SUPERVISED LEARNING WITH RANDOM LABELLING ERRORS

by

JAKRAMATE BOOTKRAJANG

A thesis submitted to
The University of Birmingham
for the degree of
DOCTOR OF PHILOSOPHY

School of Computer Science
College of Engineering and Physical Sciences
The University of Birmingham
August 2013

Abstract

Classical supervised learning from a training set of labelled examples assumes that the labels are correct. But in reality labelling errors may originate, for example, from human mistakes, diverging human opinions, or errors of the measuring instruments. In such cases the training set is misleading and in consequence the learning may suffer.

In this thesis we consider probabilistic modelling of random label noise. The goal of this research is two-fold. First, to develop new improved algorithms and architectures from a principled footing which are able to detect and bypass the unwanted effects of mislabelling. Second, to study the performance of such methods both empirically and theoretically. We build upon two classical probabilistic classifiers, the normal discriminant analysis and the logistic regression and introduce the label-noise robust versions of these classifiers. We also develop useful extensions such as a sparse extension and a kernel extension in order to broaden applicability of the robust classifiers. Finally, we devise an ensemble of the robust classifiers in order to understand how the robust models perform collectively.

Theoretical and empirical analysis of the proposed models show that the new robust models are superior to the traditional approaches in terms of parameter estimation and classification performance.

To my beloved family

Acknowledgements

First and foremost I would like to thank my family for their wonderful support. I thank my supervisor, Dr. Ata Kabán, who is always be there when I need help, inspires and guides me to the best direction. I thank my research monitoring group member: Professor Xin Yao and Dr. Peter Tinõ for their invaluable comments and suggestions. I thank my thesis examiners: Dr. Gavin Cawley and Dr. Shan He for sparing their valuable time to read and correct my thesis. I thank all friends, Dr. Bob Durrant, Rodrigo Soares, Dr. Leandro Minku, Haobo Fu, Chuangjie Xu and everyone I met in Birmingham who made me feel like I am at home. I am in debt to all of the schools and the universities I have attended – Anuban Lampang Kelangrath Anusorn School, Bunyawat Witthayalai School, Seoul National University and the University of Birmingham, as well as to all teachers and lecturers. Without them I won't be what I am today. Lastly, my greatest gratitude goes to the Royal Thai Government who generously provides financial support for my entire higher educations.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Classification & supervised learning

Classification is the task of inferring a function $h : \mathbb{R}^M \to \mathbb{R}$ which maps an observation input to a label response using a labelled set of training data so that we can use the function to predict the response of an previously unseen observation in the future. The training data are usually given as a set of $(\mathbf{x}, y)$ pairs, where $\mathbf{x}$ is an M-dimensional observation vector representing a point in an M-dimensional space and y is a discrete valued random variable representing the class label. It is assumed that the data pairs $(\mathbf{x}, y)$ are sampled from some probability distribution. The role of the class labels is to guide or to supervise a learning algorithm towards the desired classification rule which minimises the error of classifying unseen data drawn from the same distribution.

The error can be quantified in a more concrete way by using the notion of *loss* (Vapnik [1998]). Define $L(h(\mathbf{x}), y)$ to be the loss of predicting y using a decision function (i.e., a classifier) h. We are then interested in the expected loss (or risk) associated with the

classifier h over all the data pairs,

$$R[h] = \mathbb{E}_{\mathbf{x},y}[L(h(\mathbf{x}), y))] \tag{1}$$

where $\mathbb{E}[]$ is the expectation which is usually replaced by an average in finite samples cases in practice, because the true expectation is unknown. In such cases, the expected loss is called an empirical loss. According to the Empirical Risk Minimisation (ERM) principle, the optimal classifier is the one that minimises the empirical loss. This seems like a good approach provided that we have access to all the samples in the distribution and that the training labels are all correct. However, in reality, we only observe the distribution partially via a given set of training examples, which (possibly) contains labelling errors. Therefore, care must be taken when learning the classifier as the training set available might not fully represent the true underlying distribution. Picking the best classifier that minimises the expected loss could lead to a decision function that does well on the training data but performs poorly on unseen data. This phenomenon is often referred to as overfitting.

A classical remedy for the overfitting problem, regardless of its origin, is to impose some prior knowledge about the problem in the form of regularisation on the classifier h.

$$R_{reg}[h] = R[h] + \lambda\Phi(h) \tag{2}$$

Φ is used to measure the complexity of h. A classifier with high complexity will fit training data very well. The complexity measure can be defined in several ways using, for example, VC-Dimension (Vapnik [1998]), Rademacher complexity (Bartlett and Mendelson [2003]) or the norm of the parameter of h. If h is an ensemble of classifiers, the number of classifiers in the ensemble can be used as a complexity measure. Nonetheless all of the complexity measures share the same property that, the more complex h is the higher the

value of $\Phi(h)$. Consequently the regularised loss gives rise to a multi-objective problem where one has to reach a balance between fitting the observed data well using a high complexity function and having a simpler function that sacrifices some training errors for reduced overfitting by choosing an appropriate value of the regularisation parameter λ. The process of selecting a good value of λ is often referred to as a model selection problem. In general λ is chosen by cross-validation which validates a model using a small subset of holdout samples.

In this research we will consider *parametric probabilistic classifiers* where the classification rule, h, is obtained from a log ratio of class posterior probabilities. Estimating the parameters for probabilistic classifiers is usually accomplished by optimising a log loss (or negative log likelihood) using a Maximum Likelihood (ML) estimator. It can be shown that ML behaves like the ERM principle for this particular loss function. Assuming that the class label takes values from $\{0, 1\}$, a decision rule is defined as,

$$h(\mathbf{x}) = \log \frac{p(y = 1|\mathbf{x}, \theta_{y=1})}{p(y = 0|\mathbf{x}, \theta_{y=0})} \tag{3}$$

From this we decide $y = 1$ if $h(\mathbf{x}) > 0$. The most intuitive way to obtain the posterior probability is by modelling the distribution that generates the data. This leads to the generative approach where one assumes that the distribution of the data is one of the known probabilistic distributions. According to this approach, one has to estimate the parameters of class conditional probability distributions, after that the class posterior is calculated using Bayes theorem.

$$p(y|\mathbf{x}, \theta_y) = \frac{p(\mathbf{x}|\theta_y)p(y)}{\sum_y p(\mathbf{x}|\theta_y)p(y)} \tag{4}$$

An example of the generative classifier is the Normal Discriminant Analysis (NDA) where the data classes are assumed to be normally distributed according to the Gaussian

distribution. This assumption is considered reasonable as central limit theorem suggests that as the number of underlying causes that a feature is made of grows the distribution of that feature becomes approximately normal. Generative classifiers usually performs very well if the data distribution agrees with the assumption. However, if the assumption does not hold true the classification might not be accurate.

On the other hand, discriminative classifiers are based on the argument that "one should solve the classification problem directly and should never solve a more general problem as an intermediate step" (Vapnik [1998]). In other words, one should only need to find the optimal decision rule separating the data and need not care about its underlying distribution. Although the discriminative classification approach does not fully align to the argument, the approach attempts to solve classification problem more directly than the generative one, in the sense that it makes less assumptions on the data other than that the data points are independent and identically drawn from some unknown distribution. For the discriminative classifiers, what is important is the hyperplane separating the data according to the given labels. Under the discriminative model a class posterior is obtained from

$$p(y = 1|\mathbf{x}, \theta) = f(\mathbf{x}, \theta) \tag{5}$$

where f is any real-valued function. For example in Logistic Regression (LR), the sigmoid function $f(\mathbf{x}, \mathbf{w}) = 1/(1 + \exp(\mathbf{w}^T\mathbf{x}))$ is used to model the posterior probability.

1.2 Learning from mislabelled data

Regardless of the learning approach used, an integral part of the supervised learning framework is that the class labels guide the learning algorithm towards the desired classification rule. Unfortunately, there is often no guarantee that the given labels are perfect. Label errors are increasingly noticeable in today's classification tasks; as the scale and

difficulty of the labelling task increases, it becomes nearly impossible to obtain perfect label assignments. Mislabelling originates from several sources, including the subjective nature of the labelling task, the effect of communication noise and the lack of information to determine the true label of a given example. More recently, class label noise emerges as a side effect of crowdsourcing practices where annotators of different backgrounds are asked to perform labelling tasks (Snow et al. [2008], Raykar et al. [2010]). For example, Amazon's Mechanical Turk, Citizen science, Galaxy Zoo, to name just a few.

Label noise can be roughly categorised into two groups: random noise and non-random noise (Sloan [1995]). In all cases of label noise, it is likely that classical supervised learning that assumes perfect labels from the parameter estimation stage to the model selection stage, would be negatively affected. Indeed, class label noise inherent in training samples has been reported to deteriorate the performance of the existing classifiers in a broad range of classification problems (Krishnan and Nandy [1990], Lawrence and Schölkopf [2001], Yasui et al. [2004], Malossini et al. [2006]).

There is an increasing body of research literature that aims to address the issues related to learning from samples with noisy class label assignments. The seemingly straightforward approach is by means of data preprocessing where any suspect samples are removed or relabelled (Brodley and Friedl [1999], Barandela and Gasca [2000], Maletic and Marcus [2000], Sánchez et al. [2003], Muhlenbach et al. [2004], Jiang and Zhou [2004]). However, these approaches hold the risk of removing useful data too, which is detrimental to classification performance, especially when the number of training examples is limited (e.g. in biomedical domains). Most previous approaches try to detect mislabelled instances based on various heuristics, and very few take a principled modelling approach — with the notable exceptions of (Norton and Hirsh [1992], Lawrence and Schölkopf [2001], Li et al. [2007], Raykar et al. [2010]).

In this thesis we will consider the *random misclassification noise* and attempt to model

the label noise process probabilistically. The random label noise modelling is considered to be more generic than the non-random noise modelling. By contrast, the latter is more application specific as one needs to encode prior expertise into the model to reflect how non-random noise would have occurred. Interestingly, we shall see in the subsequent chapters that the random noise model works pretty well even in situations where the randomness assumption does not hold true. For example in the 'Image classification problem' (Chapter 6) where we train a classifier using images returned by an image search engine (think of the search engine as a labeller). Apparently the search engine must have used textual information around an image to determine the label of the image, hence there exist some dependency between an image and its label. Also in microarray classification (Chapter 5) it is very likely that the labelling process is not entirely independent of the microarray measurements, and yet the random noise model is successfully applied.

1.3 Research challenges & aims

Existing research on learning from data with label noise had been carried out in two unconnected streams. The first, mostly done by statisticians concerns theoretical analysis of the effects of mislabelling on traditional classification algorithms in the asymptotic regime (Chhikara and McKeon [1984], Bi and Jeske [2010]). They gave extensive analysis of how label noise affects the performance of the algorithms, but never shed light on how to deal with the problem. For example, the analysis of the effect of label noise on normal discriminant analysis and logistic regression is enlightening, but does not solve the problem, since no modification to make the method robust has been given (Lachenbruch [1974], Bi and Jeske [2010]). Asymptotic analysis is also less useful in machine learning as the possibility is low that we have infinite number of data points.

By contrast, the second stream focuses mainly on how to improve traditional learning algorithms for dealing with label noise in an ad-hoc manner. Many of them that have been

shown to work well are heuristic methods (Malossini et al. [2006], Zhang et al. [2009]). However what is missing is an analysis of why doing so can alleviate the effect of label noise. Such analysis is important so that practitioners can see whether or not the method is readily applicable to their problem, and at which point they need to modify the method in order to use it in their problem.

From the above issues and gaps in the existing literature, we identified the research challenges described in the following three subsections. These have motivated our study in the thesis.

1.3.1 Probabilistic label noise modelling

We position ourselves in a more principled approach than existing heuristics. Our main goal is to develop probabilistic classifiers that can withstand the adverse effects of mislabelling. To begin with, we choose two well known generative and discriminative methods: Normal Discriminant Analysis and Logistic Regression, as our baselines. We build a noise model on these classifiers so that a solid basis of previous results are available for comparison.

Inspired by the probabilistic approach to improving the robustness of Fisher's Discriminant Analysis for binary classification under label noise (Lawrence and Schölkopf [2001]), we think that it would be desirable and useful to generalise such probabilistic modelling to multi-class problems. It is also a challenge to incorporate a similar *robustification* approach in discriminative classification framework to yield a label noise tolerant classifier. As far as classification is concerned, it transpires from the literature that a discriminative classifier is more preferable in general (Ng and Jordan [2001]). This leads us to develop and study a probabilistic noise model in the logistic regression and the multinomial logistic regression in Chapter 4 and Chapter 5.

Equally important to the probabilistic modelling and the development of learning

algorithms is to understand their behaviour theoretically. Such analyses would enable us to say whether the new classifier is superior to the traditional classifier and to quantify the achievable classification performance, with more rigour than we can observe empirically.

1.3.2 Model selection in the presence of label noise

A common practice in model selection is to use cross-validation (Arlot and Celisse [2009]). The technique is legitimate in an idealised setting where the labels are all perfect. Unfortunately the approach might not be suitable when the training labels are erroneous as it inevitably makes use of the noisy labels. To the best of our knowledge, previous work that requires us to perform model selection in the presence of label noise simply assume that there is a *trusted validation* set for cross-validation purpose (Lawrence and Schölkopf [2001], Li et al. [2007]). We found that those trusted validation set could limit the usability of the method in the case where the number of samples is limited. We ask the question if there is a better alternative to the standard cross-validation for model selection which does not require a trusted validation set and yet it puts less emphasis on noisy labels.

To investigate the problem, we will extend the robust logistic regression presented in Chapter 4 for non-linear problems using a well-known kernelising technique (Zhu and Hastie [2001]). In addition, appropriate regularisers are added to incorporate prior knowledge about the classification task at hand. The new robust kernel logistic regression then requires us to perform model selection for both the kernel parameter and the regularisation parameter. It will enable us to gain better understanding about the problem and to come up with a good solution. We shall demonstrate in Chapter 6 that under noisy label scenario the tuning of the regularisation parameter can be accomplished efficiently and effectively using Bayesian regularisation, while the kernel parameter can be optimally selected using Multiple Kernel Learning (MKL) framework.

1.3.3 Limitations of the probabilistic noise model

Suppose that it turns out that we can successfully turn the standard classifier of interest into a robust classifier using the label noise modelling and that model selection problem can be addressed in a satisfactory manner. We are even more interested in finding out if the technique will be general enough so that any classifier can be made robust by using the proposed technique. Specifically, what is the limitation of the robustification technique? We conjecture that the complexity of the target classifier determines whether or not the robustification will be successful. To investigate this question we take an ensemble of classifiers to be our baseline which the noise model will be built on. In theory, an ensemble has an unbounded complexity and thus it will enable us to find some answer to this latter question raised. We will study the behaviour of the robust ensemble machine under label noise in Chapter 7.

In summary, the aim of this research is to come up with principled new algorithms that remedy the problem of learning in the presence of labelling errors, together with a rigorous analysis of the approaches. That is, the goal is not only developing new algorithms but also analysing the obtained algorithms. This will result in a more complete understanding of the advantages and the limitations of the methods, and hence may guide applications or modifications of the proposed methods at a later point.

1.4 Contributions of the thesis

The main contributions of the thesis to the area of learning under labelling errors are summarised as follows.

- The development of label-noise robust Logistic Regression and label-noise robust Multinomial Logistic Regression together with efficient algorithms to learn the models simultaneously with estimating the label flipping probabilities. A convergence

guarantee of the algorithms is also presented.

- A multi-class extension and an analysis of robust Normal Discriminant Analysis, and an analysis of the robust Logistic Regression. The analyses show that under *best case scenario* assumptions both of the robust algorithms output more accurate parameter estimates.

- A novel approach to model selection in the presence of label noise and empirical evidence showing that standard cross-validation is not optimal under such setting. The new approach based on a Bayesian regularisation and Multiple Kernel Learning technique puts less emphasis on noisy labels, yields a superior model and is significantly faster than cross-validation.

- The development of a robust AdaBoost together with empirical results showing that the novel boosting algorithm is resilient to label noise. We also present an efficient way to incorporate extra information about good labels to alleviate the unwanted effect of unbounded complexity of an ensemble machine.

- Demonstration of several important applications of label-noise robust classifiers, including gene expression analysis, learning from crowdsourced data and class topology discovery.

1.5 Outline of the thesis

The thesis is organised as follows. The subsequent chapter presents the background and existing work in the area of learning from data with label noise.

Chapter 3 presents a generalisation of the binary robust Fisher Discriminant Analysis of Lawrence and Schölkopf [2001] to multi-class classification problems. Empirical evaluations are then presented to demonstrate the clear benefit of the label noise model. The

chapter ends with a theoretical analysis of the model. Specifically, we derive a generalisation error bound of the model with shared covariance for binary and multi-class cases. Further analysis concerns the comparison of parameter estimation between the robust model and the traditional model.

Chapter 4 describes a development of the label-noise robust Logistic Regression and its multi-class extension. An efficient multiplicative learning algorithm is presented to learn the model as well as the label flipping parameters. The convergence of the algorithm is also proved in this chapter. In addition, theoretical analysis is given to show that the model is less likely to overfit to noisy label and that its parameter estimates are more accurate compared to the traditional model. The chapter ends with extensive empirical validation using both artificial and real world datasets.

Chapter 5 develops a robust Sparse Bayesian Logistic Regression for an important real-world problem: learning from noisy data in biomedical domain. In this domain a sparsity promoting prior is required to tackle the high dimensional data with limited data samples. A new algorithm is developed specifically for learning the robust model which employs non-convex and non-differentiable objective. The proposed method is applied to microarray datasets which have been reported to contain wrongly labelled samples with biological evidence. The experimental results demonstrate convincingly that the robust model has a performance advantage over the traditional model in this particular type of problems.

Chapter 6 extensively studies model selection in the presence of labelling errors. In this chapter, the robust Logistic Regression and the robust multinomial Logistic Regression are kernelised in order to be able to deal with non-linear classification problems. As a consequence, an optimal regularisation parameter and kernel parameter need to be determined. We adopt a Bayesian regularisation technique for determining a good value of regularisation parameter and introduce a variant of Multiple Kernel Learning framework

as a tool to choose a good kernel parameter.

Chapter 7 investigates the limitations of the label noise modelling approach, where an ensemble machine is used as the subject of the study. We robustify the AdaBoost algorithm using the modelling technique similar to those presented in the previous chapters, and we introduce a new robust boosting method called the rBoost algorithm. Initial results show that the unbounded complexity of the boosting-type classifier could interfere with the learning of the label flipping parameters. We then suggest a way to efficiently incorporate additional information about the label flipping into the new boosting algorithm. Empirical studies reveal that the rBoost algorithm is superior to the original AdaBoost when there is extra information available, and it is no worse than the the original when no such information is provided.

Finally, Chapter 8 summarises the research challenges and questions, as well as the answers to those questions. The conclusion of the study and the outlook for future work finalises the thesis.

1.6 Publications from the research

Submitted & Published refereed journal paper

- Jakramate Bootkrajang and Ata Kabán. Classification of mislabelled microarrays using robust sparse logistic regression. *Bioinformatics*, 29(7):870–877, 2013b

- Jakramate Bootkrajang and Ata Kabán. Learning kernel logistic regression in the presence of class label noise. *Pattern Recognition*, 2013c. revised and resubmitted

Published refereed conference paper

- Jakramate Bootkrajang and Ata Kabán. Multi-class classification in the presence of labelling errors. In *Proceedings of the European Symposium on Artificial Neural*

Networks, Computational Intelligence and Machine Learning, (ESANN'11), pages 345–350, 2011

- Jakramate Bootkrajang and Ata Kabán. Label-noise robust logistic regression and its applications. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, (ECML-PKDD'12)*, pages 143–158, 2012

- Jakramate Bootkrajang and Ata Kabán. Boosting in the presence of labelling errors. In *Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence, (UAI'13)*, 2013a

1.7 Software availability

A software suite containing all the label-noise robust classifiers developed in this study is available at http://cs.bham.ac.uk/∼jxb008/code/ntc.zip

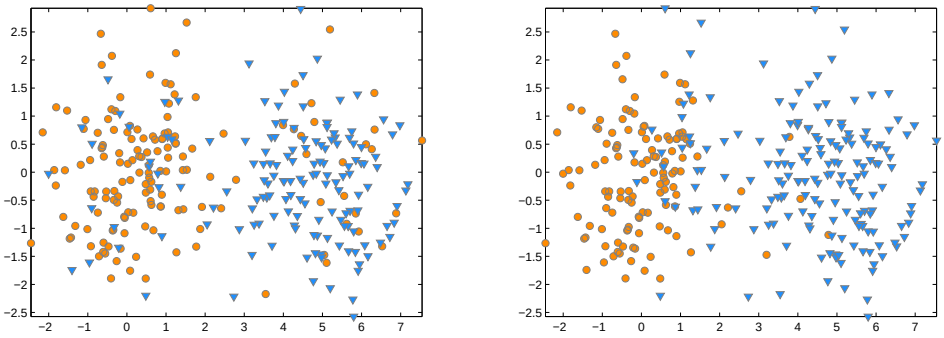
CHAPTER 2

Background and Related Work

This chapter reviews previous studies and research related to learning from data with noisy labels. Section 2.1 gives an overview of types of label noise. Section 2.2 presents studies devoted to understanding the impact of label noise on traditional classifiers. Section 2.3 then presents approaches that have been devised to counteract the effect of mislabelling. Finally existing theoretical analyses regarding learning in the presence of label noise are given in Section 2.4.

2.1 Types of label noise

Class label noise can be categorised into two major types: random noise and non-random noise. Random label noise is a noise that occurs independently from the observation input so that label flipping is equally likely to appear any where within the class. Random noise may originate from channel noise in the communication media, for example, in data acquisition by remote sensing. Recently, the noise results as a by product of crowdsourcing practice where a user is asked to perform a labelling task. The noise maybe random in this case because it is possible that the user wants to increase his productivity by simply giving out the label without actually looking at the input samples. On the other hand,



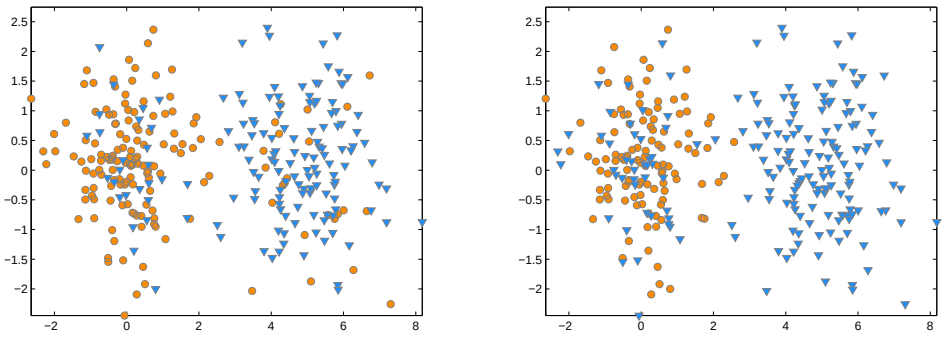
(a) Random noise (b) Non-Random noise

Figure 2.1: Random and non-random label noise

the non-random noise is a noise that depends on the observation such that the sample's features have some influence on the assignment of the label. By nature, non-random noise tends to appear more near the decision boundary where the samples are harder to classify (Takenouchi et al. [2008]). Non-random class noise can also be found in spam filtering tasks where label noise is intentionally designed to mislead the spam classifier as much as possible (Biggio et al. [2011]). In particular, let $f(\mathbf{x})$ be a function representing the probability that the label of an input $\mathbf{x}$ has flipped into the other class. The above two types of noise can be written compactly as,

$$f(\mathbf{x}) = \begin{cases} c_k, & \text{random noise, where } k \text{ is the true class of } \mathbf{x} \\ \frac{1}{|\beta^T \mathbf{x}|}, & \text{non-random noise as a function of distance from decision boundary} \end{cases} \tag{1}$$

where $c_{k=\{0,1\}}$ is a constant, and β is a vector perpendicular to the decision boundary. It can be the weight vector of the logistic regression or the vector connecting the means in the normal discriminant analysis. Figure 2.1 illustrates random and non-random label noises on a 2D classification problem. From the application viewpoint, random noise may be considered to be more generic as it requires fewer assumptions about the nature of the noise. A non-random noise, however, is more application dependent such that prior



(a) Symmetric noise (b) Asymmetric noise

Figure 2.2: Symmetric and asymmetric label noise

knowledge about the nature of noise must be encoded into the model in order to be able to deal with the noise effectively.

We can further divide random noise and non-random noise into two subtypes, namely symmetric noise and asymmetric noise. The symmetric noise is a noise that occurs uniformly across the classes. That is, for example, the probability of label flipping from positive class to negative class and vice-versa is equally likely. Asymmetric noise, however, is a noise that occurs more in one class than the other. Assuming that the classes are distributed symmetrically in the data space, symmetric label noise is *relatively* harmless compared to asymmetric noise as shown by Lachenbruch [1966] and Lugosi [1992]. For illustrative purpose, Figure 2.1 shows symmetric and asymmetric label noise on a 2D classification problem. In the next section we shall see how class noise affects the performance of traditional classifiers.

2.2 Impact of label noise

Although the problem of label errors dates back about half a decade, in the early days label noise had been naively ignored or was understood to be tolerable. Partly because the scale of the classification at that time was not large, so that perfect labelling of the training set was still possible. One of the first efforts to analyse the capability of classical

classifiers to withstand the negative effect of label noise when learning from a corrupted labelled dataset is given in Lachenbruch [1966]. There, the effect of random class noise on linear discriminant function is studied. The study concluded that the performance of the classifier is slightly effected if the label flipping is symmetric.

Later on, Lachenbruch [1974] extends the analysis to cover the non-random label noise case. The criterion for the non-random noise is based on the distance of the observation from the class mean and hence more mislabellings are observed near the decision boundary. The experiments showed that a symmetric noise is also relatively harmless in the non-random noise case. These initial works are classical examples to demonstrate that symmetric noise is less problematic. However, as we shall see in the subsequent chapters, symmetric noise is relatively harmless only when the initial distributions of the classes before label flipping are similar. If they are not, symmetric noise could be as harmful as asymmetric noise.

Chhikara and McKeon [1984] also investigated the effect of the random and non-random misallocation on parameter estimates of classical linear discriminant analysis. Based on the setting in Efron [1975], where the efficiency of linear discriminant analysis in perfect label case was analysed, the study focused on data points from two multivariate normal distributions differing in mean but not covariance, and training examples are misallocated with known label flipping probabilities. The study concluded that the Fisher classification rule (i.e., Bayes rule without class priors) is more robust than the Bayes classification rule when the prior of corrupted data is approximately equal to the true priors before flipping.

More recently Bi and Jeske [2010] compared the effect of noisy labels for linear discriminant analysis and logistic regression. Their setting is similar to that in Efron [1975] and Chhikara and McKeon [1984], the model is multivariate normal populations having a common covariance matrix. They showed that when the noise level is low the error rates

of both methods are only slightly effected. They further showed that logistic regression is more tolerant to class noise than linear discriminant analysis. In the subsequent chapter we shall see if the phenomenon can still be observed in the case where the robust version of the two classifiers are compared (Chapters 4 and 6).

Even though all of these initial work suggest that learning from noisy label can be problematic, in particular for classical learning methods, they did not give any remedy for the problem (i.e. how to avoid the adverse effect and learn a classifier more effectively).

Beyond a single classifier, an ensemble of classifiers is also susceptible to class noise. In this study we focus ourselves on a boosting-type technique called AdaBoost (Freund and Schapire [1995]). Basically AdaBoost (or boosting in general) aims to construct a strong classifier from multiple base learners or weak learners. The definition of weak learnability is the following.

Weak learnability (Kearns and Valiant [1994]) Let C and H be representation classes over X. Then C is weakly learnable from examples by H if there is a polynomial *poly* and an algorithm A taking input δ, with the property that for any target representation $c \in C$, for any target distributions D over X and for any input value $0 \leq \delta \leq 1$, algorithm A halts and outputs a representation $h_A \in H$ that with probability greater than $1 - \delta$, satisfies $err_D(h_A) \leq \frac{1}{2} - \frac{1}{poly(|c|)}$. The algorithm A is then called a weak learner.

In boosting a new base learner is directed towards classifying a data point that has been misclassified by the previous base learners. A final prediction is made by a weighted combination of the predictions from all base learners in the ensemble. From this basic philosophy of boosting it is easy to see that overfitting wrongly labelled samples in the training set is inevitable. Although Freund and Schapire have shown, for the perfect label cases, why the problem is unlikely to happen using a concept related to margin maximisation (Schapire [1999]), overfitting can still be observed in practice. Recently

Long and Servedio [2010] have shown that all convex potential boosters are affected by random label noise. These boosters include AdaBoost, where an exponential loss is used, LogitBoost (Friedman et al. [1998]) where a binary log loss is employed. All of these are optimising a convex function.

2.3 Previous solutions to the problem

Label noise treatment can be categorised into two streams – model-based approaches and filtering approaches. In a nutshell, a model-based approach is an approach that attempts to model the label noise process explicitly, while a filtering type approach is typically concerned with preprocessing the data by removing or relabelling any suspect instances.

2.3.1 Filtering approaches

More often label noise is dealt with in an ad-hoc, heuristic manner. The most common way to deal with label noise is by using the method of data preprocessing. Various techniques such as data filtering, data removal or relabelling have been reported to be effective against label noise.

For example, Barandela and Gasca [2000] introduced an algorithm called 'Depuration' to iteratively modify samples whose class label disagrees with the class label of most of their nearest neighbours and remove such point if the disagreement exceeds a predefined threshold. Sánchez et al. [2003] used several variations of the k-Nearest Neighbour (k-NN) classifier including Depuration and nearest centroid neighbourhood to detect mislabelled examples. Muhlenbach et al. [2004] also proposed an approach based on nearest neighbour framework. The method is designated to remove or relabel the suspect instances. Empirical results obtained from 1-NN classifier suggest that removal gives better result than relabelling.

The above mentioned techniques can be regarded as local learning methods. They

assume that the class label of mislabelled example tends to disagree with the class label of other examples in its close vicinity. The merit of this approach is its robustness to class shape as it makes no assumption on the distribution of data. However, there are some drawbacks associated with a local learning regime. Firstly the method relies on an assumption that most of its neighbours are correctly labelled. Secondly, mislabelling detection is performed locally without propagating decision to other examples (Valizadegan and Tan [2007]). Thirdly, the nearest neighbour method has to set the number of neighbourhood parameter, k, and a disagreement threshold. There is no clear principled way to set these parameters. Lastly the method tends to fail when data lies in a high dimensional space where the notion of neighbourhood becomes unusable. This due to the concentration property that points in very high dimensional space are equally spaced (Beyer et al. [1999]).

Brodley and Friedl [1996] investigated the use of an ensemble of algorithms to detect mislabelled example in the training data. The technique consists of two stages. The first stage involves the identification of mislabelled examples using consensus filter and majority vote. A consensus filter marks an example as being mislabelled only if it is misclassified by all the classifiers in the ensemble. A less conservative method is to consider an example to be mislabelled if its label disagrees with the majority vote of the classifiers. For the second stage, a classical classification algorithm is applied to the cleaned dataset. Their experimental results showed that data filtering improves classification accuracy for noise level up to 20%. Pechenizkiy et al. [2006] empirically studied the negative effect of label noise on supervised learning in medical domains. The work claimed that feature extraction method can be used to diminish the effects of label noise during the learning process.

