The learner should find the optimal set of predictions coming from experts that lie on the Pareto frontier.

6.6 Online Experts

The goal of the learner in this set-up is to be able to predict trade-off values for the multiple objectives in the system such that the predictions are optimal. In our example problem discussed in previous sections, by optimal solutions, it is meant that the learner should be able to provide power-performance trade-off solutions such that no alternative solution achieves lower power and shorter runtime than this. The role of the learner then is to make predictions that lie on the Pareto front of solutions. The learner has no idea about the optimum Pareto front and has access only to the predictions of the experts, and the confidence or belief, the learner has in each of the experts. We have seen in the formulation of the problem, the performance of the learner in the on-line learning set-up is evaluated by the notion of regret which is the difference in cumulative loss of the learner and that of the best expert in hindsight as in (6.1), which should sublinearly reach zero as shown in (6.2) (Cesa-Bianchi and Lugosi, 2006), also known as Hannan Consistency.

$$\frac{1}{T}(\hat{\mathbf{L}_T}^m - \min_{i,\dots,L} \mathbf{L}_{i,T}^m) \overset{T\to\infty}{\to} 0. \tag{6.2}$$

For keeping the learner's regret as low as possible, the instantaneous regret vector is used which is given by $\mathbf{r}_t^m = (r_{1,t}^m, \ldots, r_{L,t}^m) \in \mathbb{R}^L, m \in \mathbb{R}^M$, M is the number of objectives. The instantaneous regret vector is the vector of regret values of each individual expert in that round, which shows how each expert performs with respect to the best expert in hindsight. The corresponding regret vector of the learner is then given by $\mathbf{R}_T^m = \sum_{t=1}^{T} \mathbf{r}_t^m$. Additionally, a monotonically increasing convex potential function is used as a function of the learner's regret $\phi^m(\mathbf{R})$ given by $\phi\colon \mathbb{R}^L \to \mathbb{R}, \Phi(\mathbf{u}^m) = \sum_{i=1}^{L} \phi(u_i^m)$, where $u = (u_1, \ldots, u_L) \in \mathbb{R}^L$ (Cesa-Bianchi and Lugosi, 2003) to measure the distance from origin of the learner's regret in a generalized way. With the help of the potential function it is obvious that the learner regret $\mathbf{R}_t$ at round t, will be kept close to the minimum of ϕ by the Blackwell condition.

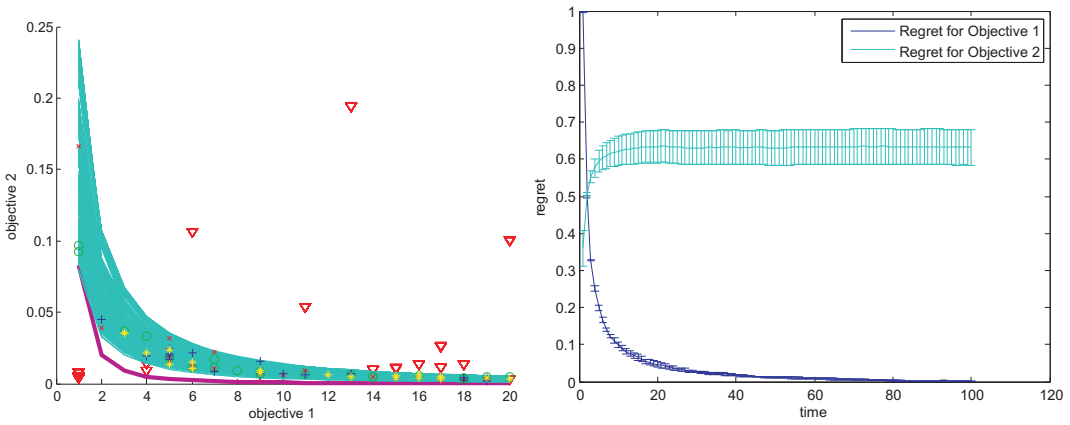
$$\sup_{y_t \in \mathcal{Y}} \mathbf{r}_t^m . \bigtriangledown \Phi(\mathbf{R}_{t-1}^m) \leq 0. \tag{6.3}$$

where $\mathbf{u.v}$ stands for the inner product of two vectors defined by $\mathbf{u.v} = u_1 v_1 + \cdots + u_N v_N$. In other words (6.3) implies that at round t, by keeping the regret vector to point away from the gradient of the potential function, the point $\mathbf{R}_t$ can be kept close to the minimum of Φ. The potential function is used to keep the drift or instantaneous regret vector in the same half-space as the negative gradient of the potential of learner's regret. Since the learner has no control on the predictions of the experts; it can only control the belief it has in each of the experts in every round and the strategy it uses for making its prediction. This suggests that the way to achieve (6.3), the belief or weights the learner assigns to each expert is crucial in keeping the regret of the learner low. Intuitively, in every round t, the weight of each of the expert up to previous round, $\mathbf{w}_{i,t-1}^m$, is related to the regret of the expert $\mathbf{R}_{i,t-1}^m$ as $\mathbf{w}_{i,t-1}^m = \phi'(\mathbf{R}_{i,t-1}^m)$ for the ith expert. A larger weight $\mathbf{w}_{i,t-1}$ is assigned to expert i, if $\mathbf{R}_{i,t-1}^m$ is large and vice versa (Cesa-Bianchi and Lugosi, 2006). The multiplicative weight update at every round t allows the selection of an instantaneous regret vector $\mathbf{r}_t^m$ for prediction that will keep its regret of the learner low in the next round. The prediction given by:

$$\hat{\boldsymbol{p}_t}^m = \frac{\sum_{i=1}^{L} \bigtriangledown \Phi(\mathbf{R}_{t-1}^m)_i \mathbf{f}_{i,t}}{\sum_{j=1}^{L} \bigtriangledown \Phi(\mathbf{R}_{t-1}^m)_j}. \tag{6.4}$$

6.6.1 Independent Weight Update (IWU)

This is the baseline method where the on-line learning framework is used without any modification to the weight update step of the experts (Cesa-Bianchi and Lugosi, 2006). The weights of the experts for each objective in the next round are independently updated based on the expert's performance in the previous round,



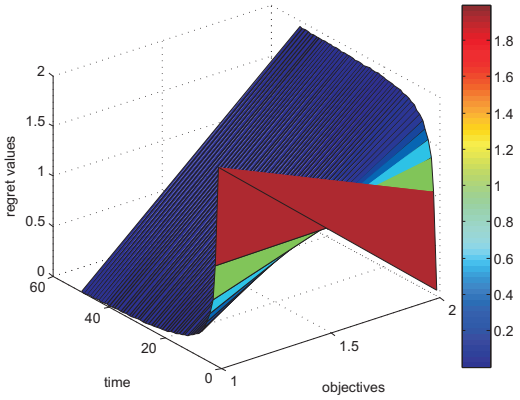


Figure 6.3: Results of IWU with Pareto problem formulation. (a) Predictions shown in ∇ made by IWU learner averaged over 1000 runs (b) Instantaneous regret over 100 runs (c) Regret surface over both objectives at the start.