Zhu et al. [2003] introduced a method based on partitioning a large data set into smaller subsets and building a corresponding classifier for each subset. The classification

rules derived from each subset can then be used to filter the whole dataset. Jiang and Zhou [2004] considered an ensemble of neural networks to edit class labels for k-NN classifier. A neural network ensemble trained from original dataset generates new class labels which are then used to replace original class labels. Venkataraman et al. [2004] also employed an ensemble approach using the support vector machine for facial recognition application. SVMs are trained on different feature subsets which results in different discriminating subspaces. They claimed that the method is effective against mislabelling. However, the problem with the approach is that the classifiers in an ensemble are built from a training set that still contains mislabelled data. Another problem is that this method requires that 1) the errors committed by the base classifiers are independent of each other, and 2) the error rates of the base classifiers should be less than 50%. In many cases these requirements cannot be fulfilled trivially as pointed out by Valizadegan and Tan [2007]. Therefore, Valizadegan and Tan [2007] reformulated label noise detection as an optimisation problem and introduced a kernel-based approach for filtering the mislabelled examples. The approach also has a mechanism to propagate label error information. They claimed that their approach is more effective than nearest-neighbour and ensemble-based schemes. However, their approach is limited to binary problems.

The classical techniques to avoid overfitting can also be applied explicitly in order to learn from mislabelled data. John [1995] proposed an iterative method to clean the dataset using C4.5 algorithm (Quinlan [1996]). A decision tree is built from an original dataset then pruned using C4.5 algorithm to avoid overfitting on noisy examples. Confusing instances are defined to be those incorrectly classified by the pruned decision tree. Those instances are then removed from the training set.

Dietterich and Bakiri [1995] developed a method for learning classifiers for multiple classes in which error-correcting scheme are employed as a distributed output representation. They viewed classification as a communication problem in which the identity of the

correct output class for a new example is being transmitted over a noisy channel. Empirical results show that the error-correction scheme can also be used to improve classification performance.

The recent trend in mining from biological datasets poses a new challenge. As Malossini et al. [2006] pointed out that traditional mislabelling detection is not robust enough to be used in the case where the number of features is much larger than the number of training instances. They presented an algorithm for detecting possible mislabelled points in microarray dataset. The key structure is the use of leave-one-out perturbation matrix, which is used to measure stability of the label of an example and produced the list of suspect examples. Those examples are then relabelled or removed and the cleaned dataset is passed along as a training set to final classifier. Zhang et al. [2009] extended the work by Malossini et al. [2006] by introducing the leave-one-out regression matrix which outputs stability measurements in a more fine-grained manner. They claimed that their approach outperforms the original perturbation measurement method of Malossini et al. [2006]. Interestingly these two approaches implicitly take into account the effect of having a point in the dataset versus having it removed. A limitation of this approach is that it assumes that the points taking part in the leave-one-out validation have correct labels. This is somewhat analogous to the limitation of the nearest neighbour type methods as pointed out earlier. In addition, one must be aware that any errors made in separate stages of analysis will necessarily accumulate.

2.3.2 Model-based approaches

In contrast to the filtering approach, a model-based approach is more principled and more transparent as it includes an explicit model of the mislabelling process as an integral part of modelling the data. To the best of our knowledge, there were only few attempts to approach the problem by modelling the noise process explicitly. Norton and Hirsh [1992]

suggested learning from noisy data by incorporating prior knowledge of the noise process. In the study, the posterior probability of each hypothesis in the hypothesis space being searched is computed in order to prune out bad hypotheses. They empirically found that their maximum a posteriori (MAP) approach is superior to the C4.5 pruning algorithm. Chittineni [1982] incorporated label noise parameters in his likelihood function and derived a maximum likelihood estimator for estimating the misclassification parameters from labelled and unlabelled data. He also introduced simple model to identify mislabelling instances in terms of thresholds on a linear discriminant functions for both two-class and multi-class cases. His approach is rather specialised to the semi-supervised learning.

Lawrence and Schölkopf [2001] incorporated probabilistic noise model in their Kernel Fisher Discriminant for binary classification. Assuming that the data class distributions are Gaussian, they empirically showed that the classification accuracy improves over the model that assumes no label noise. Based on the same model, Li et al. [2007] carried out extensive experiments on more complex datasets, which convincingly demonstrated the value of explicit modelling. The extension of the model to a multi-class setting will be presented in Chapter 3. This extension has further motivated the recent development of a label noise-tolerant Hidden Markov Model to improve segmentation (Frénay et al. [2011]).

While all these works demonstrate the great potential and flexibility of a model based approach, they most fall in the category of generative methods. For classification problems, discriminative methods are also of interest, and similar algorithmic developments for discriminative classifiers are still limited. Magder and Hughes [1997] studied logistic regression with known label flipping probabilities. Hausman et al. [1998] has given a foundation of a statistical model for binary classification problem but provide no algorithmic solution to the learning of label noise parameters.

Recently Raykar et al. [2010] proposed an EM algorithm to learn the latent model

logistic regression similar to that discussed in Hausman et al. [1998] for data with multiple instances of noisy labels. In this thesis we will present a more efficient gradient-based algorithm to optimise a latent variable logistic regression model for problems where only a single set of labels is available (Chapter 4). A sparse extension of the model will also be explored in Chapter 5. In a different but related context, Amini and Gallinari [2005] used noise modelling to approach semi-supervised learning. They randomly assign labels to an unlabelled training set and use the noise model to recover the true labels in order to enlarge the labelled dataset. Krithara et al. [2008] further used the framework in Amini and Gallinari [2005] to extend Probabilistic Latent Semantic Analysis (Hofmann [2001]) to incorporate mislabelling error. Empirical results show that their method is superior to previous methods that use a model without mislabelling error model.

There are also attempts to counteract label noise in the context of boosting. These boosters include the LogitBoost (Friedman et al. [1998]) that optimises the binary log-loss, the Gentle-AdaBoost (Friedman et al. [1998]) that is more stable because of a more conservative update step, and the Modest-AdaBoost (Vezhnevets and Vezhnevets [2005]) which penalises the ensemble when it makes a correct prediction on previously correctly predicted instances. Krieger et al. [2001] proposed a boosting algorithm in which bagging is combined with boosting to average out the adverse effect of noisy labelled data. There is also a heuristic approach by Karmaker and Kwek [2006] where points that are too difficult, i.e., those with very high weights, are removed from the training set according to a predefined threshold. Yasui et al. [2004] proposed an EM-boosting algorithm to deal with label noise in high dimensional biological data. They replaced the true label in the boosting objective with the observed label and treated the true label as missing value. An EM-type algorithm is then used to optimise the new objective. Some improvements in misclassification rates are observed with the new algorithm. However, rather curiously, all of the existing robust boosters are still optimising a convex exponential loss – which

is shown to be non-robust against random misclassification (Long and Servedio [2010]).

Motivated by the finding of Long and Servedio, Freund [2009] proposed a more robust boosting algorithm which optimises a non-convex potential function instead of the traditional exponential loss function. The general idea is to incorporate early stopping as well as a mechanism to give up if the instance is to far away on the wrong side of the decision boundary. The approach shows promising results but unfortunately the boosting process becomes more complicated in that it also introduces a free parameter that has to be tuned. Freund suggests using cross-validation to tune the parameter however we can not rely on the cross-validation if our labels are noisy, unless if we have a trusted validation set with correct labels. Takenouchi et al. [2008] proposed a multi-class robust boosting based on a Bregman U-divergence loss for non-random misclassification. The theoretical and empirical results are encouraging. However, the algorithm also has free-parameter that needs to be tuned and the author simply used standard K-fold cross-validation for that purpose. These shortcomings of the traditional boosting algorithms will be addressed and discussed in more depth in Chapter 7.

2.4 Existing theoretical analyses

In the context of learning theory, there are some work devoted to studying the problem of Probably Approximately Correct (PAC) learning from training examples with noisy labels (Angluin and Laird [1988], Kearns and Li [1988]). The PAC learning framework (Valiant [1984]) can be defined as follows.

Let X be an instance space with governing distribution D such that each $\mathbf{x} \in X$ is assigned with a probability. Define a concept class C to be a family of concepts $c_1, c_2, ..., c_n$. The task is to learn a concept of interest, denoted c_* using an algorithm A. The algorithm A will identify the target concept via a series of queries to the so-called oracle T. In each query, the oracle T will randomly draw a sample $\mathbf{x}$ from the distribution D and returns

$(\mathbf{x}, +)$ if $\mathbf{x} \in c_*$ and $(\mathbf{x}, -)$ if $\mathbf{x} \notin c_*$. The learning is said to be *Probably Approximately Correct* if given a tolerance parameter ϵ and a confidence parameter δ after a series of queries to the oracle an algorithm A outputs a concept c_a satisfying,

$$p[d(c_a, c_*) \geq \epsilon] \leq \delta \tag{2}$$

Here, $d(c_a, c_*)$ is defined to be $\sum_{\mathbf{x} \in c_a \triangle c_*} p_D(\mathbf{x})$, where $\triangle$ is symmetric difference operator. In other words, $d()$ defined an error measure that counts up the misclassification of sample $\mathbf{x}$.

Focused on learning a class of k-Conjunctive Normal Form, Angluin and Laird [1988] found that standard algorithms that are PAC-learnable under clean data can often be generalised to handle a certain amount of random noise providing that there are plenty of training examples. In this setting, the oracle T is now not faithful such that it returns $(\mathbf{x}, +)$ if $\mathbf{x} \in c_*$ with probability $1-\beta$ and returns $(\mathbf{x}, -)$ with probability β even if $\mathbf{x} \in c_*$, hence T is an imperfect teacher. They came up with an upper bound for the number of training instances required to PAC-learn the concept class. The only constraint is that the noise level, β must be lower than one half.

Kearns and Li [1988] also studied learning from imperfect data under PAC framework. The study concerns PAC learning from examples with a worst-case error model in which label noise is maliciously generated by an adversary in order to fool the learning algorithm as much as possible. They follow the assumption from Valiant [1985] that there is a fixed probability $0 \leq \beta < 1/2$ of error occurring independently but its nature is arbitrary. They obtain an upper bound on the maximum error rate at which the concept class such as, monomial, k-CNF and k-DNF are PAC-learnable using an algorithm A. Interestingly, in their study, the error model did not distinguish between label and attribute error. When considering only the case of labelling errors, those bounds are inappropriate as they are

too general.

Sloan [1995] extends the work of Angluin and Laird [1988] by considering four possible types of noise (label and attribute) in data for PAC learning. One of the case is where the noisy label is non-random. He referred to this kind of noise as a malicious misclassification noise. He showed theoretically that malicious misclassification noise is no more harmful than random misclassification noise. Gentile and Helmbold [2001] presented an information-theoretic approach for obtaining a lower bound for the number of training examples required for PAC learning in the presence of label noise. They showed that the obtained bound is tighter than the previously known lower bound. Lugosi [1992] theoretically showed that symmetric label noise is relatively harmless compared to asymmetric ones.

From the literature review, it can be seen that existing methods still have some inadequacies that limit their applicability, and it is apparent that there is still some room for improvement. As identified and described earlier in Chapter 1, we will try to address some of these limitations in this thesis. This includes a novel development of robust generative and discriminative classifier for both binary and multi-class problems together with several useful extensions, a novel approach to model selection in a noisy label environment and a robust boosting algorithm. Hopefully the novel developments proposed will correct the inadequacies and should serve as alternatives for the classification practitioner whose task contains label errors. The transparency of our approaches as well as their theoretical analysis would also enable practitioners to further modify or extend our methods to better suit their needs with ease.

CHAPTER 3

Multi-class Robust Normal Discriminant Analysis

One of the most transparent and widely used classification paradigms in machine learning is generative modelling. Under a generative assumption one tries to learn the joint distribution that generates the input-label pairs from a training dataset one has at hand. Assuming some parametric form for the distribution and estimating its parameters, class membership can then be calculated using the Bayes rule. In the presence of label noise, the noise can interfere with the parameter estimation and this leads to sub-optimal estimated distribution.

In this chapter, we take a generative classifier called multi-class quadratic and linear normal discriminant analysis and extend it to incorporate a model of the mislabelling process. We will refer to this model as the 'robust Normal Discriminant Analysis' (rNDA) to distinguish it from the classical Normal Discriminant Analysis (NDA). We shall demonstrate the clear benefits of rNDA in terms of improved estimates of the class-conditional distributions, and improved classification performance on both synthetic and real-world multi-class problems in comparison to NDA and a previous model-free approach (Depuration) (Barandela and Gasca [2000]). Some theoretical analysis of the performance of rNDA is also presented. We give a generalisation error bound of rNDA conditioned on a

fixed training set, and we also show that its parameter estimation is more accurate than that of the classical model.

3.1 The model

In Lawrence and Schölkopf [2001] Fisher discriminant analysis in the presence of label error is discussed. Inspired by this work, we incorporate a probabilistic model of the label flipping process into multi-class normal discriminant analysis. Consider a training set $S = (\mathbf{x}_n, \tilde{y}_n)_{n=1}^N$, where $\mathbf{x}_n$ are the observation input vectors and $\tilde{y}_n \in \{1, \ldots, K\}$ are their given (possibly noisy) class labels. We start by formulating a mixture model for this data. Denote the model's parameters as $\Theta = \{\theta_j\}_{j=1}^K$, we write the data likelihood as

$$\mathcal{L}(\Theta) = \prod_{n=1}^{N} p(\mathbf{x}_n|\tilde{y}_n, \theta_{\tilde{y}_n}) p(\tilde{y}_n) \tag{1}$$

Since there exist label noise in the data, learning the model from the observed labels $\tilde{y}$ is no longer valid. To address the problem, we introduce a latent variable y to represent the true label associated with the input vector $\mathbf{x}$. Under this assumption the data likelihood can be expressed as:

$$\mathcal{L}(\Theta) = \prod_{n=1}^{N} \sum_{y} p(\mathbf{x}_n|y, \tilde{y}_n, \theta_{\tilde{y}_n}) p(y, \tilde{y}_n) \tag{2}$$

From this formulation we have two ways to write down the joint probability of pair of true and observed labels.

$$p(y, \tilde{y}) = p(y|\tilde{y})p(\tilde{y}) = p(\tilde{y}|y)p(y) \tag{3}$$

These two factorisations, although are equivalent by definition, correspond to two different graphical models shown in Figure 3.1. The first, Figure 3.1(a), is more intuitive when one wants to explain how the data was generated. In this case the input vector and observed

(a) (b)

Figure 3.1: The variable y and $\tilde{y}$ represent true and observed label respectively. Naturally, an arrow points from y towards $\tilde{y}$ indicating that a wrong label originates from a true label (a). However by definition we can reverse the direction of the arrow to get more analytical friendly representation (b).

label are generated according to the true label. The second, Figure 3.1(b), is less intuitive in the context of generative modelling because it looks as if the true label was originated from the corrupted label.

As such, we proceed to treat the joint probability $p(y, \tilde{y})$ according to the first graphical model.

$$\mathcal{L}(\Theta) = \prod_{n=1}^{N} \sum_{j=1}^{K} p(\mathbf{x}_n | y = j, \theta_j) p(y = j, \tilde{y}_n) \tag{4}$$

$$= \prod_{n=1}^{N} \sum_{j=1}^{K} p(\mathbf{x}_n | y = j, \theta_j) p(\tilde{y}_n | p = y) p(y = j) \tag{5}$$

Recall that we have made an assumption that the label noise is random, i.e. it occurs independently from the observation features. Therefore we can drop $\tilde{y}$ from $p(\mathbf{x}_n | y, \tilde{y}; \theta_{\tilde{y}_n})$ using the fact that $\tilde{y}_n$ becomes conditionally independent from $\mathbf{x}_n$ after knowing y. Since the true label is unobservable we then marginalise the latent variable y in order to get the data likelihood that we can work with. Notice that now we are finding parameters associated with the true label instead of those associated with the observed label.

Further, by denoting the observed class membership vector of the n^{th} point using the indicator function $\mathbb{1}(\tilde{y}_n = k)$ we get the data log-likelihood as:

$$\mathcal{L}(\Theta, \Gamma) = \sum_{k=1}^{K} \mathbb{1}(\tilde{y}_n = k) \sum_{n=1}^{N} \log \sum_{j=1}^{K} p(\mathbf{x}_n | y = j; \theta_j) p(\tilde{y}_n = k | y = j) p(y = j) \tag{6}$$

We see that we have transformed the likelihood w.r.t. the observed label into the log-likelihood w.r.t. the true label by adding extra weighting coefficient, namely $p(\tilde{y}|y)$. This is a probabilistic factor that takes into account the random label flipping process. We define $\gamma_{jk} \stackrel{def}{=} p(\tilde{y}_n = k|y = j)$ to be a probability that the label is flipped from the true class j to the observed class k. These parameters form a label transition table which we will refer to as the gamma table, Γ. The table is summarised in Table 3.1.

		$\tilde{y}$					
		1	2	$\cdots$	k	$\cdots$	K
	1	γ_{11}	γ_{12}	$\cdots$	γ_{1k}	$\cdots$	γ_{1K}
	2	γ_{21}	γ_{22}	$\cdots$	γ_{2k}	$\cdots$	γ_{2K}
	$\vdots$				$\vdots$		
y	j	γ_{j1}	γ_{j2}	$\cdots$	γ_{jk}	$\cdots$	γ_{jK}
	$\vdots$				$\vdots$		
	K	γ_{K1}	γ_{K2}	$\cdots$	γ_{Kk}	$\cdots$	γ_{KK}

Table 3.1: Probabilistic relationship between the observed label and the true label.

3.2 The learning algorithm

In this section we proceed to derive the learning algorithm for our rNDA. Unfortunately, it is difficult and cumbersome to deal with logarithms of a sum as in Eq.(6). We will instead employ the Expectation-Maximisation (EM) methodology (Dempster et al. [1977]) to optimise the model. Following the EM algorithm, we construct a lower bound for Eq.(6) by rewriting $\log \sum_{j=1}^{K} p(\mathbf{x}_n|y = j; \theta_j)p(\tilde{y}_n = k|y = j)p(y = j)$ as

$$\log \sum_{j=1}^{K} q(y = j) \frac{p(\mathbf{x}_n|y = j; \theta_j)p(\tilde{y}_n = k|y = j)p(y = j)}{q(y = j)} \tag{7}$$

where $q(\cdot)$ is an arbitrary probability distribution of the latent variable y. By Jensen's inequality (Appendix B.2) and the concavity of logarithm, we derive the lower bound of

the above expression with

$$\log \sum_{j=1}^{K} q(y=j) \frac{p(\mathbf{x}_n|y=j;\theta_j)p(\tilde{y}_n=k|y=j)p(y=j)}{q(y=j)} \geq$$
$$\sum_{j=1}^{K} q(y=j) \log \frac{p(\mathbf{x}_n|y=j;\theta_j)p(\tilde{y}_n=k|y=j)p(y=j)}{q(y=j)} \tag{8}$$

Once we have constructed the lower bound we can go a step further by finding the optimal lower bound, which can be found by maximising Eq.(8) w.r.t. the free parameter, $q(y=j)$. Since $q(\cdot)$ is a probability we employ a Lagrange multiplier λ to enforce the constraint that the probability sums to one, which gives us the Lagrangian.

$$\begin{aligned}\mathcal{G} = &\sum_{j=1}^{K} q(y=j) \log p(\mathbf{x}_n|y=j;\theta_j)p(\tilde{y}_n=k|y=j)p(y=j) \\ &- \sum_{j=1}^{K} q(y=j) \log q(y=j) + \lambda\{1 - \sum_{j=1}^{K} q(y=j)\}\end{aligned} \tag{9}$$

Taking derivative w.r.t. $q()$, equating to 0, we solve for λ.

$$\begin{aligned}\frac{\partial \mathcal{G}}{\partial q(y=j)} &= \log\Big\{p(\mathbf{x}_n|y=j;\theta_j)p(\tilde{y}_n=k|y=j)p(y=j)\Big\} - \log q(y=j) - 1 - \lambda = 0 \\ \log q(y=j) &= \log\Big\{p(\mathbf{x}_n|y=j;\theta_j)p(\tilde{y}_n=k|y=j)p(y=j)\Big\} - 1 - \lambda \\ q(y=j) &= p(\mathbf{x}_n|y=j;\theta_j)p(\tilde{y}_n=k|y=j)p(y=j) \cdot e^{-(\lambda+1)} \\ \sum_{j=1}^{K} q(y=j) &= \sum_{j=1}^{K} p(\mathbf{x}_n|y=j;\theta_j)p(\tilde{y}_n=k|y=j)p(y=j) \cdot e^{-(\lambda+1)} \\ 1 &= \sum_{j=1}^{K} p(\mathbf{x}_n|y=j;\theta_j)p(\tilde{y}_n=k|y=j)p(y=j) \cdot e^{-(\lambda+1)} \\ \lambda &= \log \sum_{j=1}^{K} p(\mathbf{x}_n|y=j;\theta_j)p(\tilde{y}_n=k|y=j)p(y=j) - 1\end{aligned} \tag{10}$$

Plugging the expression of λ into Eq.(8), and solving for $q(y=j)$ we have

$$\begin{aligned} q(y=j) &= \frac{p(\mathbf{x}_n|y=j;\theta_j)p(\tilde{y}_n=k|y=j)p(y=j)}{\sum_{j=1}^{K} p(\mathbf{x}_n|y=j;\theta_j)p(\tilde{y}_n=k|y=j)p(y=j)} \\ &= p(y=j|\mathbf{x}_n,\tilde{y}_n=k) \end{aligned} \tag{11}$$

which is essentially the posterior probability of the true label. Now plugging Eq.(11) into Eq.(8) we get the expected complete data log-likelihood or the so-call Q function.

$$\begin{aligned} \mathcal{Q} = \sum_{k=1}^{K} \mathbb{1}(\tilde{y}_n=k)\Big\{ &\sum_{n=1}^{N}\sum_{j=1}^{K} p(y=j|\mathbf{x}_n,\tilde{y}_n=k)\log p(\mathbf{x}_n|y=j;\theta_j) \\ &+ \sum_{n=1}^{N}\sum_{j=1}^{K} p(y=j|\mathbf{x}_n,\tilde{y}_n=k)\log p(\tilde{y}_n=k|y=j) \\ &+ \sum_{n=1}^{N}\sum_{j=1}^{K} p(y=j|\mathbf{x}_n,\tilde{y}_n=k)\log p(y=j)\Big\} \end{aligned} \tag{12}$$

The E-step or *Expectation step* consists of the calculation of the posterior distribution of the latent variable y,

$$p(y=j|\mathbf{x}_n,\tilde{y}_n=k) = \frac{p(\mathbf{x}_n|y=j,\theta_j)p(\tilde{y}_n=k|y=j)p(y=j)}{\sum_{j=1}^{K} p(\mathbf{x}_n|y=j,\theta_j)p(\tilde{y}_n=k|y=j)p(y=j)} \tag{13}$$

The M-step or *Maximisation step* is the optimisation of Eq.(12) w.r.t. class parameters θ_j. Assuming that data was generated by a Gaussian distribution with mean $\boldsymbol{\mu}_j$ and covariance $\boldsymbol{\Sigma}_j$, we will now derive the update equations for these parameters.

3.2.1 Updating the mean

To get an update expression for the mean, we take the derivative of Eq.(12) w.r.t. $\boldsymbol{\mu}_j$, equate to 0 and solve for the mean:

$$\begin{aligned}
\frac{\partial \mathcal{Q}}{\partial \boldsymbol{\mu}_j} &= \sum_{k=1}^{K} \mathbb{1}(\tilde{y}_n = k) \sum_{n=1}^{N} p(y = j|\mathbf{x}_n, \tilde{y}_n = k) \cdot \frac{1}{2} \cdot 2\boldsymbol{\Sigma}_j^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_j) \\
0 &= \sum_{k=1}^{K} \mathbb{1}(\tilde{y}_n = k) \sum_{n=1}^{N} p(y = j|\mathbf{x}_n, \tilde{y}_n = k)\mathbf{x}_n - \sum_{k=1}^{K} \mathbb{1}(\tilde{y}_n = k) \sum_{n=1}^{N} p(y = j|\mathbf{x}_n, \tilde{y}_n = k)\boldsymbol{\mu}_j \\
\boldsymbol{\mu}_j &= \frac{\sum_{k=1}^{K} \mathbb{1}(\tilde{y}_n = k) \sum_{n=1}^{N} p(y = j|\mathbf{x}_n, \tilde{y}_n = k)\mathbf{x}_n}{\sum_{k=1}^{K} \mathbb{1}(\tilde{y}_n = k) \sum_{n=1}^{N} p(y = j|\mathbf{x}_n, \tilde{y}_n = k)}
\end{aligned} \tag{14}$$

3.2.2 Updating the covariance

Similarly for the covariance matrix, we take the derivative of Eq.(12) w.r.t. $\boldsymbol{\Sigma}_j$, equate to 0 and solve for the covariance. An expression to update the covariance matrix turns out to be:

$$\begin{aligned}
\frac{\partial \mathcal{Q}}{\partial \boldsymbol{\Sigma}_j} &= -\sum_{k=1}^{K} \mathbb{1}(\tilde{y}_n = k) \sum_{n=1}^{N} p(y = j|\mathbf{x}_n, \tilde{y}_n = k) \cdot \frac{1}{2}\boldsymbol{\Sigma}_j^{-1} \\
&\quad + \sum_{k=1}^{K} \mathbb{1}(\tilde{y}_n = k) \sum_{n=1}^{N} p(y = j|\mathbf{x}_n, \tilde{y}_n = k) \cdot \frac{1}{2} \cdot \boldsymbol{\Sigma}_j^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_j)(\mathbf{x}_n - \boldsymbol{\mu}_j)^T\boldsymbol{\Sigma}_j^{-1} \\
\boldsymbol{\Sigma}_j &= \frac{\sum_{k=1}^{K} \mathbb{1}(\tilde{y}_n = k) \sum_{n=1}^{N} p(y = j|\mathbf{x}_n, \tilde{y}_n = k)(\mathbf{x}_n - \boldsymbol{\mu}_j)(\mathbf{x}_n - \boldsymbol{\mu}_j)^T}{\sum_{k=1}^{K} \mathbb{1}(\tilde{y}_n = k) \sum_{n=1}^{N} p(y = j|\mathbf{x}_n, \tilde{y}_n = k)}
\end{aligned} \tag{15}$$

3.2.3 Updating the class prior

For the class prior probability, we add a Lagrange multiplier to ensure that $\sum_{j=1}^{K} p(y = j) = 1$. Then differentiating $\mathcal{Q}$, rearranging and solving the equation yields the update equation

for the prior.

$$\frac{\partial \mathcal{Q}}{\partial p(y=j)} = \sum_{k=1}^{K} \mathbb{1}(\tilde{y}_n = k) \sum_{n=1}^{N} p(y=j|\mathbf{x}_n, \tilde{y}_n = k) \cdot \frac{1}{p(y=j)} = \lambda$$

$$\lambda \sum_{j=1}^{K} p(y=j) = \sum_{k=1}^{K} \mathbb{1}(\tilde{y}_n = k) \sum_{n=1}^{N} \sum_{j=1}^{K} p(y=j|\mathbf{x}_n, \tilde{y}_n = k)$$

$$\lambda = \sum_{k=1}^{K} \mathbb{1}(\tilde{y}_n = k) \sum_{n=1}^{N} \sum_{j=1}^{K} p(y=j|\mathbf{x}_n, \tilde{y}_n = k) \tag{16}$$

Thus we have,

$$p(y=j) = \pi_j = \frac{\sum_{k=1}^{K} \mathbb{1}(\tilde{y}_n = k) \sum_{n=1}^{N} p(y=j|\mathbf{x}_n, \tilde{y}_n = k)}{\sum_{k=1}^{K} \mathbb{1}(\tilde{y}_n = k) \sum_{n=1}^{N} \sum_{j=1}^{K} p(y=j|\mathbf{x}_n, \tilde{y}_n = k)} \tag{17}$$

3.2.4 Updating the label flipping probabilities

Lastly, we give the expression to update the label flipping probability. First, note that a row in the gamma table sums to one. We then construct a Lagrangian from our Q-function to ensure that $\sum_{k=1}^{K} \gamma_{jk} = 1$.

$$\begin{aligned}
\mathcal{G} = & \sum_{k=1}^{K} \mathbb{1}(\tilde{y}_n = k) \sum_{n=1}^{N} \sum_{j=1}^{K} p(y=j|\mathbf{x}_n, \tilde{y}_n = k) \log p(\mathbf{x}_n|y=j;\theta_j) \\
& + \sum_{k=1}^{K} \mathbb{1}(\tilde{y}_n = k) \sum_{n=1}^{N} \sum_{j=1}^{K} p(y=j|\mathbf{x}_n, \tilde{y}_n = k) \log \gamma_{jk} - \lambda(1 - \sum_{k=1}^{K} \gamma_{jk}) \\
& + \sum_{k=1}^{K} \mathbb{1}(\tilde{y}_n = k) \sum_{n=1}^{N} \sum_{j=1}^{K} p(y=j|\mathbf{x}_n, \tilde{y}_n = k) \log p(y=j)
\end{aligned} \tag{18}$$

Taking derivative with respect to γ_{jk}, equating to 0 and solving for γ_{jk} yields update equation for the flip probability.

$$\frac{\partial \mathcal{G}}{\partial \gamma_{jk}} = \sum_{n=1}^{N} p(y=j|\mathbf{x}_n, \tilde{y}_n = k) \frac{1}{\gamma_{jk}} - \lambda = 0$$

$$\sum_{n=1}^{N} p(y=j|\mathbf{x}_n, \tilde{y}_n = k) = \lambda\gamma_{jk}$$
$$\sum_{n=1}^{N}\sum_{k=1}^{K} p(y=j|\mathbf{x}_n, \tilde{y}_n = k) = \lambda\sum_{k=1}^{K}\gamma_{jk}$$
$$\sum_{n=1}^{N}\sum_{k=1}^{K} p(y=j|\mathbf{x}_n, \tilde{y}_n = k) = \lambda \tag{19}$$

$$\gamma_{jk} = \frac{\sum_{n=1}^{N} p(y=j|\mathbf{x}_n, \tilde{y}_n = k)}{\sum_{n=1}^{N}\sum_{k=1}^{K} p(y=j|\mathbf{x}_n, \tilde{y}_n = k)} \tag{20}$$

Algorithm 1 summarises the steps to learn the multi-class robust Normal Discriminant Analysis model.

Algorithm 1 Learning algorithm for rNDA

Input: A set of Gaussian parameters $\Theta = \{\boldsymbol{\mu}_{j=1:K}, \pi_{j=1:K}, \boldsymbol{\Sigma}_{j=1:K}\}$, Γ

Initialise $\boldsymbol{\mu}_j \leftarrow \frac{1}{N}\sum_{n=1}^{N} \mathbb{1}(y_n = j)\mathbf{x}_n$

Initialise $\pi_j \leftarrow \frac{1}{N}\sum_{n=1}^{N} \mathbb{1}(y_n = j)$

Initialise $\boldsymbol{\Sigma}_j \leftarrow \frac{1}{N}\sum_{n=1}^{N} \mathbb{1}(y_n = j)(\mathbf{x}_n - \boldsymbol{\mu}_j)(\mathbf{x}_n - \boldsymbol{\mu}_j)^T$

while Iteration < MaxIteration **do**

Update $\boldsymbol{\mu}_j$ using Eq.(14)

Update $\boldsymbol{\Sigma}_j$ using Eq.(15)

Update π_j using Eq.(17)

Update Γ using Eq.(20)

end while

Output: Optimised Gaussian parameters Θ. Optimised Γ.

3.3 Classifying a novel point

We now have all the equations we need to estimate the parameters. One question left unanswered is how are we going to classify an unseen example $\mathbf{x}_q$. So far what we can calculate is the posterior probability which is conditioned on $\tilde{y}_n$ However, for an unseen data point $\tilde{y}$ is unknown, and hence our formulation is not directly applicable. There are two approaches, which turn out to be equivalent, to get the posterior probability of y.