$$\mathbf{w}_{i,t}^m = \frac{\mathbf{w}_{i,t-1}^m e^{-\eta \mathbf{l}^m(\mathbf{f}_{i,t},\mathbf{y}_t)}}{\sum_{j=1}^{L} \mathbf{w}_{j,t-1}^m e^{-\eta \mathbf{l}^m(\mathbf{f}_{j,t-1},\mathbf{y}_t)}}. \tag{6.5}$$

In our problem formulation however, as the objectives are conflicting in nature, updating the weight components of the weight vector $\mathbf{w}_t^m$ respective to each objective does not simultaneously decrease the regret of the learner for all objectives. This is because the learner has no information available in terms of how to relatively weigh each objective with respect to the other; the components of the weight vector $\mathbf{w}_{i,t}^m$ of expert i has information on how the expert fairs individually on the objectives, but does not relate the regret component for one objective with the regret component for the other objective. Without the guidance on relative performance on both objectives as we discuss later, the learning algorithm is not as good as it can be in addressing the objectives together.

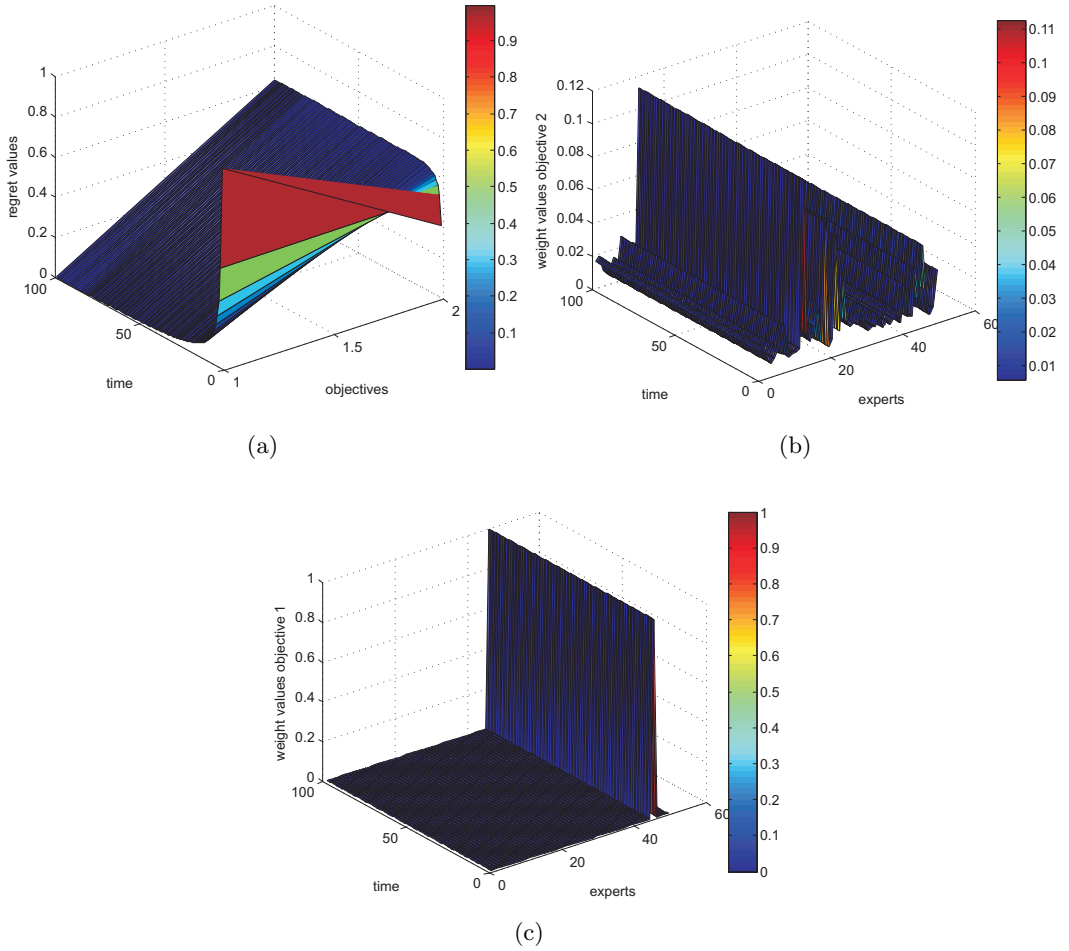


Figure 6.4: Further results of IWU with Pareto problem formulation. (a) Regret surface over both objectives over 100 runs (b) Weights for Objective 2 over 100 runs (c) Weights for Objective 1 over 100 runs.

The figure above adds to the results in previous figure to illustrate the behaviour of IWU learner.

6.6.2 Relative Weight Update (RWU)

To avoid the problems of independent weight updates, and to relate the performance of the learner with respect to each objective, we modify the multiplicative weight update step for the experts. Initially, the weight of an expert in a weighted average forecaster is allowed to vary as a convex combination of its regret on individual objectives such that the relative performance of the learner towards both objectives can be influenced by the expert's weight updates as follows:

$$\mathbf{w}_{i,t}^{m} = \frac{\mathbf{w}_{i,t-1}^{m} \sum_{k=1}^{M} e^{-\eta \mathbf{l}_k(\mathbf{f}_{i,t}, \mathbf{y}_t)}}{\sum_{j=1}^{L} \mathbf{w}_{j,t-1}^{m} e^{-\eta \mathbf{l}^m(\mathbf{f}_{j,t-1}, \mathbf{y}_t)}}. \tag{6.6}$$

The update step as in (6.6), relates performance of the expert i in terms of each of the m objectives at every step. This modification serves the basic intuition of relating the performance of the experts and hence the learner with respect to both objectives together. However, the limitation of this step is in not being able to relate the overall performance of an expert in both objectives with the desired overall performance in both objectives. The motivation being, that at the Pareto front, both objectives cannot be simultaneously improved - improving one inevitably degrades the other. An additional desirability or importance factor if associated with the process, when it converges towards the Pareto front, would enable the algorithm to meet the desired performance of minimizing regret with time while predicting Pareto optimal solutions.