The first way is to omit $\tilde{y}$ and calculate $p(y = j|\mathbf{x}_q; \theta_j)$ as follows

$$p(y = j|\mathbf{x}_q) = \frac{p(\mathbf{x}_q|y = j, \theta_j)p(y = j)}{\sum_{j=1}^{K} p(\mathbf{x}_q|y = j, \theta_j)p(y = j)} \tag{21}$$

The second way is to start with $p(y = j|\mathbf{x}_q, \tilde{y})$ and marginalising out the observed label, $\tilde{y}$.

$$\begin{aligned}
p(y = j|\mathbf{x}_q) &= \sum_{k=1}^{K} p(y = j|\mathbf{x}_q, \tilde{y} = k)p(\tilde{y} = k|\mathbf{x}_q) \\
&= \sum_{k=1}^{K} \Big(\frac{p(\mathbf{x}_q|y = j)p(\tilde{y} = k|y = j)p(y = j)}{\sum_{i=1}^{K} p(\mathbf{x}_q|y = i)p(\tilde{y} = k|y = i)p(y = i)} \\
&\times \frac{\sum_{i=1}^{K} p(\mathbf{x}_q|y = i)p(\tilde{y} = k|y = i)p(y = i)}{\sum_{l=1}^{K}\sum_{i=1}^{K} p(\mathbf{x}_q|y = l)p(\tilde{y} = i|y = l)p(y = l)} \Big) \\
&= \sum_{k=1}^{K} \left(\frac{p(\mathbf{x}_q|y = j)p(\tilde{y} = k|y = j)p(y = j)}{\sum_{l=1}^{K}\sum_{i=1}^{K} p(\mathbf{x}_q|y = l)p(\tilde{y} = i|y = l)p(y = l)} \right) \\
&= \frac{p(\mathbf{x}_q|y = j)\sum_{k=1}^{K} p(\tilde{y} = k|y = j)p(y = j)}{\sum_{l=1}^{K} p(\mathbf{x}_q|y = l)\sum_{i=1}^{K} p(\tilde{y} = i|y = l)p(y = l)} \\
&= \frac{p(\mathbf{x}_q|y = j)p(y = j)}{\sum_{l=1}^{K} p(\mathbf{x}_q|y = l)p(y = l)}
\end{aligned} \tag{22}$$

where we have used,

$$\begin{aligned}
p(\tilde{y} = k|\mathbf{x}_q) &= \frac{p(\mathbf{x}_q|\tilde{y} = k)p(\tilde{y} = k)}{\sum_{l=1}^{K} p(\mathbf{x}_q|\tilde{y} = l)p(\tilde{y} = l)} \\
&= \frac{\sum_{i=1}^{K} p(\mathbf{x}_q|y = i, \tilde{y} = k)p(\tilde{y} = k|y = i)p(y = i)}{\sum_{l=1}^{K}\sum_{i=1}^{K} p(\mathbf{x}_q|y = l, \tilde{y} = i)p(\tilde{y} = i|y = l)p(y = l)} \\
&= \frac{\sum_{i=1}^{K} p(\mathbf{x}_q|y = i)p(\tilde{y} = l|y = i)p(y = i)}{\sum_{l=1}^{K}\sum_{i=1}^{K} p(\mathbf{x}_q|y = l)p(\tilde{y} = i|y = l)p(y = l)}
\end{aligned} \tag{23}$$

From Eq.(21) and Eq.(22), we see that the two approaches are indeed equivalent. Therefore, we can use either one of them to predict the label of novel query point. We decide the class that gives maximum class posterior according to Eq.(21) or Eq.(22) to be the

label of the query point.

3.4 Empirical evaluation

3.4.1 Datasets

We evaluated our model using three synthetic and two real-world datasets. The synthetic datasets are comprised of two well-separated mixture of Gaussians and an overlapped mixture of Gaussians. We used class separation (Dasgupta [1999]) defined as $c = \min_{i \neq j} \left\| \boldsymbol{\mu}_i - \boldsymbol{\mu}_j \right\| / \sqrt{M \max(\lambda_{max}(\boldsymbol{\Sigma}_i), \lambda_{max}(\boldsymbol{\Sigma}_j))}$, where $\lambda_{max}(\boldsymbol{\Sigma})$ represents the largest eigenvalue of the covariance $\boldsymbol{\Sigma}$ and M is the dimensionality of the data, to quantify the difficulty of the datasets. A $\frac{1}{2}$-separated mixture corresponds to highly overlapping Gaussians, while a $1\frac{1}{2}$-separated (or larger) is considered to be a well-separated mixture. For real world data we use *Iris* and *Wine* data from the UCI repository (Frank and Asuncion [2010]). *Iris* dataset introduced by Fisher [1936], is 4-dimensional data of 3 classes. The goal is to predict of which class the flower belongs to. The UCI *Wine* data is also divided into 3 classes but has higher dimensionality of 13. The objective is to predict from which region the wine comes from. The details of each dataset are summarised in Table 3.2.

3.4.2 Results and discussion

The experiments are designed to answer the following research questions: 1) How does label flipping affect the parameter estimates, and the class prediction performance of

Dataset	C-Separation	# Classes	Dimensionality	# Samples
Synth-1	1.5	3	2	300
Synth-2	1.0	4	6	800
Synth-3	1.5	5	10	800
Iris	0.3	3	4	150
Wine	0.35	3	13	178

Table 3.2: Characteristics of the datasets employed.

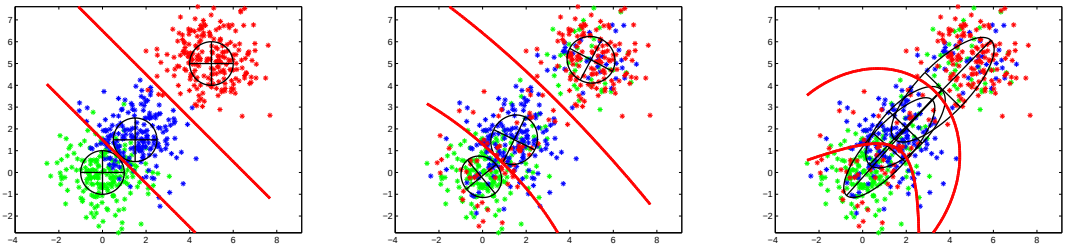
the traditional NDA? 2) Can rNDA improve performance in terms of either or both of these measures? 3) How does our rNDA compare with the existing model-free method of depuration?

We start by an illustrative example where the effect that label flipping has on parameter estimates is most apparent. We injected 30% symmetric noise[1] into *Synth-1* dataset. Figure 3.2 shows the dataset with its true mean and covariance parameters and the induced true decision boundary, in comparison with their estimated counterparts as obtained by our rNDA and the traditional NDA respectively. From this result it is quite clear that incorporating a noise model improves dramatically on the quality of parameter estimates. Without a model of the label noise process, in turn, the estimated covariances of NDA grow towards the noisy distribution. This can also affect the decision boundaries and consequently degrade the classification accuracy.

Next, we present experiments that assess the classification accuracy of the methods under study. We compared the proposed method to the Depuration (Barandela and Gasca [2000]) algorithm which has been reported to be effective against label noise. We also included a nearest neighbour (NN) classifier in our comparison as a baseline since Depuration is structurally related to NN. Figure 3.3 and Figure 3.4 summarise the results obtained. Again we artificially injected symmetric noise from 10%-50% into the datasets. At each level of label-noise we performed 100 experiments, and plotted the mean errors along with the standard deviation.

We observe that on *Synth-2* rNDA outperforms its competitors in up to 50% noise conditions. However in *Synth-3* where dimensionality is higher, rNDA lagged behind Depuration from 40% noise onwards. This is because rNDA has more parameters to estimate and hence requires more training data. We then double the number of samples in *Synth-3*. Figure 3.3(c) indeed shows that when there is large amount of data available

[1] Asymmetric noise will be discussed in Chapter 4 where we compare rNDA with robust discriminative classifiers

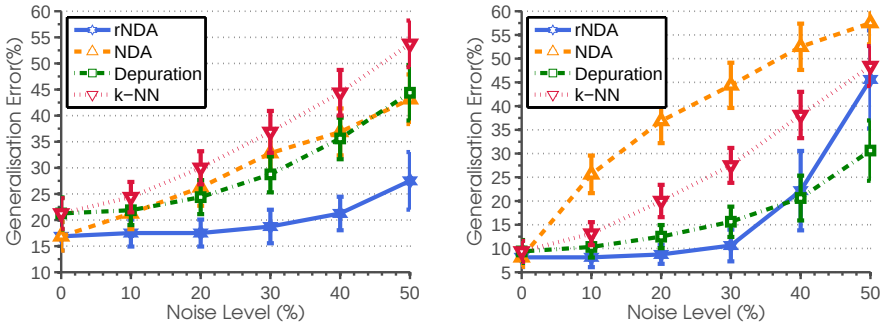


(a) True parameters (b) Estimated with rNDA (c) Estimated with NDA

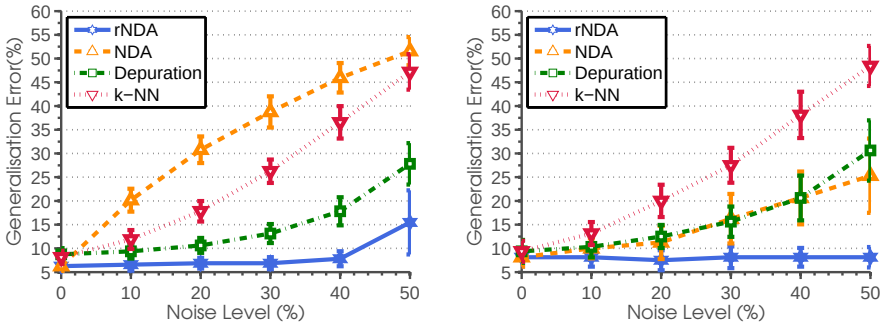
Figure 3.2: Decision boundary induced by the models at 30% noise level on *Synth-1* dataset. The black ellipses are the estimated parameters.

rNDA can perform at its best. Note in both datasets that NDA was highly affected by label noise and its classification performance degrades rapidly as contamination rate increases. For some applications getting more data might not be feasible. In such case we can switch to a diagonal covariance matrix where there are less parameters to be estimated. Figure 3.3(d) shows rather impressive results from rNDA with diagonal covariance. We observed good performance from rNDA. Using diagonal covariance also alleviates the negative effect of label noise on NDA to some extent. Still it was not as good as rNDA.

Figure 3.4 shows the results on the real datasets where the Gaussian shape of the classes, as assumed by the model, is more unlikely to hold. Yet, rNDA still ranks in the first place. On the figure, Depuration does occasionally outperform rNDA on the *Wine* dataset but these differences are marginal. We also observed that in some cases NDA performed slightly better than rNDA at 0% noise. This is expected though, because in this case NDA's assumption is correct (no label flipping), while rNDA assumes nothing about the correctness of data labels and needs to estimate more parameters, namely γ_{jk}. Depending on many factors such as the number of training points, dimensionality of the data or search algorithm used, finding the global optimum is not guaranteed. Thus, the results from rNDA could be worse than those obtained from NDA. The Depuration algorithm is also very capable. It may give better results if data distributions were not strictly Gaussians.



(a) *Synth-2* (Full) (b) *Synth-3* (Full)



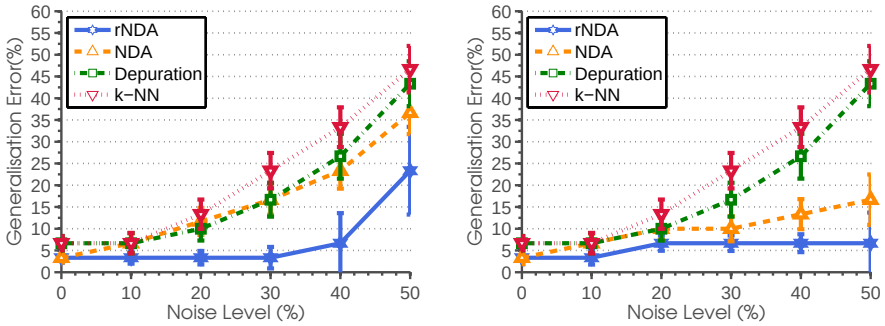
(c) *Synth-3* (Full) + more samples (d) *Synth-3* (Diag)

Figure 3.3: Classification errors (%) on *Synth-2* and *Synth-3* datasets. Full denotes that full covariance matrix is used while diagonal covariance matrix is used in Diag. The number of data examples is doubled in Figure 3.3(c)

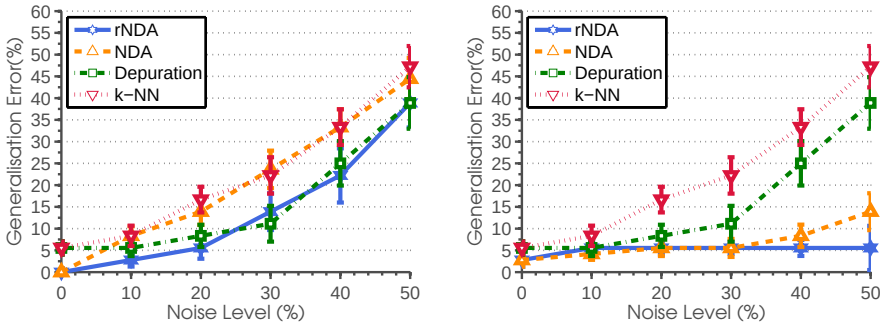
.

Note that, similar to the synthetic data case, we also observe some improvement on the performance of rNDA and NDA when diagonal covariance is employed.

Moreover, we studied various factors of the data setting that have an impact on the results. Towards quantifying these, we conducted experiments to study the effects of the number of classes, the data dimensionality, the number of training points, and the c-separation of the classes. From Figure 3.5(a) the performance of rNDA was nearly unaffected by the number of classes up to 12 classes. Beyond that point Depuration showed lower classification error. Nonetheless, we see that the benefit of the model-based approach becomes significant as the number of classes increases. The effect of data dimensionality is shown in Figure 3.5(b). We notice that rNDA tends to perform better



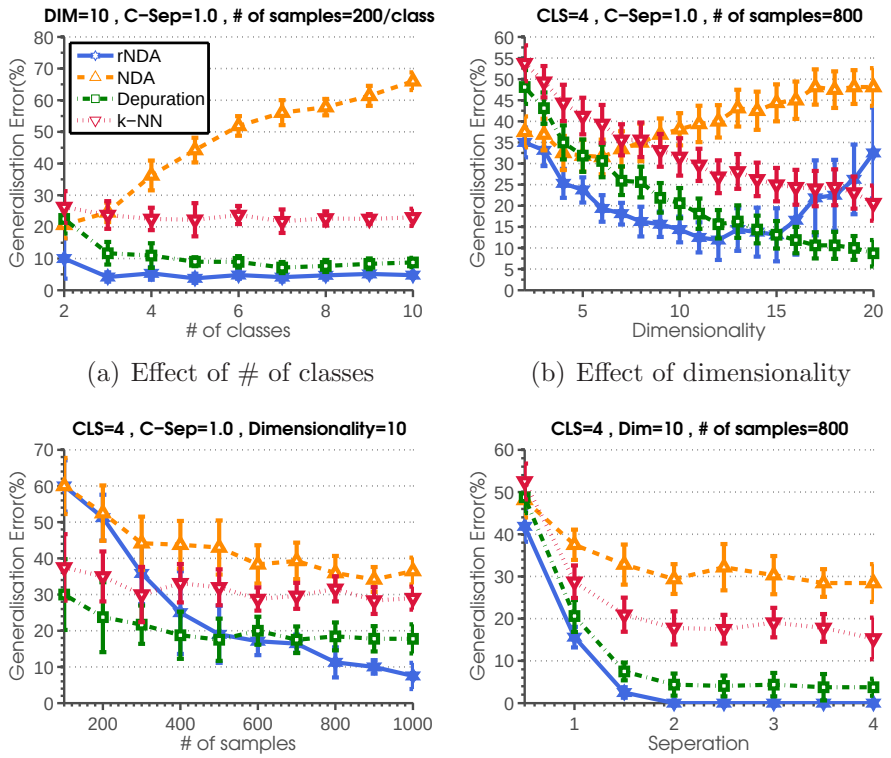
(a) UCI *Iris* (Full) (b) UCI *Iris* (Diag)



(c) UCI *Wine* (Full) (d) UCI *Wine* (Diag)

Figure 3.4: Classification errors (%) on real-world datasets.

in comparison with competitors when the data dimensionality is high. This is because Gaussianity assumption is more likely to hold in high dimensional data. However, in that case more data points are required to obtain the best performance, as seen in Figure 3.5(c). Finally, Figure 3.5(d) shows the effect of varying the c-separation where we observed the performance of all methods increases at roughly the same rate as the classes becomes more clearly separated, as one might expect indeed. These results taken together imply that rNDA will be at its best if the data is well-separated, which is obvious. The result agrees with that from Dasgupta [1999] which stated that a 2-separated mixtures is almost completely separated. That is beyond that value the best accuracy obtained is unlikely to improve.



(a) Effect of # of classes

(b) Effect of dimensionality

(c) Effect of # of sample size

(d) Effect of class separation

Figure 3.5: The effects of number of classes, data dimension, number of training points and class separation on classification accuracy of the robust Normal Discriminant Analysis. The experiments were performed at 30% symmetric noise.

3.5 Theoretical analysis

We now give a theoretical analysis of rNDA. In particular we are interested in the generalisation error of the algorithm, and the quality of the estimated parameters. To facilitate the analysis we shall restrict ourselves to a simpler case where the covariance matrix is shared across all classes. The analysis presented here assumes that the data is drawn from a class of distributions called sub-Gaussian distributions. A sub-Gaussian random variable is a random variable where moment generating function is upper bounded by that of the Gaussian. It covers a random variable drawn from, for example, a standard Gaussian, Bernoulli and bounded distributions.

Definition For a centred sub-Gaussian random variable X

$$\exists b > 0, \quad \forall t \in \mathbb{R}, \quad \mathbb{E}e^{tX} \leq e^{b^2t^2/2} \tag{24}$$

The first part of the analysis is concerned with the generalisation error bound of the rNDA. We follow the standard method in Durrant and Kabán [2010], which was used to analyse the performance of Fisher Discriminant Analysis. Let us assume for now that we work with binary classification and the labels take the values from $\{0, 1\}$.

Theorem 3.5.1 *Let $(\boldsymbol{x}_q, y) \sim \mathcal{D}$ be an example drawn from a sub-Gaussian distribution. Let $\boldsymbol{\mu}_0, \boldsymbol{\mu}_1$ be the mean of class 0 and class 1 and $\hat{\boldsymbol{\mu}}_0, \hat{\boldsymbol{\mu}}_1$ be their estimates, respectively. Let $\boldsymbol{\Sigma}$ be a shared covariance matrix and $\hat{\boldsymbol{\Sigma}}$ be its estimate. Let $\hat{\pi}_0$ be the estimate of π_0, the prior probability of class 0. Denote an instance of two-class rNDA trained from i.i.d. training set S by $\hat{h}$. Then the misclassification error of $\hat{h}$ is bounded above by*

$$p[\hat{h}(\mathbf{x}_q) \neq y] \leq \pi_0 \exp\left(-\frac{1}{8}\frac{\left\{(\hat{\boldsymbol{\mu}}_1 + \hat{\boldsymbol{\mu}}_0 - 2\mu_0)^T\hat{\boldsymbol{\Sigma}}^{-1}(\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_0) - 2\log\frac{1-\hat{\pi}_0}{\hat{\pi}_0}\right\}^2}{(\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_0)^T\hat{\boldsymbol{\Sigma}}^{-1}\boldsymbol{\Sigma}\hat{\boldsymbol{\Sigma}}^{-1}(\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_0)}\right)$$
$$+(1-\pi_0)\exp\left(-\frac{1}{8}\frac{\left\{(\hat{\boldsymbol{\mu}}_0 + \hat{\boldsymbol{\mu}}_1 - 2\mu_1)^T\hat{\boldsymbol{\Sigma}}^{-1}(\hat{\boldsymbol{\mu}}_0 - \hat{\boldsymbol{\mu}}_1) - 2\log\frac{\hat{\pi}_0}{1-\hat{\pi}_0}\right\}^2}{(\hat{\boldsymbol{\mu}}_0 - \hat{\boldsymbol{\mu}}_1)^T\hat{\boldsymbol{\Sigma}}^{-1}\boldsymbol{\Sigma}\hat{\boldsymbol{\Sigma}}^{-1}(\hat{\boldsymbol{\mu}}_0 - \hat{\boldsymbol{\mu}}_1)}\right) \tag{25}$$

Proof We begin with the equation corresponding to the decision boundary of the model. Considering two-class classification, given a query point $\mathbf{x}_q$, the algorithm predicts $\hat{y} = 1$ if $p(y = 1|\mathbf{x}_q) > p(y = 0|\mathbf{x}_q)$ and $\hat{y} = 0$ otherwise. This rule can be written as a function as follows:

$$\hat{h}(\mathbf{x}_q) = \delta\left\{\log\frac{p(y=1|\mathbf{x}_q)}{p(y=0|\mathbf{x}_q)} > 0\right\} \tag{26}$$

where $\delta(\cdot)$ is a function that returns 1 if the assertion is true and 0 otherwise. Now the

generalisation error is the probability that the algorithm gives an incorrect prediction. Without loss of generality, considering the case where $\mathbf{x}_q$ has label $y = 0$, the probability that $\mathbf{x}_q$ is misclassified could be written as,

$$p\Big\{\hat{h}(\mathbf{x}_q) \neq y | y = 0\Big\} = p\Big\{\hat{h}(\mathbf{x}_q) > 0\Big\} \tag{27}$$

where the probability and the expectation are w.r.t. $\mathbf{x}_q$. Plugging in the definition of $\hat{h}$ we obtain,

$$\begin{aligned}
&p\Big\{ \log\frac{p(y=1|\mathbf{x}_q)}{p(y=0|\mathbf{x}_q)} > 0\Big\} \\
&\quad = p\Big\{ \log \frac{p(\mathbf{x}_q|y=1)p(y=1)}{p(\mathbf{x}_q|y=0)p(y=0)} > 0\Big\} \\
&\quad = p\Big\{ \log \frac{1-\hat{\pi}_0}{\hat{\pi}_0} + \frac{1}{2}(\mathbf{x}_q - \hat{\boldsymbol{\mu}}_0)^T\hat{\boldsymbol{\Sigma}}^{-1}(\mathbf{x}_q - \hat{\boldsymbol{\mu}}_0) - \frac{1}{2}(\mathbf{x}_q - \hat{\boldsymbol{\mu}}_1)^T\hat{\boldsymbol{\Sigma}}^{-1}(\mathbf{x}_q - \hat{\boldsymbol{\mu}}_1) > 0\Big\} \\
&\quad = p\Big\{ \log \frac{1-\hat{\pi}_0}{\hat{\pi}_0} + (\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_0)^T\hat{\boldsymbol{\Sigma}}^{-1}(\mathbf{x}_q - \frac{\hat{\boldsymbol{\mu}}_0 + \hat{\boldsymbol{\mu}}_1}{2}) > 0\Big\} \\
&\quad = p\Big\{ c_0 \log \frac{1-\hat{\pi}_0}{\hat{\pi}_0} + c_0(\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_0)^T\hat{\boldsymbol{\Sigma}}^{-1}(\mathbf{x}_q - \frac{\hat{\boldsymbol{\mu}}_0 + \hat{\boldsymbol{\mu}}_1}{2}) > 0\Big\}
\end{aligned}$$

for all $c_0 > 0$. Exponentiating both sides and using Markov's inequality (Appendix B.4),

$$\begin{aligned}
&p\Big\{ \exp\left(c_0 \log \frac{1-\hat{\pi}_0}{\hat{\pi}_0} + c_0(\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_0)^T\hat{\boldsymbol{\Sigma}}^{-1}(\mathbf{x}_q - \frac{\hat{\boldsymbol{\mu}}_0 + \hat{\boldsymbol{\mu}}_1}{2})\right) > 1\Big\} \\
&\leq \mathbb{E}_{\mathbf{x}_q|y=0}\Big\{ \exp\left(c_0 \log \frac{1-\hat{\pi}_0}{\hat{\pi}_0} + c_0(\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_0)^T\hat{\boldsymbol{\Sigma}}^{-1}(\mathbf{x}_q - \frac{\hat{\boldsymbol{\mu}}_0 + \hat{\boldsymbol{\mu}}_1}{2})\right)\Big\} \\
&= \exp\left(c_0 \log \frac{1-\hat{\pi}_0}{\hat{\pi}_0} - \frac{1}{2}c_0(\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_0)^T\hat{\boldsymbol{\Sigma}}^{-1}(\hat{\boldsymbol{\mu}}_0 + \hat{\boldsymbol{\mu}}_1)\right) \\
&\quad \times \mathbb{E}_{\mathbf{x}_q|y=0}\Big\{ \exp\left(c_0(\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_0)^T\hat{\boldsymbol{\Sigma}}^{-1}\mathbf{x}_q\right)\Big\}
\end{aligned} \tag{28}$$

The expectation is in the form of the moment generating function (MGF) of a multivariate Gaussian which is given by $\mathbb{E}\exp(tX) = \exp(t^T\boldsymbol{\mu} + t^T\boldsymbol{\Sigma}t/2)$. Plugging in the MGF, we

have the probability of misclassification is bounded above by

$$\exp(c_0 \log \frac{1-\hat{\pi}_0}{\hat{\pi}_0} - \frac{1}{2} c_0 (\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_0)^T \hat{\boldsymbol{\Sigma}}^{-1} (\hat{\boldsymbol{\mu}}_0 + \hat{\boldsymbol{\mu}}_1) \cdots$$
$$\cdots + \boldsymbol{\mu}_0^T c_0 \hat{\boldsymbol{\Sigma}}^{-1} (\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_0) + \frac{1}{2} (\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_0)^T c_0^2 \hat{\boldsymbol{\Sigma}}^{-1} \boldsymbol{\Sigma} \hat{\boldsymbol{\Sigma}}^{-1} (\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_0)) \tag{29}$$

where $\boldsymbol{\mu}_0$ and $\boldsymbol{\Sigma}$ is the true mean and true covariance matrix of the class of $\mathbf{x}_q$. We can also tighten the bound by optimising for the best c_0. Because the exponential function is monotonic increasing function, minimising its argument is sufficient for minimising the bound. Taking derivative of the argument with respect to c_0, equating to zero and solving for c_0 we have,

$$c_0 = \frac{(\hat{\boldsymbol{\mu}}_1 + \hat{\boldsymbol{\mu}}_0 - 2\boldsymbol{\mu}_0)^T \hat{\boldsymbol{\Sigma}}^{-1} (\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_0) - 2 \log \frac{1-\hat{\pi}_0}{\hat{\pi}_0}}{2 (\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_0)^T \hat{\boldsymbol{\Sigma}}^{-1} \boldsymbol{\Sigma} \hat{\boldsymbol{\Sigma}}^{-1} (\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_0)} \tag{30}$$

Substituting this c_0 back to the bound yields the misclassification probability when $\mathbf{x}_q$ is from class 0 as,

$$\exp\left(-\frac{1}{8} \frac{[(\hat{\boldsymbol{\mu}}_1 + \hat{\boldsymbol{\mu}}_0 - 2\boldsymbol{\mu}_0)^T \hat{\boldsymbol{\Sigma}}^{-1} (\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_0) - 2 \log \frac{1-\hat{\pi}_0}{\hat{\pi}_0}]^2}{(\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_0)^T \hat{\boldsymbol{\Sigma}}^{-1} \boldsymbol{\Sigma} \hat{\boldsymbol{\Sigma}}^{-1} (\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_0)} \right) \tag{31}$$

The case when $\mathbf{x}_q$ is from class 1 can be derived similarly and gives

$$\exp\left(-\frac{1}{8} \frac{[(\hat{\boldsymbol{\mu}}_0 + \hat{\boldsymbol{\mu}}_1 - 2\boldsymbol{\mu}_1)^T \hat{\boldsymbol{\Sigma}}^{-1} (\hat{\boldsymbol{\mu}}_0 - \hat{\boldsymbol{\mu}}_1) - 2 \log \frac{\hat{\pi}_0}{1-\hat{\pi}_0}]^2}{(\hat{\boldsymbol{\mu}}_0 - \hat{\boldsymbol{\mu}}_1)^T \hat{\boldsymbol{\Sigma}}^{-1} \boldsymbol{\Sigma} \hat{\boldsymbol{\Sigma}}^{-1} (\hat{\boldsymbol{\mu}}_0 - \hat{\boldsymbol{\mu}}_1)} \right) \tag{32}$$

Using the law of total probability, we combine the two terms in Eqs.(31)-(32)and arrive at Theorem 3.5.1, and that concludes the proof. ■

Based on the technique to bound the performance of a two-class rNDA, we can also derive the bound for a multi-class rNDA with shared covariances.

Theorem 3.5.2 *Let $(\boldsymbol{x}_q, y) \sim \mathcal{D}$ be an example drawn from a sub-Gaussian distribution.*

Let $\boldsymbol{\mu}_j$ be the mean of class j and $\hat{\boldsymbol{\mu}}_j$ be its estimates. Let $\boldsymbol{\Sigma}$ be a shared covariance matrix, and $\hat{\boldsymbol{\Sigma}}$ be its estimate. Let $\hat{\pi}_j$ be the estimate of π_j, the prior probability of class j. Denote an instance of a multi-class rNDA trained from i.i.d. training set S by $\hat{h}$. Then the misclassification error of $\hat{h}$ is bounded above by

$$p\Big\{\hat{h}(\mathbf{x}_q) \neq y\Big\} \leq \sum_{j=1}^{K} \sum_{k=1,k\neq j}^{K} \pi_j \exp\left(-\frac{1}{8}\frac{\Big\{(\hat{\boldsymbol{\mu}}_k + \hat{\boldsymbol{\mu}}_j - 2\mu_j)^T \hat{\boldsymbol{\Sigma}}^{-1}(\hat{\boldsymbol{\mu}}_k - \hat{\boldsymbol{\mu}}_j) - 2\log\frac{\hat{\pi}_k}{\hat{\pi}_j}\Big\}^2}{(\hat{\boldsymbol{\mu}}_k - \hat{\boldsymbol{\mu}}_j)^T \hat{\boldsymbol{\Sigma}}^{-1}\boldsymbol{\Sigma}\hat{\boldsymbol{\Sigma}}^{-1}(\hat{\boldsymbol{\mu}}_k - \hat{\boldsymbol{\mu}}_j)}\right) \tag{33}$$

Proof For the bound on multi-class problems, we use one-against-all approach where the decision rule for class j is given by all binary decision rules against the other classes.

$$\hat{h}_j(\mathbf{x}_q) = \delta\Big\{\log\frac{p(y=j|\mathbf{x}_q)}{p(y=k|\mathbf{x}_q)} > 0\Big\}\,, \quad \forall k \neq j \tag{34}$$

A misclassification occurs when $\hat{h}_j(\cdot) < 0$. Similar to the proof in Theorem 3.5.1 the generalisation error bound for the case when $\mathbf{x}_q$ is from class j is given by:

$$p\Big\{\hat{h}(\mathbf{x}_q) \neq y\Big\} \leq \sum_{k=1,k\neq j}^{K} \pi_j \exp\left(-\frac{1}{8}\frac{\Big\{(\hat{\boldsymbol{\mu}}_k + \hat{\boldsymbol{\mu}}_j - 2\boldsymbol{\mu}_j)^T \hat{\boldsymbol{\Sigma}}^{-1}(\hat{\boldsymbol{\mu}}_k - \hat{\boldsymbol{\mu}}_j) - 2\log\frac{\hat{\pi}_k}{\hat{\pi}_j}\Big\}^2}{(\hat{\boldsymbol{\mu}}_k - \hat{\boldsymbol{\mu}}_j)^T \hat{\boldsymbol{\Sigma}}^{-1}\boldsymbol{\Sigma}\hat{\boldsymbol{\Sigma}}^{-1}(\hat{\boldsymbol{\mu}}_k - \hat{\boldsymbol{\mu}}_j)}\right) \tag{35}$$

Combining all the bounds of the other $K-1$ cases, i.e., $y = i, i \neq j$ using the union bound (Appendix B.3) concludes the proof. ■

Theorem 3.5.2 bounds the error of rNDA conditioned on a fixed training set. However, it is not obvious from the resulting bound why the rNDA is more robust to the label noise than NDA. To get more insight on the relative performance between the traditional NDA and the new rNDA, it is useful to look at the accuracy of the parameter estimates. We have seen that the estimate of the class prior in rNDA, Eq.(17) is different from that found in the traditional NDA and as a consequence the estimated mean and estimated

covariance are changed during the optimisation process too. We will show that *in the best case scenario* parameter estimates via rNDA model are more accurate.

Theorem 3.5.3 *Let $\hat{p}_{nda}(y)$ be the estimate of the true class prior by NDA, and $\hat{p}_{rnda}(y)$ be the estimate of the true class prior by the robust model. Let $p(y)$ be the true class prior. Assuming that the class posteriors reflect the true class memberships perfectly, then*

$$\left|\hat{p}_{nda}(y) - p(y)\right| \geq \left|\hat{p}_{rnda}(y) - p(y)\right| \tag{36}$$

Proof We begin by bounding the l.h.s. of the bound. Let N be the number of samples in the training set. Since NDA assumes that all the observed labels are the true labels, we have $\hat{p}_{nda}(y) = \hat{p}_{nda}(\tilde{y})$. For any class j the expression leads to:

$$\begin{aligned}
\left|\hat{p}_{nda}(\tilde{y}=j) - p(y=j)\right| &= \left|\hat{p}_{nda}(\tilde{y}=j) - p(\tilde{y}=j) + p(\tilde{y}=j) - p(y=j)\right| \\
&= \left|\frac{1}{N}\sum_{n=1}^{N}\mathbb{1}(\tilde{y}_n=j) - p(\tilde{y}=j) + p(\tilde{y}=j) - p(y=j)\right| \\
&\leq \left|\frac{1}{N}\sum_{n=1}^{N}\mathbb{1}(\tilde{y}_n=j) - p(\tilde{y}=j)\right| + \left|p(\tilde{y}=j) - p(y=j)\right| \qquad (37)
\end{aligned}$$

The last step is due to the triangle inequality. We then bound the first term on the r.h.s. using Hoeffding's inequality (Appendix B.1).

$$p\left[\left|\frac{1}{N}\sum_{n=1}^{N}\mathbb{1}(\tilde{y}_n=j) - p(\tilde{y}=j)\right| > \epsilon\right] \leq 2\exp(-2N\epsilon^2) \tag{38}$$

Plugging Eq.(38) back into Eq.(37) and taking limit as the number of training samples goes to infinity, we obtain the following

$$\lim_{N\to\infty}\left|\hat{p}_{nda}(\tilde{y}=j) - p(y=j)\right| \leq \lim_{N\to\infty}\left|2\exp(-2N\epsilon^2)\right| + \lim_{N\to\infty}\left|p(\tilde{y}=j) - p(y=j)\right| \tag{39}$$

It it clear that when N goes to infinity the first term of the r.h.s. in Eq.(39) will go to zero. The second term will never be zero unless 1) there is no label flipping and 2) the flipping is uniform. In such case $p(\tilde{y})$ equals $p(y)$. This shows that a traditional NDA will get confused with label noise and yields inaccurate prior estimates.

The second part of the proof is concerned with showing that the prior estimate from rNDA is more accurate. First, consider the update step for the prior in Eq.(17). Assuming that the posterior estimates reflect the true class membership, the update step can be interpreted as dividing the number of points that have the true label j but the observed label was k by the total number of point which is believed to have true label j. Under the preconditions of the theorem, we could write the r.h.s. of Eq.(36) as

$$\begin{aligned}
\left|\hat{p}_{rnda}(y=j) - p(y=j)\right| &= \left|\frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{K}\frac{\mathbb{1}(y_n=j,\tilde{y}_n=k)\mathbb{1}(\tilde{y}=k)}{\mathbb{1}(\tilde{y}=k)} - p(y=j)\right| \\
&= \left|\frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{K}\mathbb{1}(y_n=j,\tilde{y}_n=k) - p(y=j)\right| \\
&= \left|\frac{1}{N}\sum_{n=1}^{N}\mathbb{1}(y_n=j) - p(y=j)\right| \qquad (40)
\end{aligned}$$

Bounding the above equation using Hoeffding's bound we have

$$p\left[\left|\frac{1}{N}\sum_{n=1}^{N}\mathbb{1}(y_n=j) - p(y=j)\right| \leq \epsilon\right] \geq 1 - 2\exp(-2N\epsilon^2) \qquad (41)$$

Now as N approaches infinity $\hat{p}(y)$ converges to its expectation and hence their difference approaches zero. Referring back to the main theorem we see that the prior estimate of rNDA is more accurate and this concludes the proof. ■

The above theorem shows that having a label noise model improves prior estimates. As a consequence the decision boundary is shifted towards the optimal one. We can further bound the probability that the estimated means and estimated covariances are

close to their expectation using a similar technique. We shall start with the means.

Theorem 3.5.4 *Consider a training set S drawn from a sub-Gaussian distribution in $\mathbb{R}^M$ with covariance matrix $\mathbf{\Sigma}$. Let $\hat{\boldsymbol{\mu}}_j$ be the estimated mean $\boldsymbol{\mu}_j$ be the true mean of class j. Assume further that all classes share the same covariance matrix and denote its diagonal element by σ_j^2. Then,*

$$p\left[\|\hat{\boldsymbol{\mu}}_j - \boldsymbol{\mu}_j\|_1 < \epsilon\right] \geq 1 - 2M\exp(-\frac{n_j\epsilon^2}{2M^2\sigma_j^2})$$

Proof The proof begins by bounding each component $\hat{\mu}_{ij}$ of $\hat{\boldsymbol{\mu}}_j$ individually. Now we would like to say that the probability

$$p\{|\hat{\mu}_{ij} - \mu_{ij}| < \epsilon_0\} \tag{42}$$

is high. We shall start by bounding the negation of the above for one of the two cases of the absolute value function. We write:

$$\begin{aligned} p\{\hat{\mu}_{ij} - \mu_{ij} > \epsilon_0\} &= p\{\exp\left(c_0(\hat{\mu}_{ij} - \mu_{ij})\right) > \exp(c_0\epsilon_0)\} \\ &\leq \exp(-c_0\epsilon_0) \cdot \mathbb{E}_\mu[\exp\left(c_0(\hat{\mu}_{ij} - \mu_{ij})\right)] \\ &\leq \exp(-c_0\epsilon_0) \cdot \exp(\frac{1}{2}c_0^2\frac{\sigma_j^2}{n_j}) \\ &= \exp(-c_0\epsilon_0 + \frac{1}{2}c_0^2\frac{\sigma_j^2}{n_j}) \end{aligned} \tag{43}$$

The first step comes from the use of Laplace transformation. Then Markov's inequality (Appendix B.4) is used to upper bound the expression. Finally we bound the expectation term with the moment generating function of the Gaussian. The Gaussianity comes from the fact that, if the data classes were Gaussian, then the mean estimate is distributed according to a Gaussian distribution with mean μ_{ij} and covariance σ_j^2/n_j. That is $\hat{\mu}_{ij} -$

$\mu_{ij} \sim N(0, \sigma_j^2/n_j)$. When the data classes are sub-Gaussian, the mean estimates will be sub-Gaussian too, i.e., linear combination of sub-Gaussians is a sub-Gaussian (Vershynin [2010]). To tighten the bound we can optimise for the best c_0. Since the exponential function is monotonically increasing, optimising the function is equivalent to optimising its argument. Taking the derivative of the argument in the exponential w.r.t. c_0, we have

$$c_0 = \frac{\epsilon_0 n_j}{\sigma_j^2} \tag{44}$$

Substituting the optimal c_0 back into the bound we get,

$$p\{\hat{\mu}_{ij} - \mu_{ij} > \epsilon_0\} \leq \exp(-\frac{\epsilon_0^2 n_j}{2\sigma_j^2}) \tag{45}$$

Bounding the other case of the absolute value function, we have

$$p\{\hat{\mu}_{ij} - \mu_{ij} < -\epsilon_0\} \leq \exp(-\frac{\epsilon_0^2 n_j}{2\sigma_j^2}) \tag{46}$$

Now, combining the two bounds from Eq.(45) and Eq.(46), by the union bound we have,

$$p\{|\hat{\mu}_{ij} - \mu_{ij}| > \epsilon_0\} \leq 2\exp(-\frac{\epsilon_0^2 n_j}{2\sigma_j^2}) \tag{47}$$

Using the union bound to take into account all of the components of the mean we finally arrive at,

$$p\{\|\hat{\boldsymbol{\mu}}_j - \boldsymbol{\mu}_j\|_1 > M\epsilon_0\} \leq 2M\exp(-\frac{\epsilon_0^2 n_j}{2\sigma_j^2}) \tag{48}$$

Or equivalently,

$$p\{\|\hat{\boldsymbol{\mu}}_j - \boldsymbol{\mu}_j\|_1 < M\epsilon_0\} \leq 1 - 2M\exp(-\frac{\epsilon_0^2 n_j}{2\sigma_j^2}) \tag{49}$$

We conclude the proof by substituting $\epsilon = M\epsilon_0$ into the resulting bound. ■

Next we will try to bound the probability that the covariance estimation will be accurate. We borrow the technique from random matrix analysis. The collection of the data points $\mathbf{x}_n, n = 1 : N$ forms an $N \times M$ random matrix, where M is the dimensionality of data and each row represents an independent random sample point. The empirical covariance estimate is in the form

$$\hat{\mathbf{\Sigma}} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \mu)(\mathbf{x}_n - \mu)^T \tag{50}$$

Now it follows from the law of large numbers (Vershynin [2010]) that

$$\lim_{N\to\infty} \|\hat{\mathbf{\Sigma}} - \mathbf{\Sigma}\| \to 0 \tag{51}$$

where, $\|\cdot\|$ is the operator norm, i.e., largest eigenvalue of the argument. In other words, the empirical covariance estimate converges asymptotically to its expectation (true covariance) if we have infinitely many data points. However in real life we can not hope for that to be true. In fact in many classification tasks the number of data points available is very limited and obtaining more data is financially, practically expensive. For that reason a non-asymptotic analysis is required. Basically, we would like to know how large N should be for accurate estimation such that $\|\hat{\mathbf{\Sigma}} - \mathbf{\Sigma}\| \leq \epsilon\|\mathbf{\Sigma}\|$.

To do so, we make use of the following covariance estimation result for sub-Gaussian distributions from Vershynin [2010]. The result is based on a definition of sub-Gaussian from Vershynin [2010] that is slightly different from the definition we have used earlier.

Definition For a centred sub-Gaussian random variable X

$$\mathbb{E}\exp(tX) \leq \exp\left(Ct^2\|X\|_{\psi_2}^2\right) \quad \forall t \in \mathbb{R} \tag{52}$$

where $C > 0$ are absolute constants, and $\|X\|_{\psi_2} = \sup_{p\geq 1} p^{-1/2}(\mathbb{E}|X|^p)^{1/p}$.

Lemma 3.5.5 (Vershynin [2010]) *Consider a sub-Gaussian distribution in $\mathbb{R}^M$ with covariance matrix $\mathbf{\Sigma}$, and let $\epsilon \in (0,1)$, $t \geq 1$. If $N \geq c(t/\epsilon)^2 M$ then with probability at least $1 - 2\exp(-t^2 M)$ one has*

$$\|\hat{\mathbf{\Sigma}} - \mathbf{\Sigma}\| \leq \epsilon \|\mathbf{\Sigma}\| \tag{53}$$

where the constant c depends only on the sub-Gaussian norm of the distribution, and $\|\cdot\|$ denotes the operator norm.

The above lemma shows that a dataset size of $N = O(M)$ suffices for accurate covariance estimation of sub-Gaussian distributions. The lemma is readily applicable to our analysis. Since we assume that the data points are drawn from a mixture of Gaussians with shared covariance, we have:

Theorem 3.5.6 *Consider a sub-Gaussian distribution in $\mathbb{R}^M$. Let $\mathbf{\Sigma}$ be a shared covariance matrix and $\hat{\mathbf{\Sigma}}$ be its estimate. Let N be the number of points in the dataset. Let $\epsilon \in (0,1)$. Assuming that the class posteriors reflect the true class memberships perfectly, then*

$$p\left[\|\hat{\mathbf{\Sigma}} - \mathbf{\Sigma}\| \leq \epsilon \|\mathbf{\Sigma}\|\right] \geq 1 - 2\exp(-\epsilon^2 N/c) \tag{54}$$

where $\|\cdot\|$ denotes the operator norm.

Proof The proof begins by considering the update equation of the covariance, Eq.(15), for a shared covariance case, which is

$$\hat{\mathbf{\Sigma}} = \frac{\sum_{j=1}^{K}\sum_{n=1}^{N} p(y=j|\mathbf{x}_n, \tilde{y})(\mathbf{x}_n - \boldsymbol{\mu}_j)(\mathbf{x}_n - \boldsymbol{\mu}_j)^T}{\sum_{j=1}^{K}\sum_{n=1}^{N} p(y=j|\mathbf{x}_n, \tilde{y})} \tag{55}$$

Now as we assume that the posterior probability reflects class membership perfectly the above expression can be rewritten as

$$\hat{\mathbf{\Sigma}} = \frac{\sum_{j=1}^{K}\sum_{n=1}^{N} \mathbb{1}(y=j)(\mathbf{x}_n - \boldsymbol{\mu}_j)(\mathbf{x}_n - \boldsymbol{\mu}_j)^T}{\sum_{j=1}^{K}\sum_{n=1}^{N} \mathbb{1}(y=j)} \tag{56}$$

Now, note that the denominator is again the total number of points, we have

$$\hat{\boldsymbol{\Sigma}} = \frac{\sum_{j=1}^{K}\sum_{n=1}^{N}\mathbb{1}(y=j)(\mathbf{x}_n-\boldsymbol{\mu}_j)(\mathbf{x}_n-\boldsymbol{\mu}_j)^T}{N} \tag{57}$$

which is the empirical covariance estimate. We apply Lemma 3.5.5 and conclude the proof. ■

3.6 Summary

We presented a generative multi-class classifier for learning with labelling errors. We built this as an extension of quadratic normal discriminant analysis by including a model of the labelling error process. Empirical results on both synthetic datasets and real-world datasets demonstrate that the robust model indeed outperforms the traditional generative approach in terms of parameters estimation, and its classification performance is also significantly enhanced. Preliminary theoretical analyses further confirm that the generalisation error of the robust model is bounded and that parameter estimates obtained from the robust model will be more accurate than those obtained from the classical model.

CHAPTER 4

Robust Logistic Regression

In the previous chapter we investigated a way to robustify normal discriminant analysis, which is a generative classifier. In this chapter we shall consider another famous paradigm in classification: the discriminative approach. We incorporate a probabilistic label noise process into logistic regression and multinomial logistic regression and develop an efficient learning algorithm, together with a proof of its convergence. The new algorithm will learn the classifier jointly with estimating the label flipping probabilities. The second part of this chapter presents some theoretical analysis of the new model. By decomposing and rewriting the model's objective we give a new interpretation by which we can understand its ability to counteract the negative effect of mislabelling as a result of an intrinsic re-weighting mechanism. Empirical results demonstrate that the new robust model is effective against labelling errors.

4.1 The model

Consider a set of training data $S = \{(\mathbf{x}_n, \tilde{y}_n)\}_{n=1}^N$, where $\mathbf{x}_n \in \mathbb{R}^M$ and $\tilde{y}_n \in \{0, 1\}$, where $\tilde{y}_n$ denotes the observed (possibly noisy) label of $\mathbf{x}_n$. In contrast to the generative approach, discriminative classifiers do not model the distribution of the data but instead

they model the posterior probability directly using a parametrised function. The most widely used function for modelling the posterior is the logistic 'sigmoid' function in which case the posterior for class 1 is defined as:

$$p(y = 1|\mathbf{x}, \mathbf{w}) = \sigma(\mathbf{w}^T\mathbf{x}) = \frac{1}{1 + e^{(-\mathbf{w}^T\mathbf{x})}} \tag{1}$$

In the classical scenario for binary classification, the log likelihood is defined as:

$$\mathcal{L}(\mathbf{w}) = \sum_{n=1}^{N} \tilde{y}_n \log p(\tilde{y}_n = 1|\mathbf{x}_n, \mathbf{w}) + (1 - \tilde{y}_n) \log p(\tilde{y}_n = 0|\mathbf{x}_n, \mathbf{w}) \tag{2}$$

where $\mathbf{w}$ is the model's parameter representing a vector orthogonal to the decision boundary and it determines the orientation of the separating plane. If all the labels were presumed to be correct, we would have $p(\tilde{y} = 1|\mathbf{x}_n, \mathbf{w}) = \sigma(\mathbf{w}^T\mathbf{x}_n) = \frac{1}{1+e^{(-\mathbf{w}^T\mathbf{x}_n)}}$ and whenever this is above 0.5 we would decide that $\mathbf{x}_n$ belongs to class 1 and otherwise to class 0. However, when label noise is present, making predictions in this way is no longer valid. Instead we will introduce a latent variable y, to represent the true label, and we model $p(\tilde{y}|\mathbf{x}, \mathbf{w})$ as the following:

$$p(\tilde{y}_n = k|\mathbf{x}_n, \mathbf{w}) = \sum_{j=0}^{1} p(\tilde{y}_n = k|y = j)p(y = j|\mathbf{x}_n, \mathbf{w}) \stackrel{def}{=} \tilde{P}_n^k \tag{3}$$

where $k \in \{0, 1\}$. Therefore, instead of Eq.(2), we define the log likelihood of our model to be:

$$\mathcal{L}(\mathbf{w}, \Gamma) = \sum_{n=1}^{N} \tilde{y}_n \log \tilde{P}_n^1 + (1 - \tilde{y}_n) \log \tilde{P}_n^0 \tag{4}$$

and this is our objective function that needs to be optimised. The graphical representation of this model is given in Figure 4.1. In Eq.(3), $p(\tilde{y} = k|y = j) \stackrel{def}{=} \gamma_{jk}$ represents the probability that the label has flipped from the true label j into the observed label k. These parameters form a label transition table that we will refer to as the gamma table, Γ. For

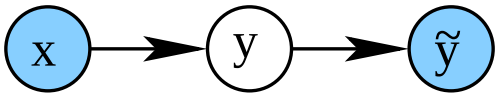


Figure 4.1: Schematic plate diagram of the proposed model. $\mathbf{x}$ and $\tilde{y}$ are observed variables while the true label y is a hidden variable.

binary classification problem the gamma table is summarised in Table 4.1. To classify a novel data point $\mathbf{x}_q$, we predict that $\hat{y}_q = 1$ whenever $p(y = 1|\mathbf{x}_q, \mathbf{w}) = \sigma(\mathbf{w}^T\mathbf{x}_q) = \frac{1}{1+e^{(-\mathbf{w}^T\mathbf{x}_q)}}$ returns a value greater than 0.5, and $\hat{y}_q = 0$ otherwise. Note that the sigmoid is now calculated from $\mathbf{w}$ w.r.t. the true label.

		$\tilde{y}$	
		0	1
y	0	γ_{00}	γ_{01}
	1	γ_{10}	γ_{11}

Table 4.1: Probabilistic relationship between observed label and true label.

Discriminative classifiers are often criticised for lack of explanatory information to accompany their predictions. In our case, however, by the model construction, we are able to directly compute, for an observation $(\mathbf{x}_n, \tilde{y}_n)$, the probability of it being misclassified, as the following:

$$p(\hat{y} \neq \tilde{y}_n|\mathbf{x}_n, \mathbf{w}) = \sum_{j=1, j\neq\tilde{y}_n}^{K} p(y = j|\mathbf{x}_n, \mathbf{w}) \tag{5}$$

This can be thought of as the models "degree of belief" that $\mathbf{x}_n$'s label is incorrect. We may use it either in this form, or in a hard-thresholded form (i.e. predict that the point $\mathbf{x}_n$ is mislabelled if $p(\hat{y} \neq \tilde{y}_n|\mathbf{x}_n, \mathbf{w}) > 0.5$).

4.2 The learning algorithm

Learning the rLR requires us to estimate the weight vector $\mathbf{w}$ as well as the label-flipping probabilities γ_{jk}.

4.2.1 Updating the weight vector

To optimise the weight vector, we can use any non-linear optimiser. Here we decided to employ conjugate gradients (CG) because of its well known computational efficiency, which basically performs the Newton update step along the direction

$$\mathbf{u} = \mathbf{g} - \mathbf{u}^{old}\beta \tag{6}$$

where $\mathbf{g}$ is the gradient of the robust objective w.r.t. the weight vector, and $\mathbf{u}^{old}$ is the direction of the previous step. The parameter β that works best in practice can be obtained from the Hestenes-Stiefel formula,

$$\beta = \frac{\mathbf{g}^T(\mathbf{g} - \mathbf{g}^{old})}{(\mathbf{u}^{old})^T(\mathbf{g} - \mathbf{g}^{old})} \tag{7}$$

To derive $\mathbf{g}_{rlr}$ we take derivative of Eq.(4) w.r.t. $\mathbf{w}$.

$$\begin{aligned}
\mathbf{g}_{rlr} &= \nabla_{\mathbf{w}} \sum_{n=1}^{N} \tilde{y}_n \log \tilde{P}_n^1 + (1 - \tilde{y}_n) \log \tilde{P}_n^0 \\
&= \sum_{n=1}^{N} \frac{\tilde{y}_n \nabla_{\mathbf{w}} \tilde{P}_n^1}{\tilde{P}_n^1} + \frac{(1 - \tilde{y}_n) \nabla_{\mathbf{w}} \tilde{P}_n^0}{\tilde{P}_n^0} \\
&= \sum_{n=1}^{N} \Big(\frac{\tilde{y}_n(\gamma_{11} - \gamma_{01})}{\tilde{P}_n^1} + \frac{(1 - \tilde{y}_n)(\gamma_{10} - \gamma_{00})}{\tilde{P}_n^0} \Big) \sigma(\mathbf{w}^T\mathbf{x}_n)(1 - \sigma(\mathbf{w}^T\mathbf{x}_n))\mathbf{x}_n
\end{aligned} \tag{8}$$

Here $\nabla_{\mathbf{w}} \tilde{P}_n^0$ are found by first expanding the terms in the summation of Eq.(3)

$$\begin{aligned}
\tilde{P}_n^0 &= p(\tilde{y} = 0 | \mathbf{x}_n, \mathbf{w}) \\
&= p(\tilde{y} = 0 | y = 0) p(y = 0 | \mathbf{x}_n, \mathbf{w}) + p(\tilde{y} = 0 | y = 1) p(y = 1 | \mathbf{x}_n, \mathbf{w}) \\
&= \gamma_{00}(1 - \sigma(\mathbf{w}^T\mathbf{x}_n)) + \gamma_{10}\sigma(\mathbf{w}^T\mathbf{x}_n)
\end{aligned} \tag{9}$$

and taking the derivative:

$$\begin{aligned}\nabla_{\mathbf{w}}\tilde{P}_n^0 &= -\gamma_{00}\nabla_{\mathbf{w}}\sigma(\mathbf{w}^T\mathbf{x}_n) + \gamma_{10}\nabla_{\mathbf{w}}\sigma(\mathbf{w}^T\mathbf{x}_n) \\ &= (\gamma_{10} - \gamma_{00})\sigma(\mathbf{w}^T\mathbf{x}_n)(1 - \sigma(\mathbf{w}^T\mathbf{x}_n))\mathbf{x}_n \end{aligned} \tag{10}$$

Similarly, for $\nabla_{\mathbf{w}}\tilde{P}_n^1$:

$$\begin{aligned}\nabla_{\mathbf{w}}\tilde{P}_n^1 &= -\gamma_{01}\nabla_{\mathbf{w}}\sigma(\mathbf{w}^T\mathbf{x}_n) + \gamma_{11}\nabla_{\mathbf{w}}\sigma(\mathbf{w}^T\mathbf{x}_n) \\ &= (\gamma_{11} - \gamma_{01})\sigma(\mathbf{w}^T\mathbf{x}_n)(1 - \sigma(\mathbf{w}^T\mathbf{x}_n))\mathbf{x}_n \end{aligned} \tag{11}$$

Having $\mathbf{g}_{rlr}$, the update equation for $\mathbf{w}$ is then simply the following:

$$\mathbf{w}^{new} = \mathbf{w}^{old} - \eta\mathbf{u}, \tag{12}$$

where η is the learning rate. One may verify that setting γ_{01} and γ_{10} to 0 and γ_{00}, γ_{11} to 1 and after some algebra, the gradient of the robust logistic regression in Eq.(8) will reduce to the well-known gradient expression of the classical logistic regression.

$$\mathbf{g}_{lr} = \sum_{n=1}^{N}(\tilde{y} - \sigma(\mathbf{w}^T\mathbf{x}_n)) \cdot \mathbf{x}_n \tag{13}$$

4.2.2 Updating the label flipping probabilities

Now, what remains is the update method for the label-flipping probabilities. To obtain the updates for the label-flipping probabilities, we introduce Lagrange multipliers to ensure that $\gamma_{00} + \gamma_{01} = 1$ and $\gamma_{10} + \gamma_{11} = 1$. In the following we will derive multiplicative fixed point update equations for these parameters. We first introduce Lagrange multipliers to

ensure that the row of the gamma table sums to 1.

$$\mathcal{L}(\mathbf{w}, \Gamma) = \sum_{n=1}^{N} \tilde{y}_n \log \tilde{P}_n^1 + (1 - \tilde{y}_n) \log \tilde{P}_n^0 + \lambda(1 - \gamma_{00} - \gamma_{01}) \tag{14}$$

To determine the Lagrange multiplier, we take derivative of Eq.(14) w.r.t. γ_{00}, and equate to 0

$$\begin{aligned}
\frac{\partial \mathcal{L}(\mathbf{w}, \Gamma)}{\partial \gamma_{00}} &= \sum_{n=1}^{N} \left(\frac{\tilde{y}_n}{\tilde{P}_n^1} \frac{\partial \tilde{P}_n^1}{\partial \gamma_{00}} + \frac{(1 - \tilde{y}_n)}{\tilde{P}_n^0} \frac{\partial \tilde{P}_n^0}{\partial \gamma_{00}} \right) - \lambda = 0 \\
\lambda &= \sum_{n=1}^{N} \frac{(1 - \tilde{y}_n)}{\tilde{P}_n^0} (1 - \sigma(\mathbf{w}^T \mathbf{x}_n)) \\
\lambda \gamma_{00} &= \gamma_{00} \sum_{n=1}^{N} \frac{(1 - \tilde{y}_n)}{\tilde{P}_n^0} (1 - \sigma(\mathbf{w}^T \mathbf{x}_n))
\end{aligned} \tag{15}$$

Likewise for γ_{01}

$$\begin{aligned}
\frac{\partial \mathcal{L}(\mathbf{w}, \Gamma)}{\partial \gamma_{01}} &= \sum_{n=1}^{N} \left(\frac{\tilde{y}_n}{\tilde{P}_n^1} \frac{\partial \tilde{P}_n^1}{\partial \gamma_{01}} + \frac{(1 - \tilde{y}_n)}{\tilde{P}_n^0} \frac{\partial \tilde{P}_n^0}{\partial \gamma_{01}} \right) - \lambda = 0 \\
\lambda &= \sum_{n=1}^{N} \frac{\tilde{y}_n}{\tilde{P}_n^1} (1 - \sigma(\mathbf{w}^T \mathbf{x}_n)) \\
\lambda \gamma_{01} &= \gamma_{01} \sum_{n=1}^{N} \frac{\tilde{y}_n}{\tilde{P}_n^1} (1 - \sigma(\mathbf{w}^T \mathbf{x}_n))
\end{aligned} \tag{16}$$

Summing (15) and (16) and using the definition $\gamma_{00} + \gamma_{01} = 1$, yields:

$$\begin{aligned}
\lambda \gamma_{00} + \lambda \gamma_{01} &= \gamma_{00} \sum_{n=1}^{N} \left[\frac{(1 - \tilde{y}_n)}{\tilde{P}_n^0} (1 - \sigma(\mathbf{w}^T \mathbf{x}_n)) \right] + \gamma_{01} \sum_{n=1}^{N} \left[\frac{\tilde{y}_n}{\tilde{P}_n^1} (1 - \sigma(\mathbf{w}^T \mathbf{x}_n)) \right] \\
\lambda(\gamma_{00} + \gamma_{01}) &= \gamma_{00} \sum_{n=1}^{N} \left[\frac{(1 - \tilde{y}_n)}{\tilde{P}_n^0} (1 - \sigma(\mathbf{w}^T \mathbf{x}_n)) \right] + \gamma_{01} \sum_{n=1}^{N} \left[\frac{\tilde{y}_n}{\tilde{P}_n^1} (1 - \sigma(\mathbf{w}^T \mathbf{x}_n)) \right] \\
\lambda &= \gamma_{00} \sum_{n=1}^{N} \left[\frac{(1 - \tilde{y}_n)}{\tilde{P}_n^0} (1 - \sigma(\mathbf{w}^T \mathbf{x}_n)) \right] + \gamma_{01} \sum_{n=1}^{N} \left[\frac{\tilde{y}_n}{\tilde{P}_n^1} (1 - \sigma(\mathbf{w}^T \mathbf{x}_n)) \right]
\end{aligned} \tag{17}$$

Substitute (14) back into (15) and (16) and get,

$$\gamma_{00} = \frac{\gamma_{00} \sum_{n=1}^{N} \left[\frac{(1-\tilde{y}_n)}{\tilde{P}_n^0} (1 - \sigma(\mathbf{w}^T \mathbf{x}_n)) \right]}{\gamma_{00} \sum_{n=1}^{N} \left[\frac{(1-\tilde{y}_n)}{\tilde{P}_n^0} (1 - \sigma(\mathbf{w}^T \mathbf{x}_n)) \right] + \gamma_{01} \sum_{n=1}^{N} \left[\frac{\tilde{y}_n}{\tilde{P}_n^1} (1 - \sigma(\mathbf{w}^T \mathbf{x}_n)) \right]} \tag{18}$$

$$\gamma_{01} = \frac{\gamma_{01} \sum_{n=1}^{N} \left[\frac{\tilde{y}_n}{\tilde{P}_n^1} (1 - \sigma(\mathbf{w}^T \mathbf{x}_n)) \right]}{\gamma_{00} \sum_{n=1}^{N} \left[\frac{(1-\tilde{y}_n)}{\tilde{P}_n^0} (1 - \sigma(\mathbf{w}^T \mathbf{x}_n)) \right] + \gamma_{01} \sum_{n=1}^{N} \left[\frac{\tilde{y}_n}{\tilde{P}_n^1} (1 - \sigma(\mathbf{w}^T \mathbf{x}_n)) \right]}. \tag{19}$$

Repeat the above steps for γ_{10} and γ_{11} we have the updates

$$\gamma_{10} = \frac{\gamma_{10} \sum_{n=1}^{N} \left[\frac{(1-\tilde{y}_n)}{\tilde{P}_n^0} \sigma(\mathbf{w}^T \mathbf{x}_n) \right]}{\gamma_{10} \sum_{n=1}^{N} \left[\frac{(1-\tilde{y}_n)}{\tilde{P}_n^0} \sigma(\mathbf{w}^T \mathbf{x}_n) \right] + \gamma_{11} \sum_{n=1}^{N} \left[\frac{\tilde{y}_n}{\tilde{P}_n^1} \sigma(\mathbf{w}^T \mathbf{x}_n) \right]} \tag{20}$$

$$\gamma_{11} = \frac{\gamma_{11} \sum_{n=1}^{N} \left[\frac{\tilde{y}_n}{\tilde{P}_n^1} \sigma(\mathbf{w}^T \mathbf{x}_n) \right]}{\gamma_{10} \sum_{n=1}^{N} \left[\frac{(1-\tilde{y}_n)}{\tilde{P}_n^0} \sigma(\mathbf{w}^T \mathbf{x}_n) \right] + \gamma_{11} \sum_{n=1}^{N} \left[\frac{\tilde{y}_n}{\tilde{P}_n^1} \sigma(\mathbf{w}^T \mathbf{x}_n) \right]}. \tag{21}$$

Algorithm 2 summarises the steps to learn our novel "robust Logistic Regression" (rLR) model.