6.6.3 Pareto Descent Weight update (PDWU)

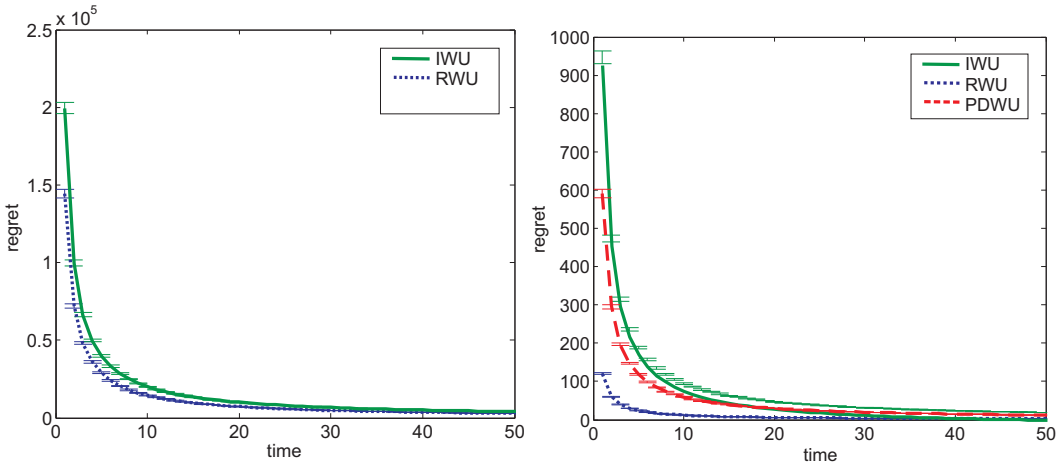
In this method, we show how to ensure that at every round t, the instantaneous vector is chosen such that, the direction of drift does not increase the regret much on individual objective and also the drift is in the direction where both the objectives are improved optimally. From our knowledge on descent directions in multi-criteria optimization, we know that if a descent direction is sought that produces Pareto optimal solutions, the best direction to choose is a Pareto descent direction (Harada and Kobayashi, 2006) which gives non-dominated solutions (optimal solutions) in the best case, and in the worst case the descent directions are automatically obtained which gives dominated (sub-optimal) solutions. The way to ensure this is to have an importance vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_m) \in \mathbb{R}^M$ with constraints for convexity $\alpha_k \geq 0$ and $\sum_{k=1}^{M} \alpha_k = 1$ that places relative importance on the objectives. We use this notion of choosing Pareto descent directions whilst choosing the instantaneous regret vector $\mathbf{r}_t^m$. The instantaneous regret vector and hence the weight of the experts at every round, control the learner's regret by using the gradient of the potential information in keeping the regret close to minimum and the convex combination of the objective vectors control the direction of movement of the learning algorithm towards the optimal Pareto front. The convex combination is given by:

$$\mathbf{r}_t^m = \sum_{k=1}^{M} \alpha_k \bigtriangledown \Phi_k(\mathbf{R}_{t-1}). \tag{6.7}$$

The weight $\mathbf{w}_{i,t}^m$ of expert i at round t for objective m is then a convex combination of the importance vector and the potential of the expert's regret in the direction of individual objectives, and is given by: $\mathbf{w}_{i,t-1}^m = \sum_{k=1}^{M} \alpha_k \phi'(\mathbf{R}_{i,t-1}^m)$

which simplifies to:

$$\mathbf{w}_{i,t}^m = \frac{\mathbf{w}_{i,t-1}^m \sum_{k=1}^{M} \alpha_k e^{-\eta \mathbf{l}_k(\mathbf{f}_{i,t}, \mathbf{y}_t)}}{\sum_{j=1}^{L} \mathbf{w}_{j,t-1}^m e^{-\eta \mathbf{l}^m(\mathbf{f}_{j,t-1}, \mathbf{y}_t)}}. \tag{6.8}$$



(a) (b)

Figure 6.5: Instantaneous regret of learner for IWU, RDU and PDWU for (a) objective *performance*, (b) objective *power*. IWU is shown in solid GREEN, RWU in BLUE dots and PDWU in dashed RED. Each line in the graph corresponds to an average over 1000 runs of the experiment.

6.6.4 Sampling Importance Vector

Another crucial part of our algorithm is the selection of the importance vector $\boldsymbol{\alpha_t}$ at every round t. Typically, the complete set of Pareto descent directions (which includes the descent directions) give a convex combination of objective vectors; that forms a convex cone pointed at the origin (Harada and Kobayashi, 2006), exploiting the simultaneous linear inequality as in (6.3) (Harada and Kobayashi, 2006). For simplicity, we can consider the convex cone to be a simplex as well, that is a triangle in this case that can be generalized to a tetrahedron or higher dimensions. We enforce the condition that this simplex or the convex cone is pointed at the origin by adding the following constraint to (6.7): $-1 \geq \mathbf{r}_{i,t}^m \leq 1$ for expert i. Any point sampled from the convex cone then should give an importance vector $\boldsymbol{\alpha}_t$ and the convex combination of $\boldsymbol{\alpha}_t$ and the objective vectors in terms of gradient of the potential functions with respect to to each objective $\phi(\mathbf{R}_t^m)$ gives the descent (drift) direction or instantaneous regret vector $\mathbf{r}_t^m$ in (6.7) for the learning algorithm to proceed. We sample $\boldsymbol{\alpha_t}$ from the convex hull at every round t of the learning algorithm.

6.7 Simulation Results

We perform preliminary simulation experiments, to model the example power and performance problem we explained in previous sections. These experiments on synthetic data illustrates our results. Typically on a multi-core based embedded device (4 cores or more), the different power-performance settings can be as many as 40. The number

of experts in our experiments is configured to 50. In reality, the power-performance policy experts are functions of the workloads currently executing on the device that predicts the objective vectors of trade-off values for each objective. Here, we use the objective vectors of power-performance trade-off predictions as our experts, rather than the actual functions themselves as in the context of on-line learning with experts, what matters are the predictions of the experts which can be imagined to be functions of the workloads. As the power performance trade-off values for any workload lie on a Pareto curve (Cochran et al., 2011), our experts are randomly chosen from a large set of convex Pareto curves; we assume that this space constitute the feasible space of objective solutions. We designate the optimal Pareto front as the true sequence of experts in hindsight (not available to the learner), that the algorithm should converge to. The number of inputs (workloads) seen by our algorithm is preset to 20 but this is not a restriction. As mentioned before, we have two objectives, that of power savings and performance delay represented as $m = 2$ in our experiments. For most of our experiments, we vary the learning rate between 0.3 to 0.5. We also run experiments on learning rate of 0.01 and 0.8 for verifying bounds. All the on-line learning games comprise 100 rounds and the results are averaged over 1000 runs. We measure the normalized regret of the learning algorithm using (6.2) as the main performance evaluation metric in these experiments. Our importance vector $\boldsymbol{\alpha}_t$ is sampled uniformly from the convex cone pointed at the origin. We perform experiments using both exponentially weighted average forecaster and weighted average forecasters. Our convex loss function is an absolute loss function. The figure below illustrates the results of our novel algorithm in the current problem formulation.