Algorithm 2 Optimisation of rLR

Input: Γ
  Initialise $\mathbf{w} \leftarrow 0$
  while Iteration $<$ MaxIteration **do**
    Update $\mathbf{w}$ using the gradient given in Eq.(8)
    Update Γ using Eq.(18) and Eq.(20)
  end while
Output: Optimised weight vector, $\mathbf{w}$. Optimised Γ.

4.3 Extension to multi-class problem

It is both useful and straightforward to generalise the two-class rLR of the previous section to multi-class problems. Assuming that $k \in \{1 \dots K\}$, and again introducing the true

class label y as a latent variable, we write the posterior probability of class k:

$$p(\tilde{y}_n = k|\mathbf{x}_n, \mathbf{w}_k) = \sum_{j=1}^{K} p(\tilde{y} = k|y = j)p(y = j|\mathbf{x}_n, \mathbf{w}_j) \stackrel{def}{=} \tilde{P}_n^k \tag{22}$$

The posterior probability of the latent variable is modelled using a softmax function (Böhning [1992]),

$$p(y = k|\mathbf{x}_n, \mathbf{w}_k) = \frac{e^{(\mathbf{w}_k^T \mathbf{x}_n)}}{\sum_{l=1}^{K} e^{(\mathbf{w}_l^T \mathbf{x}_n)}} \tag{23}$$

where $\mathbf{w}_k$ is the weight vector corresponding to class k. The table of label-flipping probabilities is also extended to K classes as the following:

		$\tilde{y}$					
		1	2	$\cdots$	k	$\cdots$	K
	1	γ_{11}	γ_{12}	$\cdots$	γ_{1k}	$\cdots$	γ_{1K}
	2	γ_{21}	γ_{22}	$\cdots$	γ_{2k}	$\cdots$	γ_{2K}
	$\vdots$				$\vdots$		
y	j	γ_{j1}	γ_{j2}	$\cdots$	γ_{jk}	$\cdots$	γ_{jK}
	$\vdots$				$\vdots$		
	K	γ_{K1}	γ_{K2}	$\cdots$	γ_{Kk}	$\cdots$	γ_{KK}

Table 4.2: Probabilistic relationship between observed label and true label (multi-class).

The maximum likelihood (ML) estimate of $\mathbf{w}_k$ is obtained by maximising the data log-likelihood,

$$\mathcal{L}(\mathbf{w}_{c=1:K}, \Gamma) = \sum_{n=1}^{N} \sum_{k=1}^{K} \mathbb{1}(\tilde{y}_n = k) \log \tilde{P}_n^k \tag{24}$$

where $\mathbb{1}(\cdot)$ is the indicator function that gives the value 1 when its argument is true and the value 0 otherwise. The optimisation is again accomplished using the conjugate gradient method. The gradient of the multi-class objective function Eq.(24) w.r.t. $\mathbf{w}_c$ turns out to be

$$\mathbf{g}_{rmlr}^c = \sum_{n=1}^{N} \sum_{k=1}^{K} \frac{\mathbb{1}(\tilde{y}_n = k)}{\tilde{P}_n^k} \nabla_{\mathbf{w}_c} \tilde{P}_n^k$$

$$= \sum_{n=1}^{N}\sum_{k=1}^{K} \frac{\mathbb{1}(\tilde{y}_n = k)}{\tilde{P}_n^k} \frac{e^{(\mathbf{w}_c^T \mathbf{x}_n)}\mathbf{x}_n \left(\sum_{j=1}^{K}(\gamma_{ck} - \gamma_{jk}) \cdot e^{(\mathbf{w}_j^T \mathbf{x}_n)}\right)}{\left(\sum_{l=1}^{K} e^{(\mathbf{w}_l^T \mathbf{x}_n)}\right)^2} \tag{25}$$

where

$$\begin{aligned}
\nabla_{\mathbf{w}_c} \tilde{P}_n^k &= \nabla_{\mathbf{w}_c} \left[\sum_{j=1}^{K} \gamma_{jk} \frac{e^{(\mathbf{w}_j^T \mathbf{x}_n)}}{\sum_{l=1}^{K} e^{(\mathbf{w}_l^T \mathbf{x}_n)}}\right] \\
&= \nabla_{\mathbf{w}_c} \left[\gamma_{1k} \frac{e^{(\mathbf{w}_1^T \mathbf{x}_n)}}{\sum_{l=1}^{K} e^{(\mathbf{w}_l^T \mathbf{x}_n)}} + \cdot + \gamma_{ck} \frac{e^{(\mathbf{w}_c^T \mathbf{x}_n)}}{\sum_{l=1}^{K} e^{(\mathbf{w}_l^T \mathbf{x}_n)}} + \cdot + \gamma_{Kk} \frac{e^{(\mathbf{w}_K^T \mathbf{x}_n)}}{\sum_{l=1}^{K} e^{(\mathbf{w}_l^T \mathbf{x}_n)}}\right] \\
&= \frac{\gamma_{ck} e^{(\mathbf{w}_c^T \mathbf{x}_n)}\mathbf{x}_n \left[\sum_{l=1}^{K} e^{(\mathbf{w}_l^T \mathbf{x}_n)} - e^{(\mathbf{w}_c^T \mathbf{x}_n)}\right]}{\left(\sum_{l=1}^{K} e^{(\mathbf{w}_l^T \mathbf{x}_n)}\right)^2} - \frac{e^{(\mathbf{w}_c^T \mathbf{x}_n)}\mathbf{x}_n \left(\sum_{j=1, j\neq c}^{K} \gamma_{jk} e^{(\mathbf{w}_j^T \mathbf{x}_n)}\right)}{\left(\sum_{l=1}^{K} e^{(\mathbf{w}_l^T \mathbf{x}_n)}\right)^2} \\
&= \frac{e^{(\mathbf{w}_c^T \mathbf{x}_n)}\mathbf{x}_n \left(\sum_{j=1, j\neq c}^{K} \gamma_{ck} e^{(\mathbf{w}_j^T \mathbf{x}_n)}\right) - e^{(\mathbf{w}_c^T \mathbf{x}_n)}\mathbf{x}_n \left(\sum_{j=1, j\neq c}^{K} \gamma_{jk} e^{(\mathbf{w}_j^T \mathbf{x}_n)}\right)}{\left(\sum_{l=1}^{K} e^{(\mathbf{w}_l^T \mathbf{x}_n)}\right)^2} \\
&= \frac{e^{(\mathbf{w}_c^T \mathbf{x}_n)}\mathbf{x}_n \left(\sum_{j=1, j\neq c}^{K} (\gamma_{ck} - \gamma_{jk}) e^{(\mathbf{w}_j^T \mathbf{x}_n)}\right)}{\left(\sum_{l=1}^{K} e^{(\mathbf{w}_l^T \mathbf{x}_n)}\right)^2} \\
&= \frac{e^{(\mathbf{w}_c^T \mathbf{x}_n)}\mathbf{x}_n \left(\sum_{j=1}^{K} (\gamma_{ck} - \gamma_{jk}) e^{(\mathbf{w}_j^T \mathbf{x}_n)}\right)}{\left(\sum_{l=1}^{K} e^{(\mathbf{w}_l^T \mathbf{x}_n)}\right)^2}
\end{aligned} \tag{26}$$

Further, the estimates of γ_{jk} in the gamma matrix again can be obtained by efficient multiplicative update equations. We first introduce the Lagrangian that enforces the row of the gamma table to sum to unity.

$$\mathcal{L}(\mathbf{w}_{c=1:K}, \Gamma) = \sum_{n=1}^{N}\sum_{k=1}^{K} \mathbb{1}(\tilde{y}_n = k) \log \tilde{P}_n^k + \lambda(1 - \sum_{k=1}^{K} \gamma_{jk}) \tag{27}$$

Now finding the value of the Lagrange multiplier we take the derivative of Eq.(27) and

equate to zero.

$$\begin{aligned}\frac{\partial \mathcal{L}(\mathbf{w})}{\partial \gamma_{jk}} &= \sum_{n=1}^{N}\sum_{c=1}^{K} \mathbb{1}(\tilde{y}_n = c)\frac{\partial \log \tilde{P}_n^c}{\partial \gamma_{jk}} + \frac{\partial \lambda(1-\sum_{k=1}^{K}\gamma_{jk})}{\partial \gamma_{jk}} \\ &= \sum_{n=1}^{N} \frac{\mathbb{1}(\tilde{y}_n = k)}{\tilde{P}_n^k}\frac{e^{(\mathbf{w}_j^T \mathbf{x}_n)}}{\sum_{l=1}^{K} e^{(\mathbf{w}_l^T \mathbf{x}_n)}} - \lambda = 0 \end{aligned} \tag{28}$$

We now multiply both sides by γ_{jk} and use the fact that $\sum_{k=1}^{K}\gamma_{jk} = 1$,

$$\begin{aligned}\gamma_{jk}\lambda &= \gamma_{jk}\sum_{n=1}^{N} \frac{\mathbb{1}(\tilde{y}_n = k)}{\tilde{P}_n^k}\frac{e^{(\mathbf{w}_j^T \mathbf{x}_n)}}{\sum_{l=1}^{K} e^{(\mathbf{w}_l^T \mathbf{x}_n)}} \\ \sum_{k=1}^{K}\gamma_{jk}\lambda &= \sum_{k=1}^{K}\gamma_{jk}\sum_{n=1}^{N} \frac{\mathbb{1}(\tilde{y}_n = k)}{\tilde{P}_n^k}\frac{e^{(\mathbf{w}_j^T \mathbf{x}_n)}}{\sum_{l=1}^{K} e^{(\mathbf{w}_l^T \mathbf{x}_n)}} \\ \lambda &= \sum_{k=1}^{K}\gamma_{jk}\sum_{n=1}^{N} \frac{\mathbb{1}(\tilde{y}_n = k)}{\tilde{P}_n^k}\frac{e^{(\mathbf{w}_j^T \mathbf{x}_n)}}{\sum_{l=1}^{K} e^{(\mathbf{w}_l^T \mathbf{x}_n)}} \end{aligned} \tag{29}$$

Substitute λ back to find update equation for γ_{jk}:

$$\gamma_{jk} = \frac{1}{C} \times \gamma_{jk}\sum_{n=1}^{N} \frac{\mathbb{1}(\tilde{y}_n = k)}{\tilde{P}_n^k}\frac{e^{(\mathbf{w}_j^T \mathbf{x}_n)}}{\sum_{l=1}^{K} e^{(\mathbf{w}_l^T \mathbf{x}_n)}} \tag{30}$$

where the constant term C equals

$$\sum_{k=1}^{K}\gamma_{jk}\sum_{n=1}^{N} \frac{\mathbb{1}(\tilde{y}_n = k)}{\tilde{P}_n^k}\frac{e^{(\mathbf{w}_j^T \mathbf{x}_n)}}{\sum_{l=1}^{K} e^{(\mathbf{w}_l^T \mathbf{x}_n)}} \tag{31}$$

To classify a new point,$\mathbf{x}_q$, we decide

$$\hat{y}_q = \arg\max_k \frac{e^{(\mathbf{w}_k^T \mathbf{x}_q)}}{\sum_{l=1}^{K} e^{(\mathbf{w}_l^T \mathbf{x}_q)}} \tag{32}$$

Algorithm 3 summarises the steps to learn our novel "robust Multinomial Logistic Regression" (rMLR) model.

Algorithm 3 Optimisation of rMLR

Input: Γ

Initialise $\mathbf{w}_{c=1:K} \leftarrow 0$

while Iteration $<$ MaxIteration **do**

For each class c, update $\mathbf{w}_c$ using the gradient given in Eq.(25)

Update Γ using Eq.(30)

end while

Output: Optimised set of weight vectors, $\mathbf{w}_{c=1:K}$. Optimised Γ.

4.4 Theoretical analysis

4.4.1 Convergence of the algorithm

We shall now prove that the learning algorithms proposed in the previous sections, for both rLR and rmLR, converge. The proof is for the multi-class algorithm which also covers the binary-class algorithm as well. The idea of the proof is to show that the objective function being optimised, Eq.(24) is nondecreasing under any of our parameter updates. Indeed, the maximisation w.r.t. the weight vector $\mathbf{w}$ by the conjugate gradient method (CG) enjoys the known property of CG to provide monotonically improving estimation of the target (Hestenes and Stiefel [1952]), which guarantees that an objective function being maximised is nondecreasing. Now, it remains to prove that our multiplicative updates for γ_{jk} are also guaranteed to be nondecreasing. To do this, we use the notion of an auxiliary function, in a similar spirit to the proofs in Lee and Seung [2001].

Definition $G(h, h')$ is an auxiliary function for $F(h)$ if

$$G(h, h') \leq F(h), \ \textit{and} \ \ G(h, h) = F(h) \tag{33}$$

are satisfied.

The definition is useful because of the following lemma.

Lemma 4.4.1 (Lee and Seung [2001]) *If G is an auxiliary function, then F is non-decreasing under the update*

$$h^{i+1} = \arg\max_{h} G(h, h^i) \tag{34}$$

Proof $F(h^{i+1}) \geq G(h^{i+i}, h^i) \geq G(h^i, h^i) = F(h^i)$

We will show that by defining an appropriate auxiliary function to the objective function Eq.(24), regarded as a function of Γ, the update equations Eq.(30) for γ_{jk} are guaranteed to converge to a local optimum.

Lemma 4.4.2 *Define*

$$G(\Gamma, \tilde{\Gamma}) = \sum_{n=1}^{N}\sum_{k=1}^{K} \mathbb{1}(\tilde{y}_n = k) \sum_{j=1}^{K} \frac{\tilde{\gamma}_{jk} p(y = j|\mathbf{x}_n, \mathbf{w})}{\sum_{l=1}^{K} \tilde{\gamma}_{lk} p(y = l|\mathbf{x}_n, \mathbf{w})} \times \left(\log \gamma_{jk} p(y = j|\mathbf{x}_n, \mathbf{w}) - \log \frac{\tilde{\gamma}_{jk} p(y = j|\mathbf{x}_n, \mathbf{w})}{\sum_{l=1}^{K} \tilde{\gamma}_{lk} p(y = l|\mathbf{x}_n, \mathbf{w})} \right) \tag{35}$$

This is an auxiliary function for

$$\mathcal{L}(\Gamma) = \sum_{n=1}^{N}\sum_{k=1}^{K} \mathbb{1}(\tilde{y}_n = k) \log \sum_{j=1}^{K} \gamma_{jk} p(y = j|\mathbf{x}_n, \mathbf{w}) \tag{36}$$

Proof For $G(\Gamma, \tilde{\Gamma})$ to be an auxiliary function it needs to satisfy the aforementioned conditions. It is straightforward to verify that $G(\Gamma, \Gamma) = \mathcal{L}(\Gamma)$. To show that $G(\Gamma^{i+1}, \Gamma^i) \leq \mathcal{L}(\Gamma^{i+1})$, we observe that:

$$\log \sum_{j=1}^{K} \gamma_{jk} p(y = j|\mathbf{x}_n, \mathbf{w}) \geq \sum_{j=1}^{K} \alpha_{jk} \log \left(\frac{\gamma_{jk} p(y = j|\mathbf{x}_n, \mathbf{w})}{\alpha_{jk}} \right), \tag{37}$$

by Jensen's inequality (Appendix B.2) and due to the concavity of the logarithm. This

inequality is valid for all non-negative α_{jk} that sum to one. Setting

$$\alpha_{jk} = \frac{\tilde{\gamma}_{jk} p(y = j|\mathbf{x}_n, \mathbf{w})}{\sum_{l=1}^{K} \tilde{\gamma}_{lk} p(y = l|\mathbf{x}_n, \mathbf{w})}, \tag{38}$$

we see that our objective function $\mathcal{L}(\Gamma)$ is always greater than or equal to the auxiliary function in Eq.(35).

Lemma 4.4.3 *The multiplicative update rule of the label flipping probability γ_{jk} given in Eq.(30) is guaranteed to converge.*

Proof The maximum of $G(\Gamma, \tilde{\Gamma})$ with respect to γ_{jk} is found by setting the derivative to zero:

$$\frac{dG(\Gamma, \tilde{\Gamma})}{d\gamma_{jk}} = \sum_{n=1}^{N} \mathbb{1}(\tilde{y}_n = k) \frac{\alpha_{jk}}{\gamma_{jk}} - \lambda = 0 \tag{39}$$

Using the fact that $\sum_k \gamma_{jk} = 1$, we obtain the value of the Lagrange multiplier λ. Putting it back into Eq. (39) we arrive at:

$$\gamma_{jk} = \frac{1}{C} \times \tilde{\gamma}_{jk} \sum_{n=1}^{N} \mathbb{1}(\tilde{y}_n = k) \frac{p(y = j|\mathbf{x}_n, \mathbf{w})}{\sum_{l=1}^{K} \tilde{\gamma}_{lk} p(y = l|\mathbf{x}_n, \mathbf{w})} \tag{40}$$

where C equals $\sum_{k=1}^{K} \tilde{\gamma}_{jk} \sum_{n=1}^{N} \mathbb{1}(\tilde{y}_n = k) \frac{p(y=j|\mathbf{x}_n, \mathbf{w})}{\sum_{l=1}^{K} \tilde{\gamma}_{lk} p(y=l|\mathbf{x}_n, \mathbf{w})}$. Writing out posterior probability $p(y = j|\mathbf{x}_n, \mathbf{w})$ as a softmax function and noting that by definition $\sum_{l=1}^{K} \tilde{\gamma}_{lk} p(y = l|\mathbf{x}_n, \mathbf{w})$ equals $\tilde{P}_n^k$, Eq.(40) then takes the same form as the update rule in Eq.(30). Since $G(\Gamma, \tilde{\Gamma})$ is an auxiliary function, it is guaranteed that the value of $\mathcal{L}$ is nondecreasing under this update.

Theorem 4.4.4 *By alternating between the updates of the weight vector* $\mathbf{w}$ *while the matrix* Γ *is held fixed, and the updates of the elements of* Γ *while* $\mathbf{w}$ *is fixed, the objective function of rmLR is nondecreasing and is thus guaranteed to converge.*

Proof The proof follows directly from the fact that optimising $\mathbf{w}$ using CG is monotonically nondecreasing and from Lemma 4.4.3, that optimising Γ is also nondecreasing. Consequently, the objective function being optimised is monotonically increasing under alternating these iterations.

4.4.2 Connection to EM based optimisation

As an alternative to the above gradient-based learning approach, the algorithm developed in Raykar et al. [2010] in the context of multiple sets of noisy labels could also be instantiated for our problem. The method in Raykar et al. [2010] proposes an EM algorithm to learn a similar model where the true label is modelled as a hidden variable. Instead, we had these hidden variable integrated out when optimising the parameters. It is hence interesting to see how these two algorithms compare.

Similar to Raykar et al. [2010], let y_n be the hidden true labels, and denote $P_n := p(y_n = 1|\mathbf{x}, \mathbf{w}, \tilde{y}_n)$ the posteriors of these. To make the link, the expected complete log likelihood (so-called Q-function) in the data space can then be written as:

$$\mathcal{Q}(\Theta) = \sum_{n=1}^{N} P_n \log(\gamma_{11}^{\tilde{y}_n}\gamma_{10}^{1-\tilde{y}_n}\sigma(\mathbf{w}^T\mathbf{x}_n)) + (1-P_n)\log(\gamma_{01}^{\tilde{y}_n}\gamma_{00}^{1-\tilde{y}_n}(1-\sigma(\mathbf{w}^T\mathbf{x}_n))) \tag{41}$$

- The E-step involves updating P_n based on the given data and the current estimate of Θ:

$$P_n = \frac{\gamma_{11}^{\tilde{y}_n}\gamma_{10}^{1-\tilde{y}_n}\sigma(\mathbf{w}^T\mathbf{x}_n)}{\gamma_{11}^{\tilde{y}_n}\gamma_{10}^{1-\tilde{y}_n}\sigma(\mathbf{w}^T\mathbf{x}_n) + \gamma_{01}^{\tilde{y}_n}\gamma_{00}^{1-\tilde{y}_n}(1-\sigma(\mathbf{w}^T\mathbf{x}_n))} \tag{42}$$

- The M-step then re-estimates the parameters using P_n from the E-step. For example the gradient for updating the weight vector is given by:

$$\mathbf{g} = \sum_{n=1}^{N} \big(P_n - \sigma(\mathbf{w}^T\mathbf{x}_n)\big)\mathbf{x}_n \tag{43}$$

Likewise, γ_{11} is updated using:

$$\gamma_{11} = \frac{\sum_{n=1}^{N} P_n \tilde{y}_n}{\sum_{n=1}^{N} P_n} \tag{44}$$

Now observe that substituting Eq.(42) into Eq.(44), we recover our multiplicative form of updates for γ_{11} — with one subtle but important difference: In the EM approach, P_n used in Eq.(44) is computed with the old values of the parameters. Instead, our multiplicative updates use the latest fresh values of all the parameters they depend on. This not only implies that our algorithm saves the storage cost of the posteriors P_n during the iterations, but it also has a better chance to converge in fewer iterations. The latter can be seen by noting that our algorithm is equivalent to a component-wise EM (Celeux et al. [2001]), that is an EM in which each component in the parameter space $\Theta = \{\mathbf{w}, \Gamma\}$ is updated sequentially. Component-wise EM has indeed been observed empirically to converge faster than standard EM (Celeux et al. [2001]). We should of course note also that P_n can be useful for interpretation and our algorithm does not compute this explicitly during its iterations. However we can compute P_n after convergence using the final values of the parameters.

4.4.3 Interpretation of rLR

We have seen from the last section that rLR makes use of a latent variable y to model the true label. However, it is not obvious why the introduction of this extra variable will lead to a more robust model. In this section we give an intuitive explanation for why this is indeed the case. To facilitate the analysis we shall consider the gradient descent update rule, in which the update for the j-th iteration is defined as:

$$\mathbf{w}^j = \mathbf{w}^{j-1} - \eta \times \mathbf{g} \tag{45}$$

where η is usually referred to as 'learning rate' and $\mathbf{g} = \nabla_{\mathbf{w}}\mathcal{L}$ is the gradient of the data log-likelihood function.

Assume a fixed training set S of size N, and assume also that the gamma matrix is known. Let N_1 be the number of points that have been assigned the label $\tilde{y} = 1$ and N_0 the number of points with $\tilde{y} = 0$. Denote by $\mathbf{g}^+$ the terms of the gradient that correspond to points with $\tilde{y} = 1$, and $\mathbf{g}^-$ those of points with $\tilde{y} = 0$, so that $\mathbf{g} = \mathbf{g}^+ + \mathbf{g}^-$. For classical logistic regression without label noise modelling, these terms are:

$$\mathbf{g}^+_{LR} = \frac{\partial \sum_{n=1}^{N} \tilde{y}_n \log(\frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}_n}})}{\partial \mathbf{w}} = \sum_{n=1}^{N_1} p(y=0|\mathbf{x}_n, \mathbf{w})\mathbf{x} \tag{46}$$

$$\mathbf{g}^-_{LR} = \frac{\partial \sum_{n=1}^{N} (1-\tilde{y}_n) \log(\frac{e^{-\mathbf{w}^T\mathbf{x}_n}}{1+e^{-\mathbf{w}^T\mathbf{x}_n}})}{\partial \mathbf{w}} = \sum_{n=1}^{N_0} -p(y=1|\mathbf{x}_n, \mathbf{w})\mathbf{x}_n \tag{47}$$

Let us now look at the corresponding two terms of the gradient for our robust logistic regression. From the definition of data likelihood we get:

$$\mathbf{g}^+_{rLR} = \sum_{n=1}^{N_1} \frac{\tilde{y}_n(\gamma_{11} - \gamma_{01})}{\tilde{P}^1_n} \sigma(\mathbf{w}^T\mathbf{x}_n)(1 - \sigma(w^T\mathbf{x}_n))\mathbf{x}_n \tag{48}$$

Now, defining $\alpha_n = \frac{(\gamma_{11}-\gamma_{01})p(y=1|\mathbf{x}_n,\mathbf{w})}{p(\tilde{y}_n=1|\mathbf{x}_n,\mathbf{w})}$, we can rewrite this as:

$$\mathbf{g}^+_{rLR} = \sum_{n=1}^{N} \alpha_n p(y=0|\mathbf{x}_n, \mathbf{w})\mathbf{x}_n \tag{49}$$

Likewise, defining $\beta_n = \frac{(\gamma_{00}-\gamma_{10})p(y=0|\mathbf{x}_n,\mathbf{w})}{p(\tilde{y}_n=0|\mathbf{x}_n,\mathbf{w})}$, we find:

$$\mathbf{g}^-_{rLR} = \sum_{n=1}^{N_0} -\beta_n p(y=1|\mathbf{x}_n, \mathbf{w})\mathbf{x}_n \tag{50}$$

We see that α_n and β_n act as weighting parameters. This weighting mechanism adjusts the contribution that particular data points make to the gradient. The weight itself, for

example α_n is composed of two components, the first is a global weight: $(\gamma_{11} - \gamma_{01})$, the second is a data point specific weight: $p(y = 1|\mathbf{w}_n, \mathbf{w})/\tilde{P}_n^1$. That is, each data point within a class will be weighted equally by the global weight, in accordance with the extent of label noise in that class. In addition, the individual points that have been potentially mislabelled are multiplied by the data-point-specific weight factor. Thus, $\mathbf{g}_{rLR}^+$ will take the incorrect information into account to a lesser extent. Similar reasoning applies to $\mathbf{g}_{rLR}^-$in a symmetric manner. Through this weighting mechanism, rLR is then able to distinguish between correctly labelled points and mislabelled points. As a consequence, it is expected that rLR will perform better in terms of generalisation error in label noise conditions, and will be able to detect mislabelled instances with high accuracy.

Finally, let us look at the behaviour of robust logistic regression in a label-noise free scenario, i.e. when the training labels are actually all correct. In this case $\tilde{y} = y$, $\gamma_{01} = 0$ and $\gamma_{11} = 1$. In consequence $\alpha_m = 1$, and likewise $\beta_m = 1$. Hence is this case we recover $\mathbf{g}_{LR}^+ = \mathbf{g}_{rLR}^+$ and $\mathbf{g}_{LR}^- = \mathbf{g}_{rLR}^-$ as in classical logistic regression.

4.4.4 Error analysis

In this section a bound on generalisation error in predicting given labels $\tilde{y}$ is derived. There is no framework to bound the expected loss of predicting the true labels, because the true labels are latent even in the training set. However, using our model architecture, we can analyse performance in terms of the expected loss of predicting the observed labels through the model. Such analysis is informative for two reasons. Firstly, it is a generalisation error bound in predicting y when there is no mislabelling, i.e. $\tilde{y} = y$. Secondly, it quantifies how well the model that includes the latent variable for the true labels explains the observed data. We obtain the bound by means of Rademacher complexity. Let $l(h, \mathbf{x}, y)$ denote the loss of a classifier (or hypothesis) $h \in \mathcal{H}$ on the input point $\mathbf{x}$, where $\mathcal{H}$ is the hypothesis class considered. Define $L_D(h) = \mathbb{E}_{(\mathbf{x},y)\sim D} l(h, \mathbf{x}, y)$ to be the generalisation

error, and $L_S(h) = (1/N) \cdot \sum_{n=1}^{N} l(h, \mathbf{x}_n, y_n)$ to be its empirical estimate. The Rademacher complexity of the composite function of the hypothesis class $\mathcal{H}$ and a training set S is defined as:

$$R(\mathcal{H} \circ S) = \frac{1}{N} \mathop{\mathbf{E}}_{\sigma \sim \{\pm 1\}^N} \left[\sup_{h \in \mathcal{H}} \sum_{n=1}^{N} \sigma_n h(\mathbf{x_n}) \right] \tag{51}$$

Obtaining the Rademacher complexity of a classifier is useful because the following lemma gives an upper bound on the generalisation error of an Empirical Risk Minimisation (ERM) classifier in terms of its Rademacher complexity.

Lemma 4.4.5 (Bartlett and Mendelson [2003]) *For a training set S of size N, let h_s be an ERM hypothesis, $h_s \in \operatorname{argmin}_{h \in \mathcal{H}} L_S(h)$. Assume that for all $\mathbf{x} \in S$ and $h \in \mathcal{H}$ we have the ρ-Lipschitz loss function l satisfying $|l(h, \mathbf{x}, y)| \leq c$ for some positive constant c. Then, with probability at least $1 - \delta$*

$$L_D(h_S) - L_S(h_S) \leq 2\rho R(\mathcal{H} \circ S) + c\sqrt{\frac{\log(1/\delta)}{2N}} \tag{52}$$

To apply this lemma, we first define the loss function $l(h, \mathbf{x}, y)$ associated with rLR that we will show to be Lipschitz [1]. We then calculate the Rademacher complexity of the hypothesis class of rLR. Finally the generalisation error bound is obtained by linking the Rademacher complexity using Lemma 4.4.5.