Figure 6.5 shows the results of the normalized regret as explained in (6.2), as a performance measure for all the three methods used in our learning algorithm. This performance measure implies that the learner should have a vanishing per-round regret or the instantaneous regret. The difference in cumulative loss of the learner and the cumulative loss of the best expert in hindsight should grow sub linearly (almost surely) irrespective of the outcome the learner sees. We use the independent weight update method as discussed in (6.5) as our baseline method that has the on-line learning with experts framework without modification to the weight update step for the experts. The methods of relative weight update and Pareto descent weight update that use the weight updates in (6.6) and (6.8), are compared against the baseline method. Figure 6.5(a) shows results for objective $performance$ while Figure 6.5(b) shows results for objective *power*. As seen from the results, RWU in Figure 6.5(a) performs better than the IWU. In this example, PDWU performs almost as good as the baseline method. Figure 6.5(b) shows that in the case of objective *power*, both methods RWU and PDWU perform better than baseline method. We see sub linear growth in the difference of cumulative loss of the learner and that of the true expert for all our methods. The graphs in Figure 6.5 show the instantaneous regret of the learner as it converges to zero. In our example problem of power-performance, this result implies that an optimal set of trade-off values

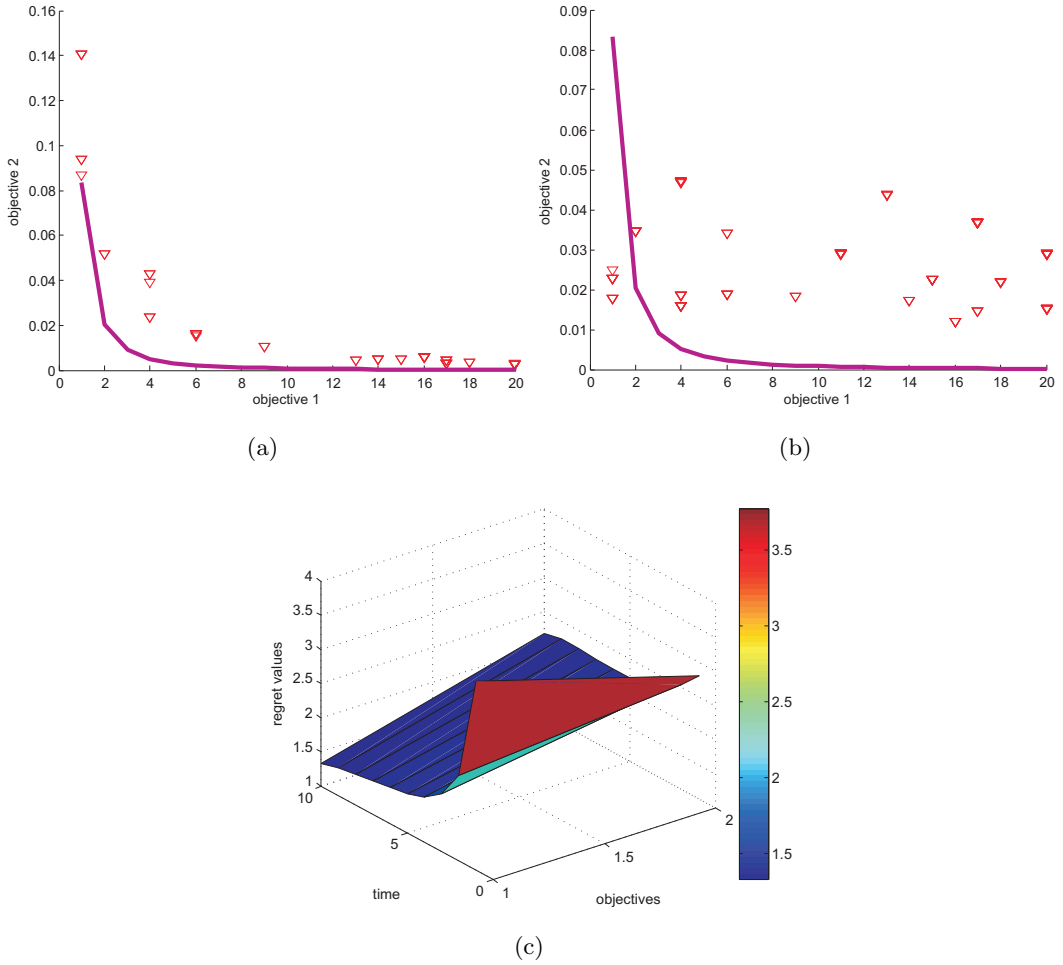


Figure 6.6: Results of PDWU with Pareto problem formulation. (a) Predictions shown in ∇ made by PDWU learner averaged over 50 runs (b) Predictions made by IWU in contrast over 100 runs (c) Regret surface over both objectives over 10 runs.

are predicted that lie on the Pareto frontier (this is because the learner's regret goes sublinearly to zero), that gives the maximum power savings with minimum performance delay.

6.8 Discussion and Conclusion

In this chapter we have seen how an on-line learning framework can be adapted to the continuous learning of multiple tasks problem. The independent weight update approach evaluates the learner in how well it addresses each of the objectives separately. The relative weight update and Pareto descent weight update methods perform a convex combination of the performance of the algorithm in addressing multiple objectives simultaneously, while making predictions from the optimal Pareto front. An importance