Let us begin with the loss function associated with robust logistic regression. Recall the data log-likelihood in Eq.(4). Because of the monotonicity of the logarithm, maximising the log-likelihood is equivalent to minimising the negative log-likelihood. It then follows that we can define the loss function to be the negative of the data log-likelihood:

$$l(h, \mathbf{x}_n, y_n) = -\tilde{y}_n \log(\tilde{P}_n^1) - (1 - \tilde{y}_n) \log(\tilde{P}_n^0) \tag{53}$$

[1] $|f(x) - f(y)| \leq \rho |x - y|$

We are now ready to consider the Rademacher complexity of rLR. By the above definition, Rademacher complexity is measured with respect to a hypothesis class and a training set. It has been shown that the complexity of the hypothesis class of a linear halfspace classifier is bounded as:

Lemma 4.4.6 (Shalev-Shwartz [2009]) *Let $\mathcal{H} = \{\mathbf{x} \mapsto \langle \mathbf{w}, \mathbf{x} \rangle, \|\mathbf{w}\|_2 \leq \lambda\}$ defines the hypothesis class of linear classifiers. Let $S = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ be a set of vectors in $\mathbb{R}^M$. Then,*

$$R(\mathcal{H} \circ S) \leq \frac{\lambda \max_n \|\mathbf{x}_n\|_2}{\sqrt{N}} \tag{54}$$

Note that rLR can be regarded as a linear classifier because the loss is a function of a linear function of $\mathbf{w}$. Hence the complexity of its hypothesis class is also defined by Lemma 4.4.6. Putting everything together, we state our result as the following:

Theorem 4.4.7 *Let $h_s \in \mathcal{H}_{rLR}$ be an ERM hypothesis from the class of rLR classifiers, and let l be the loss function defined in Eq.(53). Let S be a training set of N examples drawn i.i.d. from an unknown distribution over $\mathbb{R}^M$. We assume that $\|\mathbf{x}_n\|_2 < \infty, \forall n = 1 : N$, and also that γ_{00} and γ_{11} are bounded away from zero. Then, $\forall \delta \in (0, 1)$, with probability at least $1 - \delta$ the following bound holds:*

$$L_D(h_s) \leq L_S(h_s) + 2\frac{\lambda \max_n \|\mathbf{x}_n\|_2}{\sqrt{N}} + c\sqrt{\frac{\log(1/\delta)}{2N}} \tag{55}$$

Proof We first show that the rLR loss function is a Lipschitz function with Lipschitz constant $\rho = 1$. Note that $\tilde{y}$ can either take value 0 or 1 but not both at the same time, so the loss function decouples into two terms. In order to obtain the Lipschitz constant, we shall show that the absolute value of the derivative of each term is bounded by 1. Without loss of generality, let us consider the first term, $\log \tilde{P}_n^1$. Also for convenience, we

define $\mathbf{a} = \mathbf{w}^T\mathbf{x}$. Then,

$$\begin{aligned}
\tilde{P}_n^1 &= \frac{\gamma_{11}}{1+e^{-\mathbf{a}}} + \frac{\gamma_{01}e^{-\mathbf{a}}}{1+e^{-\mathbf{a}}} \\
&= \frac{\gamma_{11}+\gamma_{01}e^{-\mathbf{a}}}{1+e^{-\mathbf{a}}} \\
\frac{\partial \log(\tilde{P}_n^1)}{\partial \mathbf{a}} &= \frac{\partial \log(\gamma_{11}+\gamma_{01}e^{-\mathbf{a}}) - \log(1+e^{-\mathbf{a}})}{\partial \mathbf{a}} \\
&= \left| \frac{-\gamma_{01}e^{-\mathbf{a}}}{\gamma_{11}+\gamma_{01}e^{-\mathbf{a}}} + \frac{e^{-\mathbf{a}}}{1+e^{-\mathbf{a}}} \right| \\
&= \left| \frac{1}{1+e^{\mathbf{a}}} - \frac{1}{1+\frac{\gamma_{11}}{\gamma_{01}}e^{\mathbf{a}}} \right| \\
&= |f(\alpha)| \qquad (56)
\end{aligned}$$

where in the last step we divided through by $e^{-\mathbf{a}}$, and we defined $\alpha := \gamma_{11}/\gamma_{01}$. Since γ_{11} and γ_{01} are probabilities, their values are between 0 and 1. It follows that the domain of α is $[0, \infty)$. Observe that Eq.(56) attains its maximum either 1) at its end points $f(0)$ and at the limit $f(\infty)$ or 2) at a point where $f'(\alpha) = 0$. The first case can be easily checked by plugging the extreme values into Eq.(56), and we see that $f(0) \in [-1, 0]$ and $f(\infty) \in [0, 1]$. The remaining case can be verified by calculating $f'(\alpha), \alpha \in (0, \infty)$ which turns out to be,

$$f'(\alpha) = \frac{e^{\mathbf{a}}}{(1+\alpha e^{\mathbf{a}})^2} \qquad (57)$$

It can be seen that Eq.(57) is non-negative and can only be zero when $\alpha = \infty$, which is the case we have already considered. Hence, we learn that the absolute value of Eq.(56) is bounded by 1. It follows that $\log \tilde{P}_n^1$ is a 1-Lipschitz function. The case of $\log(\tilde{P}_n^0)$ can be shown to be 1-Lipschitz using similar technique. Next we need to show that the loss function is bounded. Without loss of generality, let us consider Eq.(53) where $\tilde{y}_n = 1$. It

can be shown that the term is bounded by some finite number c.

$$\begin{aligned}
\Big| -\log(\tilde{P}_n^1)\Big| &= \Big| -\log(\frac{\gamma_{01}e^{-\mathbf{a}}}{1+e^{-\mathbf{a}}} + \frac{\gamma_{11}}{1+e^{-\mathbf{a}}})\Big| \\
&= \big|\log(1+e^{-\mathbf{a}}) - \log(\gamma_{01}+\gamma_{11}e^{-\mathbf{a}})\big| \\
&\leq \big|\log(1+e^{-\mathbf{a}})\big| + \big|\log(\gamma_{01}+\gamma_{11}e^{-\mathbf{a}})\big| \\
&\leq 1 + ||\mathbf{w}||_2||\mathbf{x}||_2 + \big|\log(\gamma_{01}+\gamma_{11}e^{-\mathbf{a}})\big| \qquad (58)
\end{aligned}$$

where we have used triangle inequality to get the third line. The last line makes use of a bound on the loss function of standard logistic regression which takes the form: $|\log(1+e^{-\mathbf{a}})| \leq 1 + ||\mathbf{w}||_2||\mathbf{x}||_2$. For the second term we will show that its value is finite, that is we show that an argument to logarithm is never be 0 or infinity. Since $\mathbf{a} = \mathbf{w}^T\mathbf{x}$ is bounded because of the assumptions and, γ_{11} is bounded away from zero, it can be shown that 1) the maximum value of $\gamma_{01}+\gamma_{11}e^{-\mathbf{a}}$ is finite, and 2) its minimum is bounded away from 0. As a consequence we have that rLR's loss function is bounded by some constant c. Ultimately, since we have that the loss function of robust logistic regression is 1-Lipschitz, and it is bounded by a finite constant c, we apply Lemma 4.4.5 to conclude the statement of our theorem.

4.5 Empirical evaluation

4.5.1 Experiment setting

We pose the following questions: 1) Do the proposed methods improve upon their traditional counterpart in terms of generalisation error in label noise conditions? 2) If so, under what type of label-noise is the improvement most apparent? 3) Are the proposed methods able to identify the mislabelled instances accurately? 4) Can the gamma matrix be interpreted in terms of the structure of the dataset being studied? To answer these

questions, we conduct a number of numerical experiments.

Firstly, we compare the classification error of LR, rLR, mLR and rmLR together with two other state-of-the-art algorithms on both synthetic and real world datasets. We also compare the discriminative classifiers with the rNDA we have developed in the previous chapter. We artificially inject symmetric class label noise ranging from 10% to 50% into the datasets. We hold out 20% of the data for testing. At each level of noise we perform 100 independent repetitions. We report the error rates on the test sets in the form of the mean and standard deviation over the independent repeats. The results should enable us to draw conclusions on the performance of the proposed methods trained under the presence of labelling errors.

Secondly, we study empirically the effect of two types of label flipping: symmetric and asymmetric. Symmetric flipping is when each class is affected by label flipping in the same proportion. In contrast, asymmetric flipping is when the labels flip from one class to another and not vice-versa. This type of noise is expected to degrade the performance of traditional classifiers to a larger degree, since it modifies the decision boundary between the true classes.

In order to answer our third research question above, we employ the Receiver Operating Characteristic (ROC) curves to demonstrate the capability of our new algorithms to identify the mislabelled points. Finally, we view the gamma matrix as the adjacency matrix of a graph over the classes and examine the interpretability of the resulting graph in answer to our fourth question.

4.5.2 Datasets

We used two synthetic datasets and nine real world datasets to assess and demonstrate the effectiveness of rLR and rmLR. The first synthetic data, *Synth-1*, is generated by sampling points from a Gaussian distribution where the class label associated with each

point is assigned by a logistic function with a pre-defined weight vector $\mathbf{w}$. The dataset obtained is thus a Gaussian with each half belonging to a different class. This dataset clearly favours the discriminative model over the generative model. The second synthetic dataset, *Synth-2*, is a mixture of three Gaussians that we use to validate the performance of the multi-class classifier, rmLR.

For real world problems we used nine datasets from the UCI repository: The binary datasets *Boston*, *Pima*, *Liver*, *Adult* and *Websearch* are used to evaluate the classification performance of rLR, while the multi-class datasets *Iris*, *Wine*, *Newsgroups*[1] and *USPS* handwritten digits are employed to evaluate rmLR. Label noise of various levels was artificially injected for the purpose of systematic testing, except for *Websearch*, *Newsgroups* and *USPS*, in which we detect mislabelled points that are genuinely part of the data.

All datasets used, except *Newsgroups* and *USPS* (which are discrete valued), are standardised a priori. The latter two were normalised using cosine normalisation (Salton and Buckley [1988]), and the *Newsgroups* text corpus was also subject to tokenisation, stop words removal, and Porter stemming (Porter [1997]) to remove the word endings prior to cosine normalisation. We summarise the characteristics of each of these datasets in Table 4.3.

4.5.3 Simulated noise

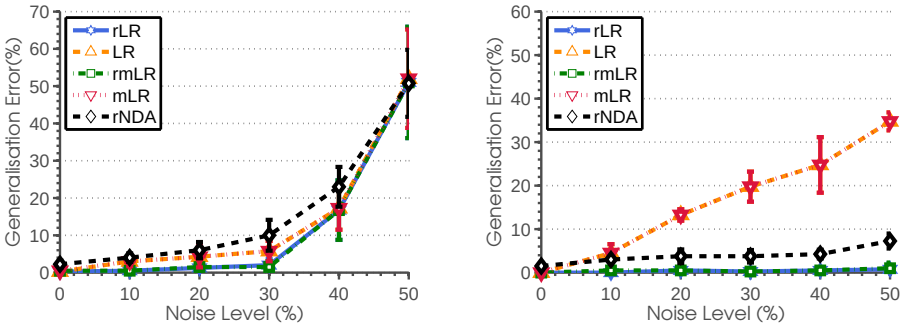
Results on synthetic data

We now present the numerical results from the experiments. Figure 4.2(a) shows the generalisation error of the algorithms on *Synth-1* dataset with symmetric label flipping. We see that rLR and rmLR achieve lower error rates than LR and mLR. They also outperform rNDA as expected. However, it is arguable that the improvement is marginal in this case, the difference in performance is less than just 5%. The reason for this is

[1]Originally the *Newsgroups* corpus comprises 20 classes of postings. We use the subset of 10 classes from Kabán et al. [2002], with term frequency count based encoding.

Dataset	# Samples	Dimensionality	# Classes
Synth-1	1000	5	2
Synth-2	1000	5	3
Boston	506	13	2
Pima	768	8	2
Liver	345	6	2
Adult	1000	6	2
Iris	150	4	3
Wine	172	13	3
Newsgroups	8000	300	10
USPS digits	11000	256	10
Websearch	1000	10038	2

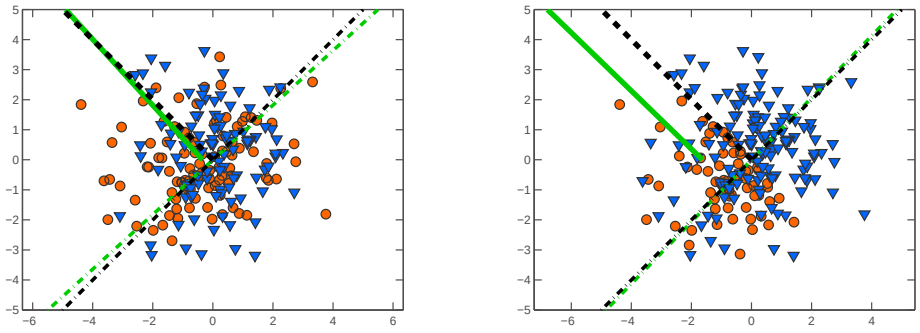
Table 4.3: Characteristics of the datasets employed in the reported experiments.



(a) Symmetric flipping (b) Asymmetric flipping

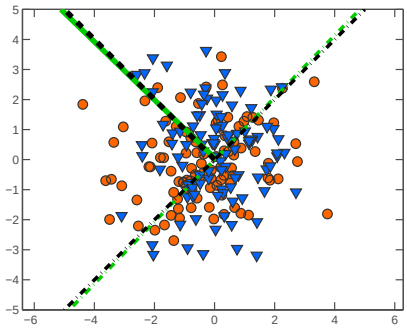
Figure 4.2: Classification error on *Synth-1* dataset

that even though the class labels are noisy, the flips are symmetrically distributed across both classes and so the decision boundary is not greatly affected. Hence, the traditional algorithms are still able to find near optimal decision boundary — which translates into near optimal classification performance. To verify our explanation, next we apply the asymmetric flipping mechanism. By doing so we expect that the decision boundary will be shifted away from the true one, which will fool the traditional algorithms. Figure 4.2 reports the performance of the algorithms under the asymmetric label flipping. It is clear that the improvement gained is much apparent than what we have obtained in the symmetric case, in general. However, it is worth noting that in an extreme case (50%

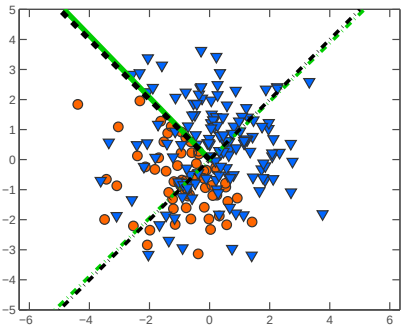


(a) LR - Symmetric flipping

(b) LR - Asymmetric flipping



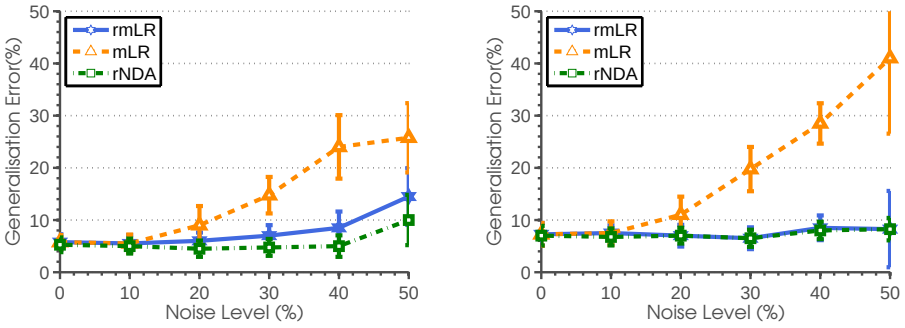
(c) rLR - Symmetric flipping



(d) rLR - Asymmetric flipping

Figure 4.3: Decision boundary obtained by LR and rLR at 40% symmetric and 40% asymmetric label noise respectively. The green lines are the estimates and the black lines are the true values. The dotted orange line represents $\mathbf{w}$ while the solid line represents the decision boundary. Clearly, the asymmetric noise shifts the true decision boundary.

noise) symmetric noise is also detrimental as it can destroy the original class structure such that an algorithm could go wrong totally. We also see that the rNDA is superior to the non-robust discriminative classifier. Still, it could not match rLR and rmLR. This is mainly because of the underlying distribution which is more challenging for the generative approach. To further illustrate the effect of asymmetric flipping, we present a 2D example in Figure 4.3. Here we see the decision boundary of LR and rLR obtained from applying the algorithms on a synthetic dataset at 40% asymmetric label noise. It is clear that asymmetric flipping indeed perturbs the decision boundary to a significant degree. Needless to say, this behaviour persists in the multi-class case too. Before moving



(a) Symmetric flipping (b) Asymmetric flipping

Figure 4.4: Classification error on *Synth-2* dataset

on, it may be useful to note also that the performance of rLR and rmLR is practically identical in these 2-class settings. This was expected, since rmLR is the generalisation of rLR, and so in fact rmLR can be used for both multi-class and binary classification problems.

Next, we validate our rmLR in a multi-class classification problem, using *Synth-2*. Figure 4.4 summarises the results and shows that mLR stays effective up to 10% noise but after that point rmLR gives better results. As in the binary case, again our algorithm with noise modelling is substantially superior in comparison to its traditional counterpart. We also see that the rNDA has a slight edge over rmLR. This is because the dataset, which is a mixture of Gaussian, fits the Gaussianity assumption of the rNDA.

Results on real data

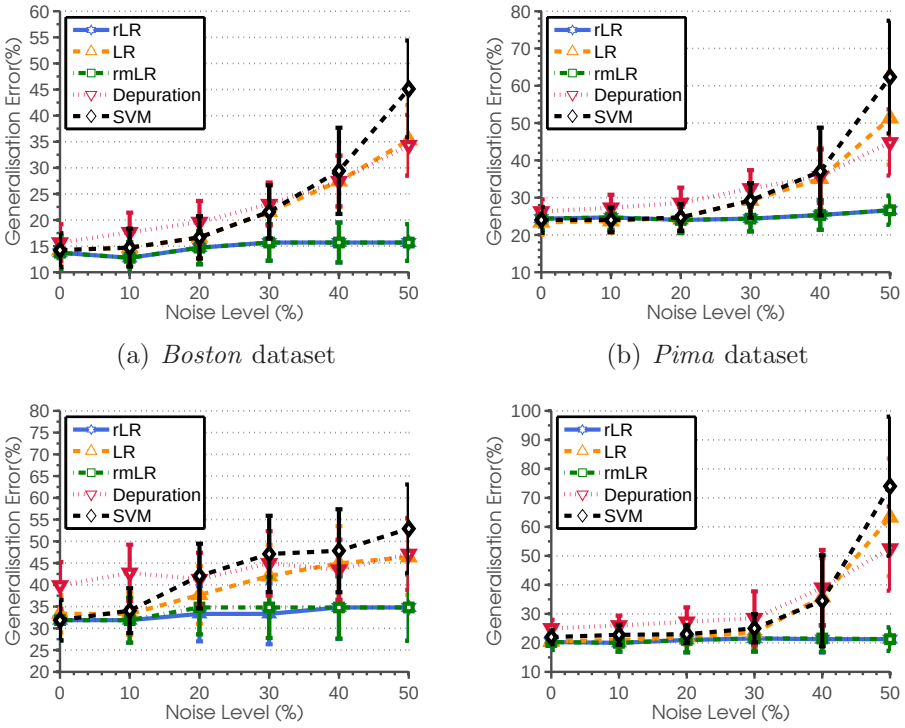
Real data tends to be more complex, so we further assess our methods on real world datasets. In addition, we will compare our result to two existing methods: (i) Depuration (Barandela and Gasca [2000]), which is a non-parametric method based on nearest neighbours, previously proposed for the same problem of dealing with label-noise in classification; and (ii) Support Vector Machines (SVM) (Cortes and Vapnik [1995]), which has the well-known margin and slack-variable mechanism built in, and which may provide some robustness. The reason to compare with SVM is to find out to what extent class

label noise could be considered to be a normal part of any classification problem — and conversely, to what extent it actually needs the special treatment that we developed in the previous sections. Since rLR and rmLR are both linear classifiers, we used linear kernel in the SVM. We shall, however, study non-linear problems in Chapter 6.

It should be noted that when applying Depuration and SVM, we again face with the problem of model selection. For a nearest-neighbour based algorithm, the number of neighbours needs to be determined as well as the model's additional threshold parameter. SVM requires an optimal C parameter to be set. A general approach to model selection is a standard cross validation technique. Although this works well in a traditional setting where all class labels are correct, it is no longer applicable here. This problem was also reported in Lawrence and Schölkopf [2001], where they need to choose the hyperparameter of their kernel Fisher discriminant. However, the solution they resort to is simply to assume that there is a trusted validation set available. This may be unrealistic in many real situations, and especially so in small-sample problems as in Zhang et al. [2009]. Moreover, even if we have a trusted validation set, the additional computation time required might also be a concern.

For Depuration, this problem is still open, and we use the default values suggested in Sánchez et al. [2003], where it was found that a neighbourhood size of $k = 3$ and a threshold of $k' = 2$ works well in practice. For the SVM, we will do an idealised hyperparameter selection for the purpose of our comparison experiments: we use cross validation on the clean data to pick the hyperparameter C and then use that value throughout the noisy label experiments. Clearly this should work in favour of SVM, so if SVM is unable to outperform our methods in label noise conditions then we can conclude that modelling the noise was necessary.

Figure 4.5 summarises our results on four binary classification datasets. It is clear that both rLR and mrLR outperform each algorithm on each of the datasets tested, except



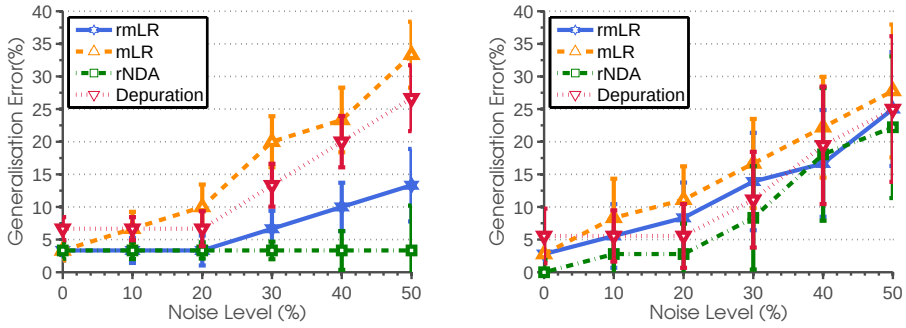
(a) *Boston* dataset (b) *Pima* dataset

(c) *Liver* dataset (d) *Adult* dataset

Figure 4.5: Classification errors on real world datasets when the labels are asymmetrically flipped

when there is no label noise in the dataset. At 0% noise, SVM, LR and mLR output the best result. In that case, the extra complexity of modelling the noise probabilities in unnecessary and the price to pay for estimating these extra parameters is a slightly worse performance or the necessity of more training points. However, in all noisy cases the improvement is apparent. Depuration tends to perform well in a very high level of noise (i.e. 40% upwards) while at the lower range, its performance is slightly worse.

The comparative results with SVM also demonstrate convincingly that class label noise does need special attention and it is naive to consider label noise as a normal part of classification problems. We see that our algorithm developed explicitly for this problem does indeed achieve improved classification performance overall. The picture is consistently similar in multi-class problems. Figure 4.6(a) and Figure 4.6(b) show



(a) *Iris* dataset (b) Wine dataset

Figure 4.6: Classification errors on multi-class problems using the *Iris* and *Wine* datasets. The labels are asymmetrically flipped.

the performance of rmLR for the *Iris* and *Wine* datasets. Again, the proposed modelling approach significantly outperforms the traditional approaches in asymmetric label-flipping conditions on both datasets tested. We also see the rNDA also performs very well. It stays effective up to 30% noise in *Iris* dataset and even outperforms rmLR in *Wine* dataset. We speculate that the underlying distributions of these two datasets are approximately Gaussian.

Next, we assess our methods' ability to detect the instances that were wrongly labelled. There are two types of possible errors: (i) a false positive is when a point is believed to be mislabelled when in fact it is labelled correctly; and (ii) a false negative is when a point is believed to be labelled correctly when in fact its label is incorrect. A good way to summarise both, while also using the probabilistic output given by the sigmoid or the softmax functions, may be obtained by constructing the Receiver Operating Characteristic (ROC) curves. Figure 4.7 shows the ROC curves for all six real world datasets tested, at an asymmetric noise level of 30%. Superimposed for reference we also plotted the ROC curves that correspond to the traditional classifier that believes that all points have the correct labels. The gap between the two curves is well apparent in all six cases tested, and it quantifies the gain obtained by our modelling approach in each setting. The area under the ROC curve signifies the probability that a randomly drawn and mislabelled

example would be flagged by our method. For the sake of clarity of the graph, the results from Depuration and SVM were not included here as we have already seen that they are inferior to rLR and rmLR. What is surprising about these results is that despite the real world datasets are not perfectly separable by a linear classifier, and the shapes of their multivariate distributions are not controlled in any way, still, our classifiers have been able to capture essential regularities in the data so that label assignments that are inconsistent with it turn out to predict the actual mislabelling to a relatively high degree of accuracy.

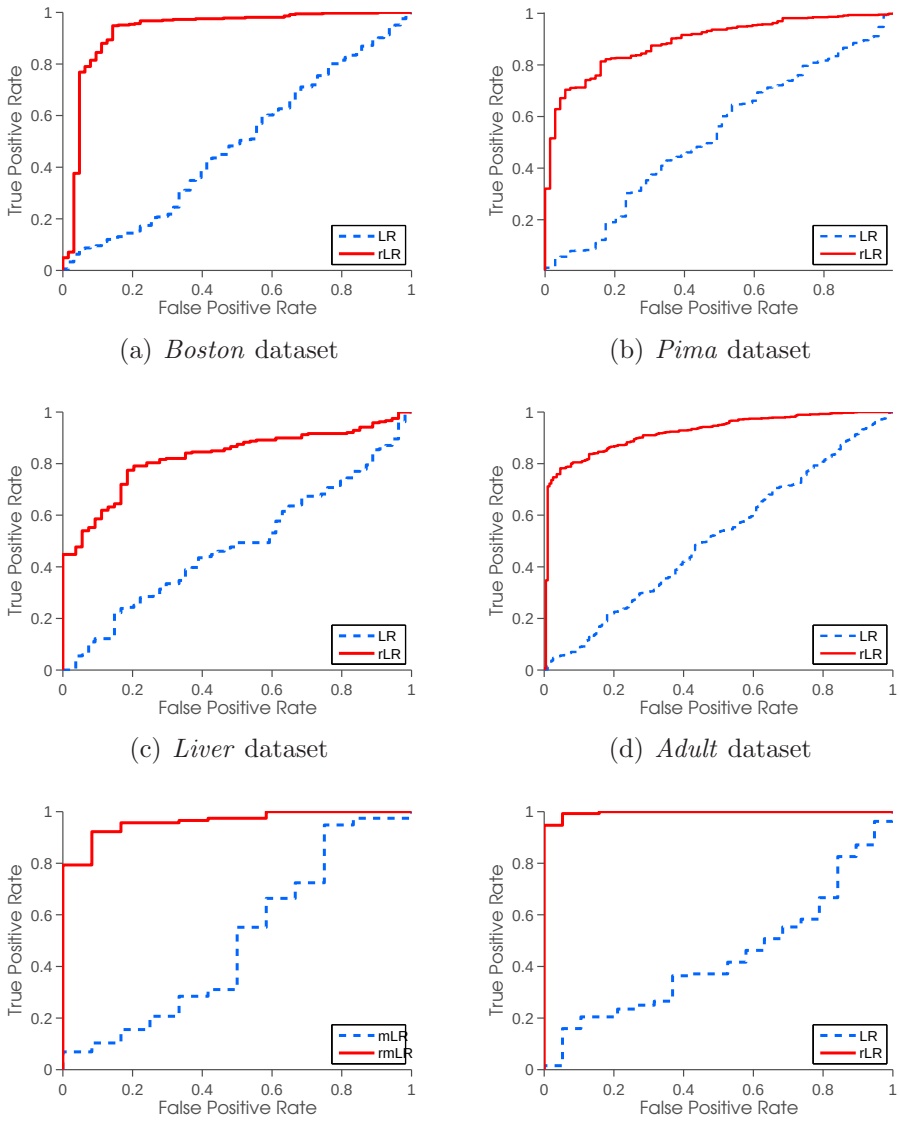
4.5.4 Real data with inaccurate label

So far we presented controlled experiments where the label-noise was artificially created. It is now most interesting to demonstrate the effectiveness of our approach on a dataset whose labels are genuinely inaccurate.

Learning to classify images using cheaply obtained labelled data

It is well reckoned that careful labelling of large amounts of data by human experts is extremely tiresome. Suppose we were to train a classifier to distinguish an image of 'bike' from other type of images. The standard machine learning approach is to collect training images and manually label each of them — rather labourious. Here, we suggest that we could reduce human expert intervention and obtain the training data cheaply using annotated data from search engines. By searching for images using keyword 'bike' we obtain a set of images that are loosely categorised into 'bike' class, and similarly 'not bike' class by using its negation. This allows us to acquire a large number of training data quickly and cheaply. The problem is of course that the annotations returned by the search engine are somewhat unreliable. This is where rLR comes into play. Here we collected 515 images using the keyword 'bike' and 515 images using the keyword 'not bike' that we call the *WebSearch*[1] dataset. We also manually label all images: a 'bike'

[1]Collected using Google image, available at http://cs.bham.ac.uk/∼jxb008/data/websearch.zip



(a) *Boston* dataset (b) *Pima* dataset

(c) *Liver* dataset (d) *Adult* dataset

(e) *Iris* dataset (f) *Wine* dataset

Figure 4.7: Receiver Operating Characteristic curves. Labels are asymmetrically flipped at 30% noise

image is one that contains a bike as its main object and we make no distinction between a bicycle and a motorbike, everything else is labelled as 'not bike'. This reveals 83 flips from 'bike' to 'not bike' images and 100 flips from 'not bike' to 'bike' category. The manually labelled set is only used for testing purposes. The images are passed through a series of preprocessing including extracting meaningful visual vocabulary using SIFT (Lowe [1999]) and extracting texture information using LBP (Ojala et al. [2002]), which are ultimately transformed into a 10038-dimensional vector representation.

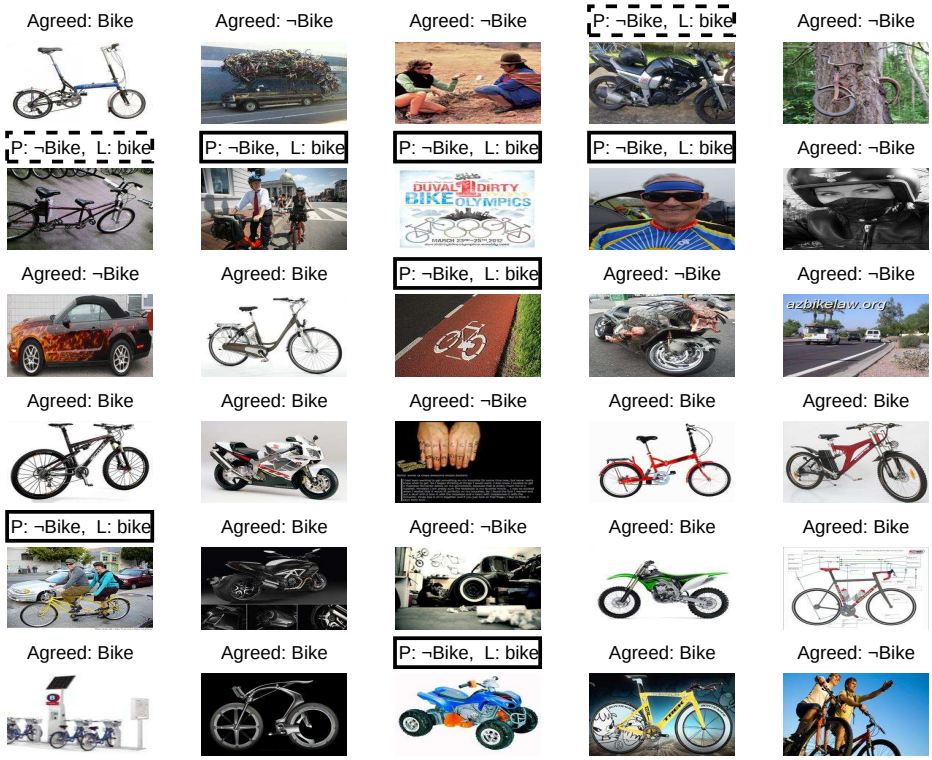


Figure 4.8: Bike search result. P is the prediction from the classifier while L is the given label from search engine. Boxed instances are the ones that P and L don't agree while dotted boxes are false alarms.

In Figure 4.8 we show examples of detecting mislabelled images. The top 30 test images sorted by their posterior probabilities are shown. We see that out of a total of 8 suspicious detections made (boxed), only 2 were false alarms (denoted by dotted box in the figure). Comparatively, the traditional LR model produced 4 false alarms (not

shown). To give statistical figures, we then tested these two classifiers that were both trained on 90% of whole dataset using the cheap noisy labels from the search engine, and tested on the remaining 10%, against the manual labels. We performed 100 independent bootstrap repetitions of this experiment. The average generalisation error rates and their standard errors were 15.67% ± 0.04 for rLR and 18.09% ± 0.04 for standard LR. The improvement of rLR over LR is statistically significant, as tested at the 5% level using a Wilcoxon Rank Sum test. This suggests that there is high potential for learning from unreliable data from the Internet using the label-noise robust algorithm proposed.

Inferring a class topology in multi-class problems, and detecting peculiar or mislabelled instances

The last set of experiments demonstrates a different use of our label-robust classifier, namely to infer the internal topological structure of the data classes. For many real-world classification tasks the labelling process is somewhat subjective as there is no clear-cut boundary between the classes. For example, in the case of classifying text messages according to topic, some instances could be assigned to more than one category. In fact we can imagine the topological relationship between the topic classes in the form of a graph. We posit that this relationship graph can be captured by our algorithm via the gamma table. Thus, interpreting the gamma matrix as the adjacency matrix of a directed graph could reveal the internal structure of the dataset under study. To demonstrate this idea, we employed rmLR on the *10 Newsgroups* and the *USPS* handwritten digits dataset, and we analyse the gamma matrices obtained.

Figure 4.9 shows the graph derived from the gamma matrix as obtained from *10 Newsgroups*. Each node corresponds to a topic class while the length of an edge connecting two nodes represents the strength of relationship between them. The direction of arrows then correspond to the label flipping directions. To draw the graph we used the Pajek network visualisation software (Batagelj and Mrvar [1998]). It can be seen from this graph

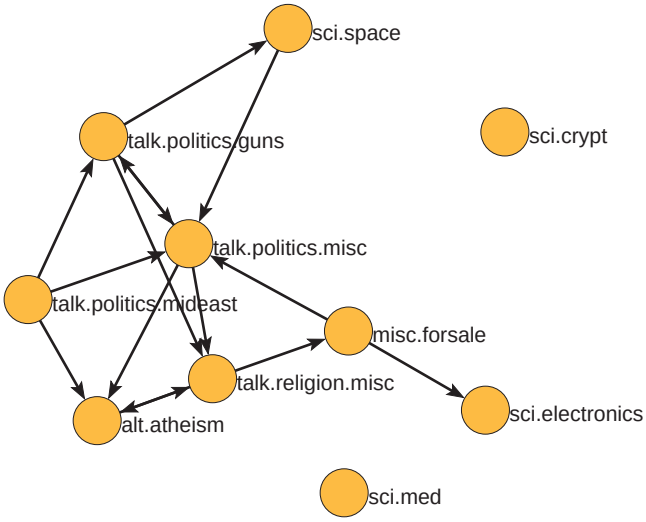


Figure 4.9: Adjacency graph of the ten topics on the *Newsgroups* dataset.

that "atheism" and "religion" are related topics by looking at the distance between the two as well as the bi-directional flipping relation, which indeed agrees with our commonsense. A similar observation can also be made between the "electronics" and "for-sale" postings. Further, the graph also visually suggests various sub-groupings: For example, all classes related to politics are clustered nearer to each other. Further, besides this global view of the whole dataset we can look up the actual content of the messages that have the highest mislabelling probabilities. Five such examples are given in Table 4.4. The ambiguity of topic label assignments is quite apparent in all of these cases.

Labelled	sci.electronics	talk.religion.misc	talk.religion.misc	talk.politics.misc	talk.politics.guns
Predicted	misc.forsale	alt.atheism	talk.politics.misc	talk.politics.guns	talk.religion.misc
	'sell'	'peopl'	'write'	'peopl'	'write'
	'price'	'true'	'state'	'price'	'fire'
	'don'	'belief'	'peopl'	'american'	'system'
	'electron'	'statem'	'de'	'death'	'fact'
	'comput'	'moral'	'point'	'person'	'don'
	'circuit'	'articl'	'fact'	'card'	'forc'
	'condition'	'thing'	'claim'	'kill'	'happen'
		'point'	'email'	'children'	'bill'
		'horu'		'bill'	'commit'
		'frank'		'koresh'	'center'

Table 4.4: Examples of text messages that are inferred to be most likely mislabelled. The content of each message is displayed using up to ten word stem features.

The first example in Table 4.4 was originally labelled as "sci.electronics" while our

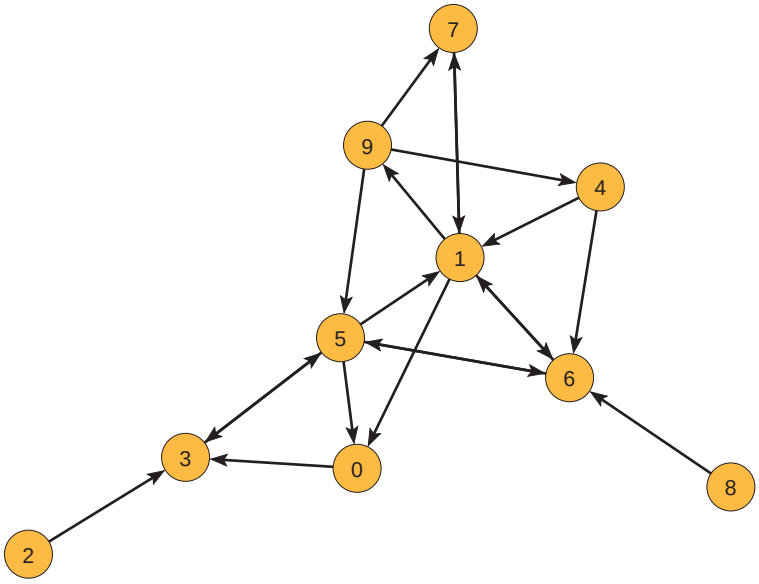
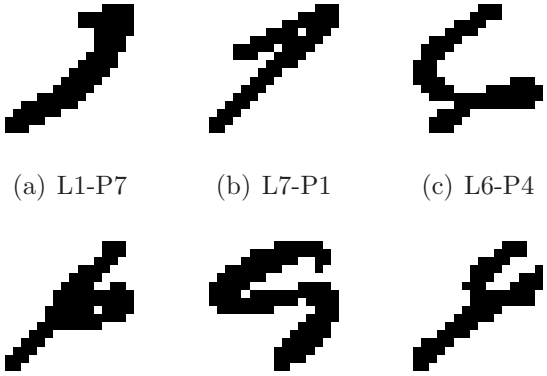


Figure 4.10: Adjacency graph of the ten classes of the *USPS* handwritten digits dataset.

algorithm predicts that it should rather belong to "misc.forsale". Indeed, terms such as 'sell', 'price' and 'condition' that appear in this document support this prediction. Through this analysis we found that "sci.med" and "sci.crypt" are topic classes that are unrelated to the rest of topics, while "sci.space" is related to topics about guns and politics. It is worth mentioning that the gamma matrix needs not be symmetric. That is because, say classified ads may contain terms related to electronics but electronics postings may never have terms like 'sell', 'price' and 'condition'.

We performed a similar experiment to obtain the class relationship graph of the *USPS* handwritten digits dataset. The result is presented in Figure 4.10. At a first glance, the resulting graph again agrees well with our common-sense as, for example, digit 7 and digit 1 can sometimes be very difficult to distinguish in handwriting. A similar confusion arises between digit 6 and 5, digit 6 and 4, and so on. Though, we could imagine many other cases where digit n has been written too similarly to digit m, and those imaginary cases may never be found in our dataset. For more concreteness, we show in Figure 4.11 the instances that our classifier infers to be most likely mislabelled.



(a) L1-P7 (b) L7-P1 (c) L6-P4

(d) L6-P4 (e) L9-P5 (f) L4-P9

Figure 4.11: Examples of the images that the algorithm infers to be most likely mislabelled. Ln-Pm means that the image was labelled as digit n but was predicted by the algorithm as digit m.

The first two of these examples confirm that 1 and 7 are sometimes written very similarly which could induce a confusion to both the human labeller as well as the algorithm. Indeed, there is no absolute right or wrong when assigning an image to a particular class in all cases, and our method is able to find the most peculiar examples. The image could be perceived differently depending on one's mental experience and interpretation. The point is, it is not easy to argue that those confusing images were actually wrongly labelled but we can at least see how the classes of digits, as suggested by the algorithm, are distributed. This can be useful when one would like to know more about the underlying structure of the dataset. From the other viewpoint, if we happened to have prior knowledge of the distribution of classes, we could pre-define the gamma matrix in hoping for a better classification performance.

4.6 Summary

We proposed a label-robust logistic regression algorithm for both two-class (rLR) and multi-class (rmLR) classification learning in the presence of labelling errors. The numerical experiments on artificial data with simulated label noise and on real applications

that genuinely contain label noise suggest that rLR and rmLR are superior to their traditional counterparts when the training data contains labelling errors, and more significantly so when the label-flipping distribution is asymmetric. These empirical observations are backed up by an error analysis and a model decomposition analysis.

In the next chapter we shall investigate more about an important and challenging real application of the proposed discriminative model. We will study the classification of microarray data which has been reported to contain wrong labelled samples with biological evidence.

CHAPTER 5

Robust Bayesian Logistic Regression

High-throughput microarray technologies make it possible to simultaneously measure the expression levels of thousands of genes. Our ability to use these data to reliably predict the presence of a certain disease and to better understand the biological mechanisms underlying the development of disease is of fundamental importance from the perspective of treatment and prevention. Statistical machine learning methods have already shown a lot of promise towards these goals, and methods that can deal with high dimensional and low sample size settings have been the subject of considerable research efforts over the last decade. Previous studies reported that labelling errors are not uncommon in microarray datasets. In such cases the training set may become misleading, and the ability of classifiers to make reliable inferences from the data is compromised.

In this chapter we address the above problems by developing an integrated approach where the ambiguity of the given label assignments is modelled explicitly during the training of a classifier. This allows us to build on classifiers that have been successful for microarray classification by developing an extension to account for possible label noise. Specifically, here we will harness the sparse Bayesian logistic regression (BLogReg) model proposed by Cawley and Talbot [2006] with a robustness against label noise. From our

model formulation, we then derive a new algorithm that alternates between training the classifier and estimating the label noise probabilities. The regularisation parameter is automatically set using Bayesian regularisation, which not only saves the computation time that cross-validation would take, but also eliminates any unwanted effects of label noise when setting the regularisation parameter. Extensive experiments with both synthetic data and real microarray datasets demonstrate that our approach is able to counter the bad effects of labelling errors in terms of predictive performance, it is effective at identifying marker genes, and simultaneously it detects mislabelled arrays to high accuracy.

5.1 The model

The robust model used in this chapter is identical to that presented in Chapter 4. We shall revisit the model formulation for the sake of readability. Consider a set of training data $S = \{(\mathbf{x}_n, \tilde{y}_n)\}_{n=1}^N$, where $\mathbf{x}_n \in \mathbb{R}^M$ and $\tilde{y}_n \in \{0, 1\}$, where $\tilde{y}_n$ denotes the observed label of $\mathbf{x}_n$. Using a latent variable y to denote the true class label the robust objective can be written as:

$$\mathcal{L}(\mathbf{w}, \Gamma) = \sum_{n=1}^{N} \tilde{y}_n \log \tilde{P}_n^1 + (1 - \tilde{y}_n) \log \tilde{P}_n^0 \tag{1}$$

where

$$p(\tilde{y}_n = k|\mathbf{x}_n, \mathbf{w}) = \sum_{j=0}^{1} p(\tilde{y}_n = k|y = j)p(y = j|\mathbf{x}_n, \mathbf{w}) \stackrel{def}{=} \tilde{P}_n^k \tag{2}$$

We decide that $\mathbf{x}_q$ belongs to class 1 whenever $p(y = 1|\mathbf{x}_q, \mathbf{w}) \geq 0.5$.

5.1.1 Sparsity prior

Microarray data are high dimensional with more features than observations while only a subset of the features is relevant to the target. A vast literature demonstrates that sparsity-inducing regularisation approaches are effective in such cases (MacKay [1995], Shevade and Keerthi [2003], Cawley and Talbot [2006]). Hence we now incorporate spar-

sity in our model described in the previous section. Following Shevade and Keerthi [2003] and Cawley and Talbot [2006], we will employ an $L1$ regularisation term which results in the following objective function:

$$\max_{\mathbf{w}} \sum_{n=1}^{N} \log p(\tilde{y}_n|\mathbf{x}_n, \mathbf{w}) - \lambda\|\mathbf{w}\|_1 \tag{3}$$

where λ is the Lagrange multiplier (or regularisation parameter) that balances between fitting the data well and having small parameter values. The $L1$-norm in the regularisation term is defined as,

$$\|\mathbf{w}\|_1 = \sum_{m=1}^{M} |w_m| \tag{4}$$

Now, the regularisation parameter λ needs to be determined. We cannot use cross-validation, not only for its computational demand, but primarily because it would need a validation set with trusted correct labels, which may be not available. Hence we adopt the Bayesian regularisation approach of Cawley and Talbot [2006], which bypasses the need for cross validation and determines λ automatically by putting a Jeffreys' prior on λ and integrating it out from the model. This yields the following (see Cawley and Talbot [2006] for details)[1].

$$\lambda = \frac{P}{\sum_{d=0}^{P} |w_d|} \tag{5}$$

where P denotes the number of non-zero parameters, i.e. those with $w_d \neq 0$ – so $P \leq M$.

5.2 Parameter estimation

It now remains to estimate $\mathbf{w}$ and Γ. Notice that Eq.(3) is not differentiable at the origin. Therefore, the algorithm we proposed in Chapter 4 is not applicable. Fortunately, Shevade and Keerthi [2003] proposed a simple yet effective algorithm to optimise the non-smooth

[1]The reader should refer to Chapter 6 where we also treat model selection problem using Bayesian regularisation.

but convex objective function of sparse logistic regression (SLogReg) using the Gauss-Seidel method and employing coordinate-wise descent. We will create a modification of this approach in order to make it applicable to our non-convex objective.

5.2.1 Updating the weight vector

We define $F_d = \frac{\partial \mathcal{L}(\mathbf{w},\Gamma)}{\partial w_d}$, where $w_{d=0}$ is the bias term that is usually left unregularised. The first order optimality conditions for the objective function can be derived from geometry, similarly as in Shevade and Keerthi [2003] and Cawley and Talbot [2006]:

- Case 1: d = 0

  As w_0 corresponds to the bias term which needs not be regularised, we have that Eq.(3) is differentiable w.r.t. w_0 and attains its optimum when $F_0 = 0$. Any value of F_0 that deviates from zero is an 'optimality violating state'. The degree of violation is quantified as $viol_0 = |F_0|$. Note that it is difficult to achieve $viol_d = 0$ because the algorithm is asymptotically convergent. In practice, we only need to ensure that $viol_d$ is smaller than some pre-defined tolerance τ, i.e. $viol_d \leq \tau,\ \forall_d$.

- Case 2: $w_d \neq 0, d \neq 0$

  When $w_d > 0$, Eq.(3) is not differentiable at zero but it is differentiable on $w_d \in (0, \infty)$. Taking derivative of Eq.(3) w.r.t. w_d and equating it to zero, we have $\lambda - F_d = 0$.

  Likewise, when $w_d < 0$ we have $-\lambda - F_d = 0$. We thus define the value of $|\lambda - F_d|$ and $|\lambda + F_d|$ as the 'optimality violating state'.

- Case 3: $w_d = 0, d \neq 0$

  The Eq.(3) is only directionally differentiable at zero. We require that 1) the right-side derivative be non-negative, $\lambda - F_d \geq 0$ and 2) the left-side derivative be non-positive, $-\lambda - F_d \leq 0$. Combining both requirements we have, $-\lambda \leq F_d \leq \lambda$. A violation is then defined to be the case where F_d falls out of this interval.

The optimality conditions for Eq.(3) can be stated algebraically as the following:

$$\begin{array}{lcll} F_d & = & 0 & \text{if } d = 0 \\ F_d & = & \lambda & \text{if } w_d > 0, d > 0 \\ F_d & = & -\lambda & \text{if } w_d < 0, d > 0 \\ -\lambda \leq & F_d & \leq \lambda & \text{if } w_d = 0, d > 0 \end{array}$$

Accordingly, the violation from optimality of w_d may be summarised as:

$$\begin{array}{lcll} viol_d & = & |F_d| & \text{if } d = 0 \\ & = & |\lambda - F_d| & \text{if } w_d > 0, d > 0 \\ & = & |\lambda + F_d| & \text{if } w_d < 0, d > 0 \\ & = & \max(F_d - \lambda, -\lambda - F_d, 0) & \text{if } w_d = 0, d > 0 \end{array}$$

We start optimising the component w_d that makes the largest violation to an optimality condition. At this point, if the objective function was convex then it would be possible to use gradient information to bracket the region where the optimal w_d lies by specifying upper and lower limits (H and L). For example, Shevade and Keerthi [2003] identify 10 different cases for their sparse logistic regression model (Table 5.1). However, since our likelihood term is non-convex, the cases identified there are not applicable because the sign of gradients give no information about the interval where the optimal solution resides. Therefore we introduce a simple modification by performing two searches: one in the range $\mathbb{R}^+ \cup \{0\}$ and another in the range $\mathbb{R}^- \cup \{0\}$. We then choose the solution that returns a higher value of the objective function. This modified searching approach is more general and will work on any locally differentiable function at the expense of a slight increase in computation time. In practice, L and H are finite – provided that the design matrix is standardised and appropriate regularisation is imposed on the solution, it is sufficient to search in the $(0, 1000)$ and $(-1000, 0)$ intervals.

5.2.2 Updating the label flipping probabilities

Finally, having completed the optimisation of $\mathbf{w}$, it remains to derive the update rule for the label-flipping probabilities. Conveniently, these can be estimated via fixed point

Case	w_d	F_d	L-hand F_d	R-hand F_d	L	H
1	0	-	< 0	< 0	0	$+\infty$
2	0	-	> 0	> 0	0	$-\infty$
3	< 0	> 0	-	-	$-\infty$	w_d
4	> 0	< 0	-	-	w_d	$+\infty$
5	< 0	< 0	> 0	-	w_d	0
6	> 0	> 0	-	< 0	0	w_d
7	< 0	< 0	-	> 0	0	$+\infty$
8	> 0	> 0	< 0	-	$-\infty$	0
9	< 0	< 0	≤ 0	≥ 0	0	0
10	> 0	> 0	≤ 0	≥ 0	0	0

Table 5.1: Bracketing cases for convex objective function in BLogReg.

update equations. By introducing a Lagrange multiplier to ensure that the probabilities in each row of the gamma table sum to 1 and solving the stationary equations, we obtain the following update equations. Derivation details are similar to those presented in Chapter 4.

$$\gamma_{00} = \frac{g_{00}}{g_{00} + g_{01}}, \gamma_{01} = \frac{g_{01}}{g_{00} + g_{01}} \tag{6}$$

$$\gamma_{10} = \frac{g_{10}}{g_{10} + g_{11}}, \gamma_{11} = \frac{g_{11}}{g_{10} + g_{11}}. \tag{7}$$

where

$$g_{00} = \gamma_{00} \sum_{n=1}^{N} \frac{(1 - \tilde{y}_n)}{\tilde{P}_n^0} (1 - \sigma(\mathbf{w}^T \mathbf{x}_n)) \tag{8}$$

$$g_{11} = \gamma_{11} \sum_{n=1}^{N} \frac{\tilde{y}_n}{\tilde{P}_n^1} \sigma(\mathbf{w}^T \mathbf{x}_n) \tag{9}$$

$$g_{01} = \gamma_{01} \sum_{n=1}^{N} \frac{\tilde{y}_n}{\tilde{P}_n^1} (1 - \sigma(\mathbf{w}^T \mathbf{x}_n)) \tag{10}$$

$$g_{10} = \gamma_{10} \sum_{n=1}^{N} \frac{(1 - \tilde{y}_n)}{\tilde{P}_n^0} \sigma(\mathbf{w}^T \mathbf{x}_n) \tag{11}$$

The optimisation of the log-likelihood is then to alternate between optimising $\mathbf{w}$ along

with updating λ according to Eq.(5) until convergence is reached, and we alternate this with the fixed point update equations of the label flipping probabilities. The entire optimisation procedure is summarised in Algorithms 4-5.

Algorithm 4 Main loop

Input: Training examples.
  Initialise $\mathbf{w} \leftarrow 0$, $\lambda \leftarrow 0$, $I_{nz} \leftarrow \{w_0\}$, $I_z \leftarrow \{w_{d,d\in\{1,M\}}\}$, Γ.
  while Optimality violator exists in I_z **do**
    Find the greatest optimality violator, ν, in I_z
    repeat
      Optimise w_ν using Algorithm 5
      $I_z \leftarrow I_z \backslash \{w_\nu\}$
      $I_{nz} \leftarrow I_{nz} \cup \{w_\nu\}$
      Find the maximum optimality violator, ν, in I_{nz}
    until No violator exists in I_{nz}
    Update the entries of Γ by Eqs.(6)-(7)
    Update regularisation parameter, λ by Eq.(5)
  end while
Output: Optimised weight vector, $\mathbf{w}$. Optimised Γ.

Algorithm 5 Optimisation of $\mathbf{w}$

Input: Violating component, w_ν, I_z, I_{nz}
  $w_\nu \leftarrow 0$
  if w_ν satisfies optimality conditions **then**
    $I_z \leftarrow I_z \cup \{w_\nu\}$
    $I_{nz} \leftarrow I_{nz} \backslash \{w_\nu\}$
    break
  else
    Restore previous value of w_ν
    $t_1 \leftarrow$ Optimise Eq.(1) w.r.t. w_ν in the range $(-lim, 0)$ range
    $t_2 \leftarrow$ Optimise Eq.(1) w.r.t. w_ν in the range $(0, lim)$ range
  end if
  $w_\nu \leftarrow t_i$ that maximise Eq.(1), where $i \in \{1, 2\}$.
Output: Optimised w_ν

5.3 Detecting mislabelled points

For an observation $(\mathbf{x}_n, \tilde{y}_n)$, the probability of it being mislabelled can be computed as the following:

$$p(y \neq \tilde{y}_n|\mathbf{x}_n) = \sum_{j=0, j\neq\tilde{y}_n}^{1} p(y = j|\mathbf{x}_n) \tag{12}$$

This may be thought of as the models "degree of belief" that $\mathbf{x}_n$'s label is incorrect. We may use it either in this form, or in a hard-thresholded form (i.e. predict that the point $\mathbf{x}_n$ is mislabelled if $p(y \neq \tilde{y}_n|\mathbf{x}_n) \geq 0.5$).

5.4 A note on low sample size, high dimensional data

Since additional parameters Γ are being estimated from the data, we expect that RLogReg will require more training examples to deliver its full potential. In microarray datasets the training set size is often of the order of tens only. A possible workaround in such cases is to guide the algorithm by presetting the gamma table from domain knowledge about the likely proportion of mislabelled data. When such knowledge exists, the values of gamma may either be fixed through-out the optimisation process or they may be seeded initially and then optimised.

5.5 Empirical evaluation

5.5.1 Experiment setting

We will compare the classification performance of RLogReg, RLogReg with fixed gamma table (denoted RLogReg-F) and its traditional counterpart, i.e. BLogReg of Cawley and Talbot [2006]. The reader is referred to Cawley and Talbot [2006] for a comparison between BLogReg against the Relevance Vector Machine (RVM) and SLogReg (Shevade and Keerthi [2003]) where BLogReg was shown to be superior. We shall demonstrate that

our proposed robust extension of BLogReg performs better than the original BLogReg in terms of classification performance when there is label noise present in the training set. Moreover, our model can be used to identify mislabelled arrays for potential follow-on study.

Before proceeding, recall that symmetric and asymmetric label flipping have very different consequences in classification. Symmetric or uniform flipping means that each class is affected by label flipping in the same proportion. In contrast, asymmetric or non-uniform flipping is when the label flips from one class to another more often than vice-versa. The latter type of label flipping has been theoretically shown (Lugosi [1992]) to degrade the performance of an algorithm to a much larger degree, since it modifies the decision boundary between the true classes. Our empirical study in the previous chapter also demonstrated this. Therefore we will mainly focus our attention on datasets with asymmetric label noise and indeed expect the advantages of our approach to be most apparent in that setting.

To demonstrate the benefit of having a label noise model embedded in the classifier, we start with experiments on synthetic data where labels were asymmetrically flipped at the rate of 30%. The use of synthetic data for controlled experiments is standard in bioinformatics (see e.g. Zhang et al. [2009]), since it allows us assess the performance of a new approach against a ground truth. We shall then move on to analysing real microarray datasets where label noise has not been injected artificially. These datasets have been previously reported to contain wrongly labelled samples. Finally, we shall assess the ability of our proposed approach to identify mislabelled instances, employing Receiver Operating Characteristics (ROC) analysis.

5.5.2 Datasets

We generate these synthetic datasets following the protocol in Ng [2004], by sampling points from a standard Gaussian distribution where the class label associated with each point is assigned by a logistic function with a predefined weight vector $\mathbf{w}$. The pre-defined $\mathbf{w}$ were as follows:

- one relevant feature: $w_1 = 10, w_d = 0, \forall d > 1$
- three relevant features: $w_1 = w_2 = w_3 = 10/3, w_d = 0, \forall d > 3$
- features with exponentially decaying relevance: $w_d = (1/2)^{d-1}\sqrt{75}, \forall d \geq 1$.

For each of these, we create sets with 500 training points and sets with 100 training points together with independent test sets of 100 points, and call these datasets *Synth-500* and *Synth-100* respectively. The dimensionality of the synthetic datasets ranges from 100 up to 1000. Asymmetric label noise was artificially injected into each synthetic dataset at the 30% rate.

Further, we use three real microarray datasets: *Colon* (Alon et al. [1999]) *Breast* (West et al. [2001]) and *Leukaemia* (Golub et al. [1999])– all of which are known to contain some mislabelled arrays. No artificial label flipping is injected in these data. We standardise these datasets so the rows of the $N \times M$ design matrix (where N is the number of observations and M is the dimensionality) of the input sample will have zero mean and unit variance. Table 5.2 summarises the characteristics of all of these datasets employed.

5.5.3 Error measures

While in the case of synthetic data the true labels can be used to validate the predictive accuracy of our algorithm, in the real microarray data there is no absolute ground truth. Since the labels given in the datasets may be incorrect, the issue of what should count as

a miclassification must be defined. We define two variants for measuring out-of-sample error rates:

- Corrected (CRT): Count misclassification errors against the 'corrected' labels where corrections are made cf. the mislabellings reported in the literature.

- Cleansed (CLN) : Exclude any mislabelled suspects (known in the literature) from the test sets for the purpose of evaluation, so these are always placed into the training set instead; then count the misclassification errors on test sets in the usual way.

5.5.4 Results and discussion

Results on synthetic data

The average misclassification error rates on the *Synth-500* and *Synth-100* datasets are shown in Figure 5.1 as the data dimension is varied. Each point on these plots represents the average misclassification rate on the test sets, where the average is taken over 500 independent repetitions of the experiment. We see that RLogReg achieves substantially lower error rates than BLogReg on the datasets that contain more training examples (*Synth-500*). This clearly demonstrates the advantage of modelling the label noise process. On the smaller size dataset (*Synth-100*), however, the performance gain becomes marginal — this is because the accurate estimation of the additional parameters (label flipping

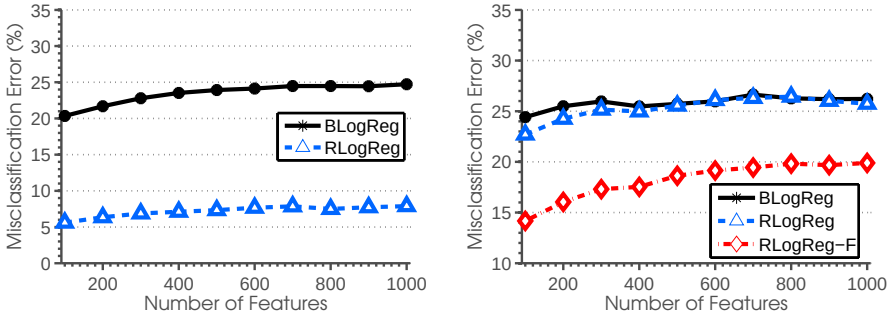
Dataset	# samples		# genes	# wrong labels	
	class 1	class 2		class 1	class 2
Synth-500	250	250	100-1000	0	75
Synth-100	50	50	100-1000	0	15
Colon cancer	40(T)	22(N)	2000	5	4
Breast cancer	25(ER+)	24(ER-)	7129	4	5
Leukaemia	25(AML)	47(ALL)	7129	1	0

Table 5.2: Characteristics of the datasets employed in the reported experiments.

probabilities) requires sufficient training data for our approach to achieve its full potential. Nevertheless, it is should be noticed that even in the small sample setting, RLogreg performs no worse than BLogReg on all the datasets tested. More importantly, the rightmost plot shows that we can counter the problem of small sample sizes by using prior knowledge about the extent of label noise, e.g. by pre-defining the gamma table. We denote this version as RLogReg-F in the figure, and we see this substantially improves the classification accuracy in the small sample setting. Beyond classification performance, it is of interest to evaluate the methods' ability to identify the relevant predictive genes. Figure 5.2 shows the estimated weight vectors as obtained by BLogReg and RLogReg respectively from 100-dimensional synthetic data with only the first 3 features being relevant. The classifiers were trained on 250 training examples per class that were subjected to 30% asymmetric label flipping. We see that RLogReg achieved a more accurate estimation of the weight vector, while BLogReg became confused by the noisy labels and selected too many false non-zero weights. This is an important advantage of RLogReg over BLogReg when it comes to finding a small set of predictive marker genes.

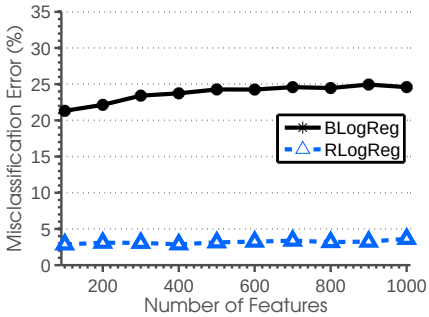
Results on *Colon* cancer dataset

The colon cancer classification task aims to distinguish between normal tissue and tumour. According to Alon et al. [1999] there is biological evidence that the samples T2, T30, T33, T36, T37, N8, N12, N34, N36 may be mislabelled. The proportion of mislabelling in the two classes is unequal, hence this is a case of asymmetric label flipping that can distort the correct decision boundary of the classes. The very limited number of training observations implies that a good estimate of the gamma table may be difficult to obtain from the data alone (as we have seen in the previous section), nevertheless prior knowledge of the noise proportions may still allow us to exploit the advantages of having a noise model as integral part of our classifier. Therefore we include RLogReg-F in our experiments, with the gamma table set to the true label flipping proportions. Table 5.3 reports the leave-one-

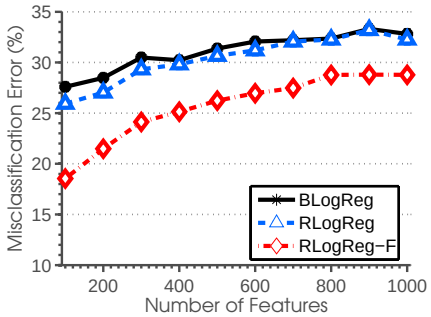


(a) *Synth-500*, 1 relevant feature

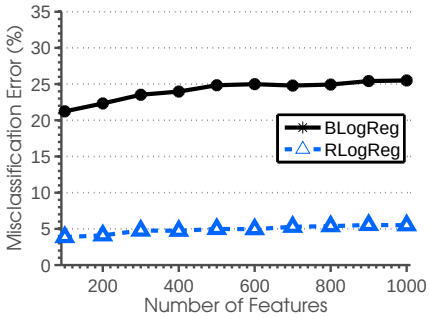
(b) *Synth-100*, 1 relevant feature



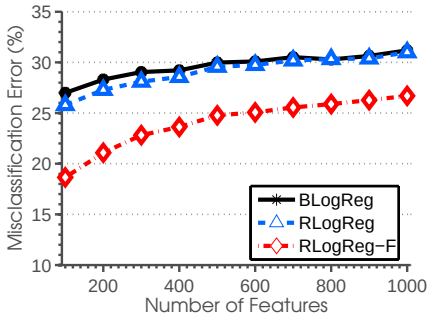
(c) *Synth-500*, 3 relevant features



(d) *Synth-100*, 3 relevant features



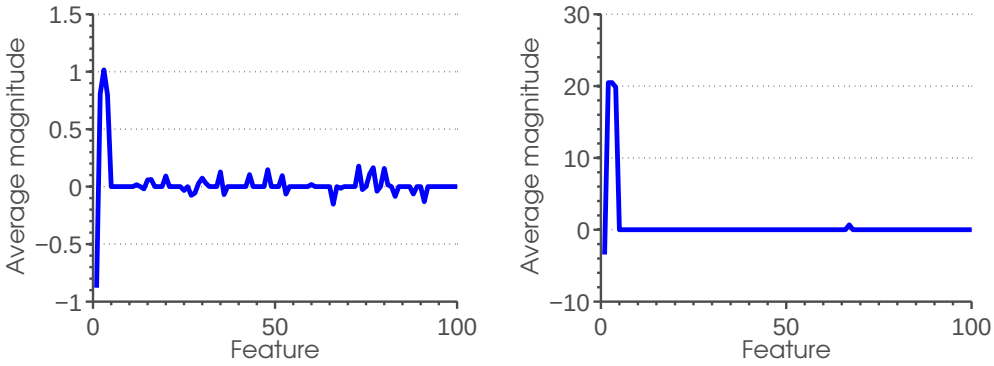
(e) *Synth-500*, exp decay relevance



(f) *Synth-100*, exp decay relevance

Figure 5.1: Misclassification on test set, as obtained by RLogReg and BLogReg respectively, on synthetic datasets with 30% asymmetric label noise. Left: Training sets of size 500; Right: Training sets of size 100. RLogReg-F denotes the version of RLogReg with the gamma matrix pre-set to its correct value.

out (LOO) errors in terms of the error measures defined in Sec. 5.5.3, and we also give the average number of genes selected by the three methods considered. The results confirm the expectations. RLogReg that attempts to estimate the gamma table along with all other



(a) BLogReg (b) RLogReg

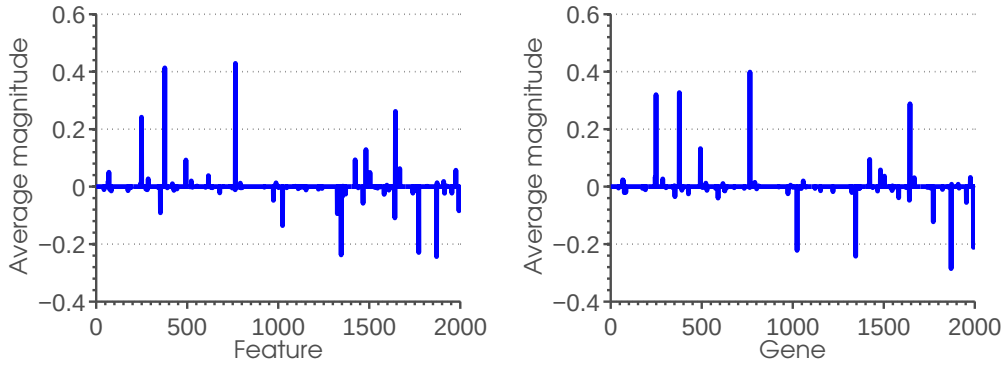
Figure 5.2: Comparison of the magnitude of weights for the 100 features as obtained in one run of BLogReg and RLogReg respectively, on synthetic data that contains only 3 relevant features (250 training examples in each class, 30% asymmetric label noise). We see that BLogReg selects too many features whereas RLogReg has a better ability to turn off the irrelevant ones.