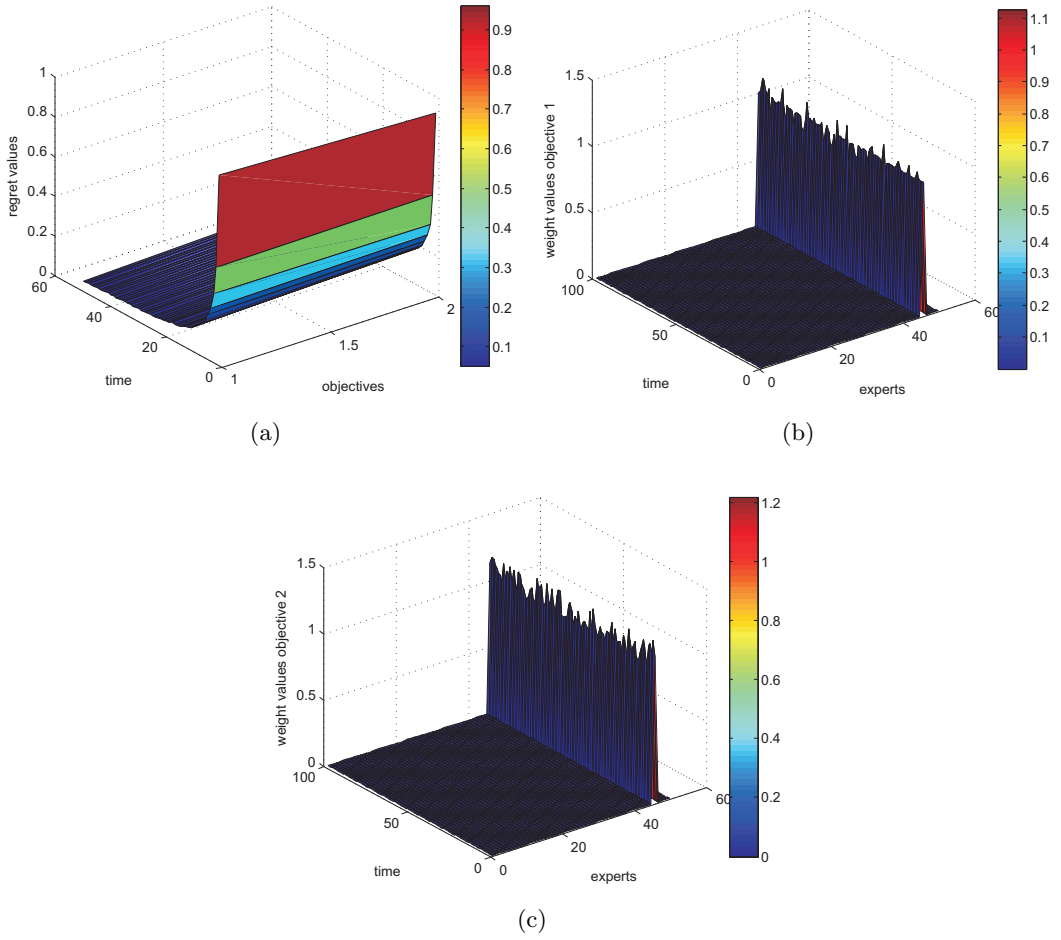


Figure 6.7: Further results of PDWU with Pareto problem formulation (a) Regret surface over both objectives over 50 runs (b) Weights for Objective 1 over 100 runs (c) Weights for Objective 2 over 100 runs.

vector is used to relatively weigh the objectives in each round. The convex combination of the importance vector and the function of learner's regret provides a Pareto descent direction for faster convergence on the Pareto optimal front.

A future direction of this work is to provide a theoretical analysis of the learning approaches proposed and to achieve generalization bounds. We would also like to perform extensive experiments on empirical data obtained from the real embedded device and the data servers to determine how the lifetime of the battery or power source is maintained while balancing the power-performance trade-off. We need to analyse if restricting the expert weights to a convex subset has any other implication other than fast convergence.

Chapter 7

Conclusions

The research presented in this thesis investigates the online learning approaches to data with combinatorial structure. We study the large datasets with graphical structure, multiple objectives and combinatorial sets of actions in the context of sequential decision making. We are interested in such prediction problems where the data undergoes dynamic changes, and the number of solutions are typically exponentially big in the size of the input. This motivates the requirement for adaptive learning methods that can learn from dynamic data, with a polynomial complexity for large datasets.

The goal of this thesis was to develop techniques that enhance the robustness capabilities of sequential online learning algorithms in the context of input data with rich combinatorial structure. We addressed this main objective through the development of three novel online learning methods working with sequential graph labelling in a semi supervised environment and one multiple objective prediction within supervised learning with experts. The first piece of work addressed the open question of minimizing the p-semi norm interpolation for consistent graph labelling when $p \to 1$, in the limit of zero temperature on an Ising model. The main accomplishment from this body of work was a deterministic approximation algorithm that used a neat characterization to reduce multiplicity of a combinatorial set called PQ graphs, to predict the label of the vertex over any sequence of queries. We also provided promising mistake bounds with one bound that matched the optimal bound on trees. In the second major work, we presented a probabilistic approach to the same problem. The work developed a mean field variational approximation technique for predicting the marginal probability of the label of the vertex queries, in the similar semi supervised setting. In both bodies of work, extensive empirical evaluation was carried out and several experimental results were provided for validating the results.

The third novel contribution in this thesis is the work on Ising bandits, which took a direction to enhance the robustness of online feedback algorithms within the similar semi supervised graph labelling, that served as the contextual setting. A novel linear

programming relaxation to the $p \to 1$ setting was provided within this context. The last body of work studied the full feedback setting of experts within the multiple objective optimization structure. A novel convex combination of the prediction of the experts was constructed to predict from the Pareto front. Through these four main lines of work in this thesis, novel techniques were presented to handle combinatorial data through a series of robustness structural settings, within the the sequential prediction online games between the learning algorithm and the environment.

This thesis advanced the state of the art in online prediction on combinatorial data with applications to Big Data, sequential data and data with different feedback assumptions. As more data gets generated, we believe that developing sequential algorithms would be key to cope with the size, source, management and storage of data. Although modern computational and processing power advancement in the terms of GPUs and clusters can get around the the problem of data size as of now, developing robust theoretically sound algorithms that can learn from data on the fly, is definitely the way forward.

7.1 Future Work

The techniques developed in this thesis present new possibilities of addressing the sequential decision making with combinatorial data. There are potentially a multitude of ways in extending the work in future.

The key body of work on online graph labelling within the Ising model assumption at the limit of zero temperature works on uniformly labelled clusters in large sparse graphs. The prediction algorithm of longest path in its current state, does not incorporate the connectivity while making the prediction. For example, when a vertex is queried, the algorithm finds the longest path the vertex is present on, within the PQ direct acyclic graph and predicts with the nearest neighbour on that path. The intuition being that in case of a mistake, the labels on the longest segment of the path are deduced thereby reducing the number of the mistakes in the long run. However, the algorithm does not take into affect the connectivity while making the prediction, posing as a limitation. In Herbster et al. (2015), this point is mentioned as part of the discussion. We have indeed implemented and performed experimentation with two such connectivity based prediction approaches. First one of them called `edge-majority` by Mark Herbster, where the algorithm predicts with the label of the super vertex in question, present in the PQ graph such that in case of a mistake, that allows maximum number of edges to be collapsed to that supervertex. The other method is the `vertex-majority` by Mark Herbster, that predicts such that in case of a mistake, the maximum number of vertices collapse to the super-vertex or in other words, the labels of maximum number of vertices are deduced. The experiments showed promising results with minor improvement on `longest-path`. In principle, we believe that similar connectivity exploiting prediction

mechanism will benefit the algorithm in a holistic way. One possibility is to encode the connectivity information in the meta-graph called second order chain that can be approximated from the PQ hyper-graph and use a monotone, permutation invariant dynamic program to recursively predict the labels (Çeliktutan et al., 2015).

A possible continuation of the work on variational mean field approximation can be an extended empirical evaluation on multi-class labelling problems in computer vision as the algorithm is capable of multi-class classification. One could compare this method with the deterministic approach in Chapter 3. A natural extension of this work is also in online multi-instance labelling where the input is a set of labelled bags; the mean field approximation naturally fits into this problem. One could also consider parallelizing the online mean field approximation of semi supervised learning across GPU clusters for relative speed up of the computation process. All the non-neighbours that do not depend on each other could be updated in parallel. The interesting challenge will be in discovering sets of vertices whose labelling are independent of each other. One possibility is to execute the algorithm beforehand to detect non-interacting sets as the prior. The assumption excludes fully connected graphs, where every state depend on everything else. The other challenge is to prevent simultaneous updating to avoid cycling. In practice, a simple strategy of randomly selecting two different sections of the graph to update, works relatively well.

Further, as mentioned in Chapter 3, we would like to investigate in what sense are the predictions of `longest-path` deterministic approximate predictions of `0-Ising`. Let the vectors $\boldsymbol{y}, \hat{\boldsymbol{y}}^{\texttt{0I}}, \hat{\boldsymbol{y}}^{\texttt{LP}} \in \{0,1\}^n$ denote the true labelling of the graph, the vectors of the sequential predictions of `0-Ising` and `longest-path` and let $B(\mathcal{S})$ denote a mistake bound of `0-Ising` and `longest-path` on an example sequence $\mathcal{S}$. Then we have:

$$\|\hat{\boldsymbol{y}}^{\texttt{0I}} - \hat{\boldsymbol{y}}^{\texttt{LP}}\|_1 \leq \|\boldsymbol{y} - \hat{\boldsymbol{y}}^{\texttt{0I}}\|_1 + \|\boldsymbol{y} - \hat{\boldsymbol{y}}^{\texttt{LP}}\|_1 \leq 2B(\mathcal{S})$$

Thus the sequence of predictions $\hat{\boldsymbol{y}}^{\texttt{LP}}$ from the deterministic approximation `longestPath` are guaranteed to never differ from the sequence of predictions $\hat{\boldsymbol{y}}^{\texttt{0I}}$ of `0-Ising` except in at most $2B(\mathcal{S})$ trials. Analysing the quality of approximations carried out in both Chapters 3 and 4, would contribute immensely in the increased effectiveness of the online methods developed in this thesis.

As a continuation of Chapter 5 on Ising bandits, we would like to evaluate the method on the real life advertisement and recommendation data available on the internet through product recommendation datasets. Further, we would like to study the effects of removing the relaxation that the adversary cannot increase the "cut-size". It would be interesting to allow the adversary to be flexible for worst-case guarantees. With the current relaxation, one step ahead could be to bound the number of mistakes or mistaken predictions per game; every time the cut changes, a different game begins. A possible extension of the work in Chapter 6 could be in casting the multiple objective

Pareto online optimization problem in the framework of the recent work in Pareto regret frontier (Koolen, 2013).

References

Agarwal, S. (2011). Online learning from experts: Weighed majority and hedge. http://www.shivani-agarwal.net/Teaching/E0370/Aug-2011/Lectures/20-scribe1.pdf. [Online; accessed 22-Jan-2016].

Alamgir, M. and von Luxburg, U. (2011). Phase transition in the family of p-resistances. In Shawe-Taylor, J., Zemel, R. S., Bartlett, P. L., Pereira, F. C. N., and Weinberger, K. Q., editors, *NIPS*, pages 379–387.

Allenberg, C., Auer, P., Györfi, L., and Ottucsák, G. (2006). Hannan consistency in on-line learning in case of unbounded losses under partial monitoring. In *Algorithmic Learning Theory*, pages 229–243. Springer.

Amin, K., Kearns, M., and Syed, U. (2012). Graphical models for bandit problems. *arXiv preprint arXiv:1202.3782*.

Auer, P. (2003). Using confidence bounds for exploitation-exploration trade-offs. *The Journal of Machine Learning Research*, 3:397–422.

Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002a). Finite-time analysis of the multi-armed bandit problem. *Machine learning*, 47(2-3):235–256.

Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. E. (1995). Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on*, pages 322–331. IEEE.

Auer, P., Cesa-Bianchi, N., and Gentile, C. (2002b). Adaptive and self-confident on-line learning algorithms. *Journal of Computer and System Sciences*, 64(1):48–75.

Ball, M. O. and Provan, J. S. (1983). Calculating bounds on reachability and connectedness in stochastic networks. *Networks*, 13(2):253–278.

Barabasi, A.-L. and Oltvai, Z. N. (2004). Network biology: understanding the cell's functional organization. *Nature reviews genetics*, 5(2):101–113.

Barzdin, J. M. and Frievald, R. V. (1972). On the prediction of general recursive functions. *Soviet Math. Doklady*, 13:1224–1228.

Bastian, M., Heymann, S., and Jacomy, M. (2009). Gephi: An open source software for exploring and manipulating networks. In *International AAAI Conference on Weblogs and Social Media*.

Belkin, M., Matveeva, I., and Niyogi, P. (2004). Regularization and semi-supervised learning on large graphs. In *COLT*, volume 3120, pages 624–638. Springer.

Belkin, M. and Niyogi, P. (2004). Semi-supervised learning on riemannian manifolds. *Mach. Learn.*, 56(1-3):209–239.

Berry, D. A. and Fristedt, B. (1985). *Bandit problems: sequential allocation of experiments (Monographs on statistics and applied probability)*. Springer.

Blackwell, D. (1954). Controlled random walks. In *Proceedings of the International Congress of Mathematicians*, volume 3, pages 336–338.

Blackwell, D. (1956). An analog of the minimax theorem for vector payoffs. *Pacific Journal of Mathematics*, 6(1):1–8.

Blum, A. and Chawla, S. (2001). Learning from labeled and unlabeled data using graph mincuts. In *ICML*, pages 19–26.

Bokrantz, R. and Forsgren, A. (2011). A dual algorithm for approximating pareto sets in convex multi-criteria optimization. Technical Report TRITA-MAT-2011-OS3, Department of Mathematics, Royal Institute of Technology.

Bousquet, O. (2002). Transductive learning: Motivation, models, algorithms. `http://www.kyb.mpg.de/fileadmin/user_upload/files/publications/pdfs/pdf2527.pdf`.

Brown, M. and Smith, R. (2003). Effective use of directional information in multi-objective evolutionary computation. In *Genetic and Evolutionary Computation (GECCO-2003)*, pages 197–197. Springer.

Bubeck, S. and Cesa-Bianchi, N. (2012). Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 5(1):64–87.

Cameron, P. (2007). Notes on Combinatorics. `http://www.maths.qmul.ac.uk/~pjc/notes/comb.pdf`. [Online; accessed 25-Jan-2016].

Çeliktutan, O., Wolf, C., Sankur, B., and Lombardi, E. (2015). Fast exact hypergraph matching with dynamic programming for spatio-temporal data. *Journal of Mathematical Imaging and Vision*, 51(1):1–21.

Cesa-Bianchi, N., Gentile, C., and Vitale, F. (2009). Fast and optimal prediction on a labeled tree. In *Proceedings of the 22nd Annual Conference on Learning*. Omnipress.

Cesa-Bianchi, N., Gentile, C., Vitale, F., and Zappella, G. (2010). Random spanning trees and the prediction of weighted graphs. In *Proceedings of the 27th International Conference on Machine Learning (27th ICML)*, pages 175–182.

Cesa-Bianchi, N. and Lugosi, G. (2003). Potential-based algorithms in on-line prediction and game theory. *Machine Learning*, 51(3):239–261.

Cesa-Bianchi, N. and Lugosi, G. (2006). *Prediction, learning, and games*. Cambridge University Press.

Chapelle, O., Schölkopf, B., et al. (2006a). *A discussion of semi-supervised learning and transduction*. MIT Press Cambridge.

Chapelle, O., Schölkopf, B., Zien, A., et al. (2006b). *Semi-supervised learning*. MIT Press Cambridge.

Chapelle, O., Weston, J., and Schölkopf, B. (2003). Cluster kernels for semi-supervised learning. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15*, pages 601–608. MIT Press.

Cochran, R., Hankendi, C., Coskun, A., and Reda, S. (2011). Pack & cap: adaptive dvfs and thread packing under power caps. In *Proceedings of the 44th annual IEEE/ACM International Symposium on Microarchitecture*, pages 175–185. ACM.

Dantzig, G. and Fulkerson, D. R. (2003). On the max flow min cut theorem of networks. *Linear inequalities and related systems*, 38:225–231.

Delalleau, O., Bengio, Y., and Le Roux, N. (2005). Efficient non-parametric function induction in semi-supervised learning. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, pages 96–103. Citeseer.

Dhiman, G. and Rosing, T. (2009). System-level power management using online learning. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 28(5):676–689.

Di Castro, D., Gentile, C., and Mannor, S. (2011). Bandits with an edge. In *CoRR, abs/1109.2296*.

Eban, E., Birnbaum, A., Shalev-Shwartz, S., and Globerson, A. (2012). Learning the experts for online sequence prediction. *Arxiv preprint arXiv:1206.4604*.

Esmaeilzadeh, H., Blem, E., St Amant, R., Sankaralingam, K., and Burger, D. (2011). Dark silicon and the end of multicore scaling. In *Proceeding of the 38th annual international symposium on Computer architecture*, pages 365–376. ACM.

Esmaeilzadeh, H., Cao, T., Yang, X., Blackburn, S., and McKinley, K. (2012). What is happening to power, performance, and software? *Micro, IEEE*, 32(3):110–121.

Executive Office of the President (2014). Big data: Seizing opportunities, preserving values.

Fliege, J. and Svaiter, B. (2000). Steepest descent methods for multicriteria optimization. *Mathematical Methods of Operations Research*, 51(3):479–494.

Ford, L. R. and Fulkerson, D. R. (1956). Maximal Flow through a Network. *Canadian Journal of Mathematics*, 8:399–404.

Freund, Y. and Schapire, R. E. (1996). Game theory, on-line prediction and boosting. In *Proceedings of the ninth annual conference on Computational learning theory*, pages 325–332. ACM.

Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139.

Freund, Y. and Schapire, R. E. (1999). Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29(1):79–103.

Gansner, E. R. and North, S. C. (2000). An open graph visualization system and its applications to software engineering. *Software - Practice and Experience*, 30(11):1203–1233.

Gärtner, T. and Garriga, G. C. (2007). The cost of learning directed cuts. In *Proceedings of the 18th European Conference on Machine Learning*, pages 152–163.

Gentile, C., Li, S., and Zappella, G. (2014). Online clustering of bandits. *arXiv preprint arXiv:1401.8257*.

Gentile, C. and Orabona, F. (2014). On multilabel classification and ranking with bandit feedback. *The Journal of Machine Learning Research*, 15(1):2451–2487.

Getz, G., Shental, N., and Domany, E. (2006). Semi-supervised learning–a statistical physics approach. *arXiv preprint cs/0604011*.

Ghahramani, Z. (2012). Graph-based semi supervised learning at machine learning summer school. http://mlg.eng.cam.ac.uk/zoubin/talks/lect3ssl.pdf. [Online; accessed 22-Jan-2016].

Ghahramani, Z. (2015). Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553):452–459.

Ghosh, S., Lovell, C. J., and Gunn, S. R. (2013). Towards pareto descent directions in sampling experts for multiple tasks in an on-line learning paradigm. In *Proceedings of the AAAI Spring Symposium Series of Lifelong Machine Learning 2013*, volume 13 of *SS-13-05*. AAAI Press.

Ghosh, S. and Prügel-Bennett, A. (2015a). Ising bandits with side information. In *Machine Learning and Knowledge Discovery in Databases*, volume 9284 of *Lecture Notes in Computer Science*, pages 448–463. Springer International Publishing.

Ghosh, S. and Prügel-Bennett, A. (2015b). Online mean field approximation for automated experimentation. In *Proceedings of The 4th Workshop on Machine Learning for Interactive Systems*, volume 43, pages 31–35. Journal of Machine Learning Research.

Gittins, J., Glazebrook, K., and Weber, R. (2011). *Multi-armed bandit allocation indices*. John Wiley & Sons.

Goldberg, L. A. and Jerrum, M. (2007). The complexity of ferromagnetic ising with local fields. *Combinatorics, Probability & Computing*, 16(1):43–61.

Grandvalet, Y. and Bengio, Y. (2004). Semi-supervised learning by entropy minimization. In *Advances in neural information processing systems*, pages 529–536.

Harada, K. and Kobayashi, S. (2006). Local search for multiobjective function optimization: Pareto descent method. In *8th Annual Conference on Genetic and Evolutionary Computation (GECCO-2006)*, pages 659–666. ACM Press.

Hazan, E. and Seshadhri, C. (2009). Efficient learning algorithms for changing environments. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 393–400. ACM.

Herbster, M. (2008). Exploiting cluster-structure to predict the labeling of a graph. In *Proceedings of the 19th International Conference on Algorithmic Learning Theory*, pages 54–69.

Herbster, M. and Lever, G. (2009). Predicting the labelling of a graph via minimum p-seminorm interpolation. In *Proceedings of the 22nd Annual Conference on Learning Theory (COLT'09)*.

Herbster, M., Lever, G., and Pontil, M. (2009). Online prediction on large diameter graphs. In *Advances in Neural Information Processing Systems*, pages 649–656.

Herbster, M., Pasteris, S., and Ghosh, S. (2015). Online prediction at the limit of zero temperature. In *Advances in Neural Information Processing Systems 28*, pages 2917–2925. Curran Associates, Inc.

Herbster, M. and Pontil, M. (2006). Prediction on a graph with a perceptron. In *Advances in neural information processing systems*, pages 577–584.

Herbster, M., Pontil, M., and Wainer, L. (2005). Online learning over graphs. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 305–312, New York, NY, USA. ACM.

Herbster, M. and Warmuth, M. (1998). Tracking the best expert. *Machine Learning*, 32(2):151–178.

Hu, X., Huang, Z., and Wang, Z. (2003). Hybridization of the multi-objective evolutionary algorithms and the gradient-based algorithms. In *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, volume 2, pages 870–877. IEEE.

Hutter, M. and Poland, J. (2004). Prediction with expert advice by following the perturbed leader for general weights. In *Algorithmic Learning Theory*, pages 279–293. Springer.

Hutter, M. and Poland, J. (2005). Adaptive online prediction by following the perturbed leader. *Arxiv preprint cs/0504078*.

Ising, E. (1925). A contribution to the theory of ferromagnetism. *Z. Phys*, 31(1):253–258.

Joachims, T. (1999). Transductive inference for text classification using support vector machines. In *ICML*, volume 99, pages 200–209.

Kalyanakrishnan, S., Tewari, A., Auer, P., and Stone, P. (2012). Pac subset selection in stochastic multi-armed bandits.

Koolen, W. M. (2013). The pareto regret frontier. In *Advances in Neural Information Processing Systems*, pages 863–871.

Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The annals of mathematical statistics*, pages 79–86.

Lawler, E. (1985). *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. Wiley, New York.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

Letham, B., Rudin, C., McCormick, T. H., and Madigan, D. (2015). Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *Ann. Appl. Stat.*, 9(3):1350–1371.

Lichman, M. (2013). UCI machine learning repository. `http://archive.ics.uci.edu/ml`. [Online; accessed 22-Jan-2016].

Littlestone, N. (1988). Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318.

Littlestone, N. and Warmuth, M. K. (1994). The weighted majority algorithm. *Information and computation*, 108(2):212–261.

Lugosi, G., Papaspiliopoulos, O., and Stoltz, G. (2009). Online multi-task learning with hard constraints. *Arxiv preprint arXiv:0902.3526*.

Luxburg, U. V., Radl, A., and Hein, M. (2010). Getting lost in space: Large sample analysis of the resistance distance. In *NIPS 23*, pages 2622–2630.

Meisner, D., Sadler, C., Barroso, L., Weber, W., and Wenisch, T. (2011). Power management of online data-intensive services. In *Computer Architecture (ISCA), 2011 38th Annual International Symposium on*, pages 319–330. IEEE.

Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press.

Nadler, B., Srebro, N., and Zhou, X. (2009). Statistical analysis of semi-supervised learning: The limit of infinite unlabelled data. In *NIPS*, pages 1330–1338.

Picard, J.-C. and Queyranne, M. (1980). On the structure of all minimum cuts in a network and applications. In Rayward-Smith, V., editor, *Combinatorial Optimization II*, volume 13 of *Mathematical Programming Studies*, pages 8–16. Springer Berlin Heidelberg.

Provan, J. S. and Ball, M. O. (1983). The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM Journal on Computing*, 12(4):777–788.

Rakhlin, A., Abernethy, J., Agarwal, A., Bartlett, P., Hazan, E., and Tewari, A. (2009). Lecture notes on online learning draft. `http://www-stat.wharton.upenn.edu/~rakhlin/courses/stat991/papers/lecture_notes.pdf`. [Online; accessed 22-Jan-2012].

Rakhlin, S. (2008). Online line lecture 09 9.520. http://www.mit.edu/~9.520/spring08/Classes/online_learning_2008.pdf. [Online; accessed 22-Jan-2016].

Roweis, S. (2006). Data for matlab hackers. `http://www.cs.nyu.edu/~roweis/data.html`. [Online; accessed 29-April-2015].

Seldin, Y., Cesa-Bianchi, N., Laviolette, F., Auer, P., Shawe-Taylor, J., and Peters, J. (2011). Pac-bayesian analysis of the exploration-exploitation trade-off. *Arxiv preprint arXiv:1105.4585*.

Shalev-Shwartz, S. (2012). Online learning and online convex optimization. *Foundations and Trends in Machine Learning: Vol. 4: No 2, pp 107-194.*

Shannon, C. E. (1949). Communication in the presence of noise. *Proceedings of the IRE*, 37(1):10–21.

Shannon, C. E. (2001). A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55.

Shannon, C. E. and Weaver, W. (2015). *The mathematical theory of communication*. University of Illinois press.

Singhee, A. (2011). Pareto sampling using simplicial refinement by derivative pursuit. US Patent 20,110,307,430.

Szummer, M. and Jaakkola, T. (2001). Partially labeled classification with markov random walks. In *NIPS*, pages 945–952.

Trevisan, L. (2011). Lecture 15 CS261 Optimization. `http://theory.stanford.edu/~trevisan/cs261/lecture15.pdf`.

Valko, M., Munos, R., Kveton, B., and Kocák, T. (2014). Spectral bandits for smooth graph functions. In *31th International Conference on Machine Learning*.

Vitale, F., Cesa-Bianchi, N., Gentile, C., and Zappella, G. (2011). See the tree through the lines: The shazoo algorithm. In *Advances in Neural Information Processing Systems*, pages 1584–1592.

Von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416.

Wainwright, M. J. and Jordan, M. I. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1-2):1–305.

Wang, S., Peng, J., Ma, J., and Xu, J. (2016). Protein secondary structure prediction using deep convolutional neural fields. *Scientific reports*, 6.

Webspam (2007). Web spam challenge home page. `http://webspam.lip6.fr/wiki/pmwiki.php`. [Online; accessed 30-April-2015].

Zhou, D., Bousquet, O., Lal, T. N., Weston, J., and Schölkopf, B. (2003). Learning with local and global consistency. In *NIPS*.

Zhu, X. (2005). Semi-supervised learning literature survey. Technical Report Computer Sciences 1530, University of Wisconsin-Madison.

Zhu, X. and Ghahramani, Z. (2002). Towards semi-supervised classification with markov random fields. Technical Report CMU-CALD-02-106, Carnegie Mellon University.

Zhu, X., Ghahramani, Z., and Lafferty, J. D. (2003). Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, pages 912–919.

Zitzler, E., Laumanns, M., and Bleuler, S. (2004). A tutorial on evolutionary multiobjective optimization. *Metaheuristics for Multiobjective Optimisation*, pages 3–37.