Table 5.3: LOO misclassification (%) on *Colon* dataset. The average number of selected genes (± standard deviation) was computed from the CLN runs.

Algorithm	LOO-CRT	LOO-CLN	# Genes
BLogReg	8.06 ± 0.44	7.55 ± 0.64	11.94 ± 0.41
RLogReg	9.68 ± 0.48	9.43 ± 0.66	11.85 ± 0.41
RLogReg-F	4.83 ± 0.35	1.88 ± 0.54	9.21 ± 0.45

parameters is marginally worse than BLogReg (although not statistically significantly so, according to the unpaired t-test), while RLogReg-F improves over BLogReg in all validation criteria used, and it also selects a smaller fraction of relevant features.

Figure 5.3 shows the average magnitude of each gene according to BLogReg and RLogReg-F respectively. These are averages of $\mathbf{w}$ estimates across 1000 bootstrap repetitions in order to inspect possible systematic differences. These average weights turned out to be quite similar for BLogReg and RLogReg-F with the exception of a few genes that had been ranked differently by the two methods. To see this, a summary of top ten selected genes and their estimated weights are given in Tables 5.4-5.5.



(a) BLogReg (b) RLogReg-F

Figure 5.3: Comparison of the average weights of features selected by BLogReg and RLogReg-F on the *Colon* dataset over 1000 bootstrap repeats (50 train/12 test).

Table 5.4: Relative importance of top 10 genes selected by the BLogReg algorithm.

Gene No.	Gene Annotation	Avg. Magnitude
765	Human cysteine-rich protein (CRP) gene, exons 5 and 6	0.4289
377	H.sapiens mRNA for GCAP-II/uroguanylin precursor	0.4132
1644	C4-DICARBOXYLATE TRANSPORT SENSOR PROTEIN DCTB (Rhizobium leguminosarum)	0.2618
1870	PEPTIDYL-PROLYL CIS-TRANS ISOMERASE, MITOCHONDRIAL PRECURSOR (HUMAN)	-0.2435
249	Human desmin gene, complete cds.	0.2416
1346	60S RIBOSOMAL PROTEIN L24 (Arabidopsis thaliana)	-0.2376
1772	COLLAGEN ALPHA 2(XI) CHAIN (Homo sapiens)	-0.2292
1024	ATP SYNTHASE A CHAIN (Trypanosoma brucei brucei)	-0.1356
1482	Human spermidine synthase gene, complete cds	0.1290
1641	Human enkephalin B (enkB) gene, exon 4 and 3' flank and complete cds	-0.1090

Table 5.5: Relative importance of the top 10 genes selected by RLogReg-F algorithm.

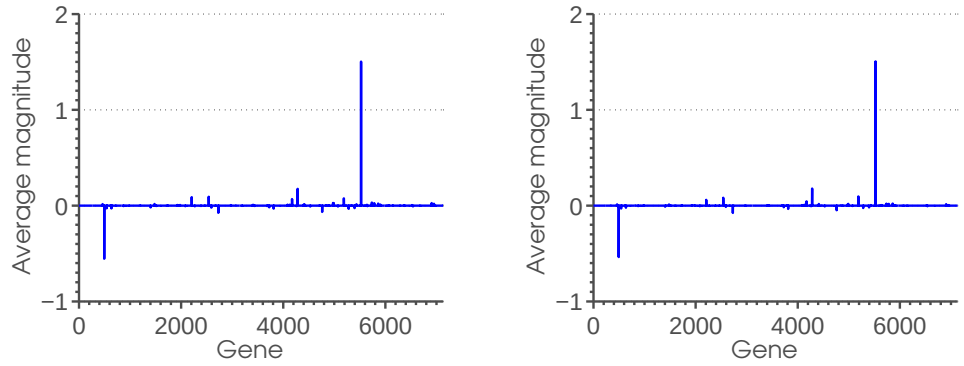
Gene No.	Gene Annotation	Avg. Magnitude
765	Human cysteine-rich protein (CRP) gene, exons 5 and 6	0.3988
377	H.sapiens mRNA for GCAP-II/uroguanylin precursor	0.3273
249	Human desmin gene, complete cds.	0.3201
1644	C4-DICARBOXYLATE TRANSPORT SENSOR PROTEIN DCTB (Rhizobium leguminosarum)	0.2883
1870	PEPTIDYL-PROLYL CIS-TRANS ISOMERASE, MITOCHONDRIAL PRECURSOR (HUMAN)	-0.2852
1346	60S RIBOSOMAL PROTEIN L24 (Arabidopsis thaliana)	-0.2420
1024	ATP SYNTHASE A CHAIN (Trypanosoma brucei brucei)	-0.2220
1993	Human hormone-sensitive lipase (LIPE) gene, complete cds	-0.2114
493	MYOSIN HEAVY CHAIN, NONMUSCLE (Gallus gallus)	0.1325
1772	COLLAGEN ALPHA 2(XI) CHAIN (Homo sapiens)	-0.1224

Results on *Breast* cancer dataset

We further apply the proposed model on the *Breast* cancer dataset from West et al. [2001]. The aim is to discriminate between estrogen positive and estrogen negative ob-

Table 5.6: LOO misclassification (%) on *Breast* dataset. The average number of selected genes (± standard deviation) was computed from the CLN runs.

Algorithm	LOO-CRT	LOO-CLN	# Genes
BLogReg	18.37 ± 0.79	2.50 ± 0.40	9.22 ± 0.58
RLogReg	18.37 ± 0.79	2.50 ± 0.40	9.10 ± 0.63
RLogReg-F	16.33 ± 0.76	0.00 ± 0.29	7.58 ± 0.50



(a) BLogReg (b) RLogReg-F

Figure 5.4: Comparison of the average weights of features selected by BLogReg and RLogReg-F on the *Breast* dataset over 1000 bootstrap repeats (39 train/10 test).

servations. According to West et al. [2001], there is biological evidence that the arrays 11,14,16,31,33,40,43,45,46 are mislabelled. However, unlike *Colon* dataset, we observe the nature of label flipping in *Breast* dataset is rather close to symmetric. As a consequence, mislabelling might do less harm to traditional classifiers in terms of class prediction on future arrays. Table 5.6 summarises LOO error rates together with the numbers of genes selected by the classifiers. The picture is quite similar to what we have seen in the case of *Colon*, although the differences tend to be smaller since the label noise here is more symmetric.

We also see that RLogReg did pretty well with a very limited amount of training data, but of course the difficulty of accurate estimation of the gamma table from such few points remains an issue. In fact, the estimated gamma table of RLogReg may converge to identity in such conditions, which statistically will result in a weight vector that is

Algorithm	LOO-CRT	LOO-CLN	# Genes
BLogReg	9.72 ± 0.41	8.45 ± 0.41	13.66 ± 0.84
RLogReg	9.72 ± 0.41	8.45 ± 0.41	13.21 ± 1.66
RLogReg-F	9.72 ± 0.41	8.45 ± 0.41	13.14 ± 0.76

Table 5.7: LOO misclassification (%) on *Leukaemia* dataset. The average number of selected genes (± standard deviation) was computed from the CLN runs.

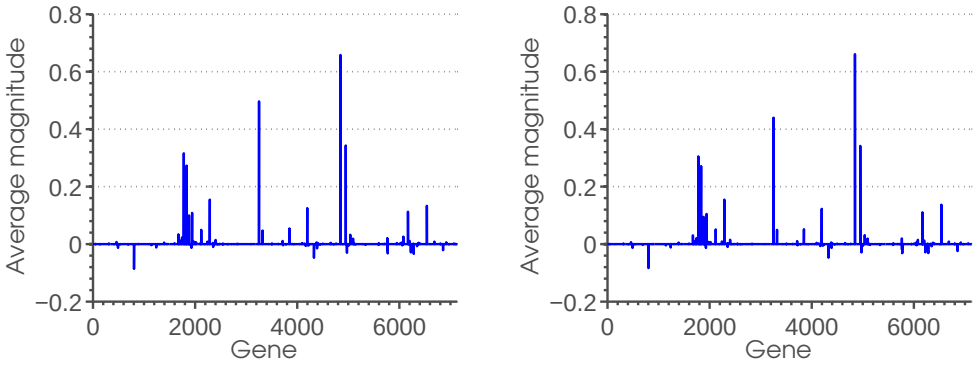
identical to that of BLogReg. As previously, knowledge of the extent of noise can be employed here, resulting in a slight improvement for RLogReg-F. Finally, as somewhat expected, the average magnitude of gene weights from BLogreg and RLogreg-F look very similar, as shown in Figure 5.4, which was expected by the symmetric nature of the label noise in this dataset.

Results on the *Leukaemia* dataset

In addition to the two real gene array datasets, we have also applied the proposed model to the *Leukaemia* dataset (Golub et al. [1999]). The aim is to classify acute myeloid leukaemia (AML) and acute lymphoblastic leukaemia (ALL).

The literature does not indicate any biological evidence that the dataset would contain mislabellings. However, by consensus of previous studies of labelling error detection algorithms, one gene array was identified as possibly mislabelled.

Hence, label noise modelling is not expected to improve classification performance here since the gamma table is so close to identity. We give in Table 5.7 the LOO cross validation error rates together with the number of features selected by the algorithms. We see that indeed all classifiers achieved the same LOO rates. Figure 5.5 shows the average frequency with which each gene is selected by BLogReg and RLogReg respectively, as computed from 1000 independent bootstrap repetitions – these also appear to be indistinguishable. This results in turn are comforting since they imply that the label noise modelling did not introduce any unwanted side effects when the data had in fact no label noise.



(a) BLogReg (b) RLogReg-F

Figure 5.5: Comparison of the average magnitude of features selected by BLogReg and RLogReg on the *Leukaemia* dataset over 1000 bootstraps (58 train/14 test). No artificial label noise was introduced in this experiment.

Computation time

We should give an indication of the added computation overhead required by our noise modelling relative to the existing BLogReg. One LOO loop on all datasets considered took on average 4 seconds for RLogReg, while BLogReg required roughly 0.2 seconds on an Intel's Core-i5 3.2 GHz machine. We believe this extra computation time is most worthwhile especially when the training set size is sufficiently large to exploit the full potential of the presented approach.

Detecting mislabelled instances

One of the most appealing features of our proposed algorithm is the possibility to detect mislabelled examples from the data, in addition to classification and gene selection. There are two types of possible errors: (i) a false positive is when a sample is believed to be mislabelled despite it is in fact labelled correctly; and (ii) a false negative is when a sample is believed to be labelled correctly despite its label is in fact incorrect. A good way to summarise both is by constructing the Receiver Operating Characteristic (ROC) curves. The area under the ROC curve signifies the probability that a randomly drawn

and mislabelled example would be flagged by the proposed algorithms. Figure 5.6 shows the ROC curves for *Synth-500* and *Colon* datasets. Superimposed for reference we also plotted the ROC curves that correspond to BLogReg. BLogReg considers that all points have the correct labels, and it has not been designed to spot mislabelled points. The best we can do is to take that mistakes made on the training points are mislabelling predictions. From Figure 5.6, we see the gap between the two curves is significant and well apparent in the experiment on *Synth-500*. This quantifies the gain that our modelling approach is able to obtain. The gain for *Colon* is smaller but still significant, despite the dataset size is so limited, provided that RLogReg incorporates knowledge about the proportion of mislabelling (i.e. RLogReg-F).

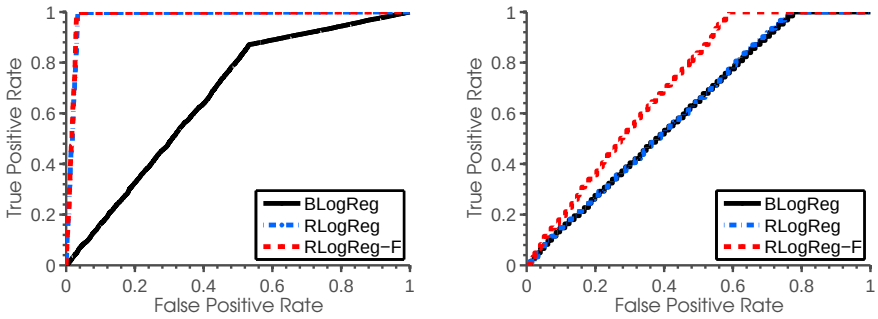
Comparison with previous findings

In addition to comparisons that quantify the benefits of having a noise model, we compare our results with previously identified mislabelling in the *Colon* samples. We conduct 100 bootstrap repetitions drawing subsets of size 50 from the total of 62 points randomly while imposing that none of the suspects from the literature are left out. In Table 5.8, after quoting the previous detections from the literature, we report the mislabelling detections obtained by BLogReg-F and BLogReg respectively, in two forms: (i) from the run that returned the largest number of detections, and (ii) the percentage that a particular array was flagged up as a mislabelling during the 100 repetitions.

It is interesting to note that RLogReg-F was able to identify up to 7 mislabelled points, and these also agree with the majority of previously reported detections using other algorithms (i.e. for T30, T33, T36, N34 and N36). BLogReg is also able to find up to 7 mislabelled samples but with fewer true positives and more false positives.

From both figures we see that RLogReg-F is able to identify mislabelled arrays more often than BLogReg can. Recall that BLogReg has no information about mislabellings, so for this method we considered a detection when a training point falls on the wrong

side of decision boundary. On the other hand, the mislabelling detections of RLogReg-F are based upon its explicit model of label flipping, and computed as in Section 5.3.



(a) *Synth-500* (b) *Colon*

Figure 5.6: Average ROC curves for BLogReg, RLogReg and RLogReg-F on *Synth-500* and *Colon* benchmarks. For consistency with classification result bootstrap is performed on *Synth-500* while LOO is employed to obtain the result for *Colon*. The prediction is based on hard-thresholded rule.

Table 5.8: Identifying mislabelled samples in *Colon* dataset. The detections for RLogReg-F are based on the hard threshold rule ($p(\tilde{y} \neq y|\mathbf{x}, \mathbf{w}) \geq 0.5$). The first line is the 'gold standard' that is backed up by biological evidence in the literature.

Source	Suspects identified									Extra samples identified
Alon et al. [1999]	T2	T30	T33	T36	T37	N8	N12	N34	N36	
Furey et al. [2000]	-	○	○	○	-	○	-	○	○	
Li et al. [2001]	-	○	○	○	-	-	-	○	○	
Kadota et al. [2003]	○	-	-	-	○	○	-	○	○	T6,N2
Malossini et al. [2006] (RAPIV)	-	○	○	○	○	○	-	○	○	N28,N29,N40
Malossini et al. [2006] (PRAPIV)	-	○	○	○	○	○	-	○	○	N2,N28
BLogReg	○	○	○	○	-	○	○	○	○	T3,T32,N35,N40
BLogReg (%)	1	9	14	63	0	9	18	32	37	
RLogReg-F	○	○	○	○	-	○	○	○	○	N2,T32,N40
RLogReg-F (%)	4	22	55	79	0	15	15	37	68	

5.6 Summary

We proposed a robust extension of sparse Bayesian logistic regression for classification in the presence of labelling errors. The numerical experiments suggest that our approach is superior to its traditional counterpart when the training data contains labelling errors. Simultaneously, our methods are effective in identifying marker-genes and detecting mislabelled data. Since our robust model needs to estimate the label flipping probabilities together with the parameters of the classifier, it does require more training data to achieve its full potential. However, in our experience, RLogReg performs statistically no worse than BLogReg even when the training set sizes are small. The need for more data can also be relaxed by incorporating knowledge about the extent of label noise.

Even though, the family of robust logistic regression (RLogReg in this chapter, rLR and rmLR presented in Chapter 4) performs well under mislabelling, one limitation of these algorithms is that being a linear classifier in nature they cannot cope with nonlinear dataset. Kernelising the model is the straightforward direction to proceed. However that will also introduce a problem of selecting kernel's parameters. As we have pointed out, cross-validation is not the best solution to model selection problem in noisy label settings. In the next chapter we will thoroughly investigate how to perform model selection under label noise. Specifically, we shall study kernel parameter and regularisation parameter selection for our robust Kernel Logistic Regression model.

CHAPTER 6

Robust Kernel Logistic Regression

In the previous two chapters we saw that the robust Logistic Regression is effective against labelling errors. However the model is restricted to linear problems. In real world, of course, it cannot be guaranteed that all classification problems can be solved using a linear model. Therefore, it is desirable to extend our approaches to non-linear settings, which we will do in this chapter.

Since the introduction of the kernel trick, many linear classifiers have been harnessed with an ability to solve non-linear problems, whereby their usage extends to a wider range of applications. Generally, deploying a kernel machine also involves determining good kernel parameters, and Cross-Validation (CV) has long been an established standard approach. However, when class label noise is present, it becomes unclear why CV would be a good approach, since all candidate models will be validated against noisy class labels. The issue has also been briefly discussed in Lawrence and Schölkopf [2001] and Bouveyron and Girard [2009]. In Lawrence and Schölkopf [2001], the authors resort to using a 'trusted validation set' to select optimal kernel parameters. The trusted set must be labelled carefully, which seriously restricts the applicability of the method. For example in crowdsourcing it would be very difficult (if not impossible) to construct such a trusted

set.

In this chapter we start by straightforwardly formulating a robust Kernel Logistic Regression (rKLR) as an extension of our robust Logistic Regression (rLR). We present a simple yet effective algorithm to learn the classifier and investigate whether or not CV is a reasonable approach for model selection in the presence of labelling errors. As we shall see, we find that performing CV in noisy environments gives rise to a slightly under-fitted model. We then propose a robust Multiple Kernel Logistic Regression algorithm (rMKLR) based on the so-called Multiple Kernel Learning (MKL) framework and the Bayesian regularisation technique (Cawley and Talbot [2007]) to automate the model selection step without using any cross validation. From this we obtain improvements in both generalisation performance and learning speed. The lineage of the robust Logistic Regression family, which serves as a roadmap of the developments in this chapter, is summarised in Figure 6.1.

6.1 The robust kernel machine

Consider a set of training samples $S = \{(\mathbf{x}_n, \tilde{y}_n)\}_{n=1}^N$, where $\mathbf{x}_n \in \mathbb{R}^M$ and $\tilde{y}_n \in \{0, 1\}$ denotes the observed (possibly noisy) label of $\mathbf{x}_n$. Kernel logistic regression produces a nonlinear decision boundary, $f(\mathbf{x})$, by forming a linear decision boundary in the space of the non-linearly transformed input vectors. By the representer theorem (Kimeldorf and Wahba [1971]), the optimal $f(\mathbf{x})$ has the form:

$$f(\mathbf{x}) = \sum_{n=1}^{N} w_n \kappa(\cdot, \mathbf{x}_n) \tag{1}$$

where $\kappa(\cdot, \cdot)$ is a positive definite reproducing kernel that gives an inner product in the transformed space.

Denoting by $\mathbf{w}$ the parameter vector with entries $w_n, n = 1, ..., N$, we define the

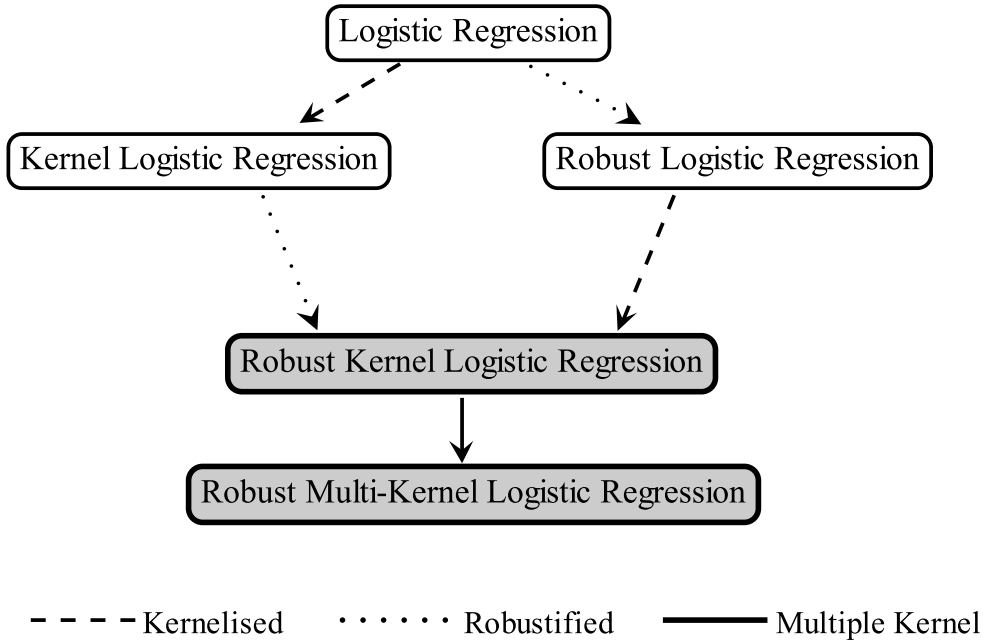


Figure 6.1: Genealogy of the robust Kernel Logistic Regression and the robust Multi-Kernel Logistic Regression methods. The highlighted boxes are the classifiers proposed in this chapter. Note that there are two paths to arrive at the robust Kernel Logistic Regression.

probability of an observed label $\tilde{y}_n$ as a linear combination of the probabilities that the true label of a point is 0 or 1 respectively:

$$\begin{aligned} p(\tilde{y} = k|\kappa(\cdot, \mathbf{x}_n), \mathbf{w}) &= \sum_{j=0}^{1} p(\tilde{y} = k|y = j)p(y = j|\kappa(\cdot, \mathbf{x}_n), \mathbf{w}) \\ &= \sum_{j=0}^{1} \gamma_{jk} p(y = j|\kappa(\cdot, \mathbf{x}_n), \mathbf{w}) \end{aligned} \tag{2}$$

Again, $p(\tilde{y} = k|y = j)$ are probabilistic factors representing the probability that the true label j flips into the observed label k. These parameters form a label transition table, Γ. The full set of parameters for this robust model will be denoted as $\Theta = \{\mathbf{w}, \Gamma\}$. Now, fitting the robust kernel logistic regression is equivalent to maximising the following

log-likelihood:

$$\mathcal{L}(\Theta) = \sum_{n=1}^{N} \sum_{k=0}^{1} \mathbb{1}(\tilde{y}_n = k) \log p(\tilde{y}_n = k | \kappa(\cdot, \mathbf{x}_n), \Theta) - \zeta \sum_{n=1}^{N} w_n^2 \tag{3}$$

where we included an $L2$ regularisation term to express our preference for a smooth (and non-sparse) model.

In Eq.(3), the term $p(\tilde{y}_n = k | \kappa(\cdot, \mathbf{x}_n), \Theta)$ is defined in Eq.(2), in which we use a sigmoid function to model the probability of the true label:

$$p(y = 1 | \kappa(\cdot, \mathbf{x}_n), \mathbf{w}) = \sigma(\mathbf{w}^T \kappa(\cdot, \mathbf{x}_n)) = \frac{1}{1 + \exp(-\mathbf{w}^T \kappa(\cdot, \mathbf{x}_n))} \tag{4}$$

Learning the robust model requires us to estimate the weight vector $\mathbf{w}$ as well as the label-flipping probabilities γ_{jk}. To optimise the weight vector, we again employ conjugate gradients because of its well known computational efficiency, The optimisation technique basically performs the Newton update step along the direction $\mathbf{u} = \mathbf{g} - \mathbf{u}^{old}\nu$, where $\mathbf{g}$ is the gradient of the log-likelihood w.r.t. the weight vector.

Define $\tilde{P}_n^k = p(\tilde{y} = k | \kappa(\cdot, \mathbf{x}_n), \Theta)$, the gradient is given by:

$$\begin{aligned} \nabla_{\mathbf{w}} \mathcal{L}(\theta) = \sum_{n=1}^{N} \Bigg[\Bigg(& \frac{\mathbb{1}(\tilde{y}_n = 1)(\gamma_{11} - \gamma_{01})}{\tilde{P}_n^1} + \frac{\mathbb{1}(\tilde{y}_n = 0)(\gamma_{10} - \gamma_{00})}{\tilde{P}_n^0} \Bigg) \\ & \times \sigma(\mathbf{w}^T \kappa(\cdot, \mathbf{x}_n))(1 - \sigma(\mathbf{w}^T \kappa(\cdot, \mathbf{x}_n))) \times \kappa(\cdot, \mathbf{x}_n) \Bigg] - 2\zeta \sum_{n=1}^{N} w_n \end{aligned} \tag{5}$$

The Hestenes-Stiefel formula, $\nu = \mathbf{g}^T(\mathbf{g} - \mathbf{g}^{old}) / (\mathbf{u}^{old})^T(\mathbf{g} - \mathbf{g}^{old})$ is used to calculate the step length. The update equation for $\mathbf{w}$ is then the following:

$$\mathbf{w}^{new} = \mathbf{w}^{old} - \frac{\mathbf{g}^T \mathbf{u}}{\mathbf{u}^T \mathbf{H} \mathbf{u}} \mathbf{u}, \tag{6}$$

where the Hessian matrix $\mathbf{H}$ is calculated only once at the first iteration. We should note

that other schemes such as the Flectcher-Reeves or Polak-Ribère formulae could also be used.

The following multiplicative update equations are then used to update the elements of the gamma matrix. These are derived as in Chapter 4 using the method of Lagrangian multipliers to ensure that the row of the gamma table sums to 1.

$$\gamma_{00} = \frac{\gamma_{00} \sum_{n=1}^{N} \left[\frac{\mathbb{1}(\tilde{y}_n=0)}{\tilde{P}_n^0} (1 - \sigma(\mathbf{w}^T \kappa(\cdot, \mathbf{x}_n)) \right]}{\gamma_{00} \sum_{n=1}^{N} \left[\frac{\mathbb{1}(\tilde{y}_n=0)}{\tilde{P}_n^0} (1 - \sigma(\mathbf{w}^T \kappa(\cdot, \mathbf{x}_n)) \right] + \gamma_{01} \sum_{n=1}^{N} \left[\frac{\mathbb{1}(\tilde{y}_n=1)}{\tilde{P}_n^1} (1 - \sigma(\mathbf{w}^T \kappa(\cdot, \mathbf{x}_n)) \right]} \tag{7}$$

$$\gamma_{11} = \frac{\gamma_{11} \sum_{n=1}^{N} \left[\frac{\mathbb{1}(\tilde{y}_n=1)}{\tilde{P}_n^1} \sigma(\mathbf{w}^T \kappa(\cdot, \mathbf{x}_n)) \right]}{\gamma_{10} \sum_{n=1}^{N} \left[\frac{\mathbb{1}(\tilde{y}_n=0)}{\tilde{P}_n^0} \sigma(\mathbf{w}^T \kappa(\cdot, \mathbf{x}_n)) \right] + \gamma_{11} \sum_{n=1}^{N} \left[\frac{\mathbb{1}(\tilde{y}_n=1)}{\tilde{P}_n^1} \sigma(\mathbf{w}^T \kappa(\cdot, \mathbf{x}_n)) \right]} \tag{8}$$

With all the ingredients in place, the learning algorithm is then to alternate between updating each parameter in turn, until convergence. Given an unseen query point $\mathbf{x}_q$, we predict that $\hat{y}_q = 1$ whenever $p(\hat{y} = 1 | \kappa(\cdot, \mathbf{x}_q), \mathbf{w}) = \sigma(\mathbf{w}^T \kappa(\cdot, \mathbf{x}_q))$ returns a value greater than 0.5, and $\hat{y} = 0$ otherwise. This algorithm to efficiently learn rKLR is summarised below in Algorithm 6.

Algorithm 6 Optimisation of rKLR

Input: κ, Γ
  Initialise $\mathbf{w} \leftarrow 0$
  while Iteration < MaxIteration **do**
    Update $\mathbf{w}$ using the gradient in Eq.(5)
    Update Γ using Eq.(7) and Eq.(8)
  end while
Output: Optimised weight vector, $\mathbf{w}$. Optimised Γ.

6.1.1 Selecting the kernel width: A multi-kernel approach

For any kernel machine, the value of the kernel parameters are critical to the generalisation performance, and determining these is an important part of the task. Here we focus on radial kernels for the sake of concreteness. In this case the kernel parameter is the width of the kernel. A usual way of finding optimal kernel width is by means of cross-validation. However this technique makes use of class labels of a validation set. It is unclear if this

would be useful in the noisy labels scenario since the class labels of the validation set are possibly incorrect.

We propose to employ the Multiple Kernel Learning (MKL) framework [1], giving it a different purpose. In MKL a combination of several kernels is learnt in order to get a good representation of the data. We adopt the framework as a method to find optimal kernel width automatically without performing cross-validation. In contrast to the majority of MKL literature where the aim is centred around combining heterogeneous data sources (Pavlidis et al. [2001], Lanckriet et al. [2004]), our adoption of MKL focuses on the combination of multiple kernels that correspond to different notions of similarity, as defined by different kernel widths. This approach will bypass the need for cross-validation and as a by-product of this it also speeds up the learning process.

There are several ways to combine kernels. We will use a conic combination, as the following:

$$\kappa(\cdot,\cdot) = \sum_{i=1}^{S} \eta_i \kappa_i(\cdot,\cdot) : \qquad \eta_i \geq 0 : \forall i \tag{9}$$

Conic combinations represent a popular way to combine kernels. It is less constrained than a convex combination would be, and the positivity constraint ensures that the kernel weighting parameters do not cancel out each other. The latter is important since linear combinations may lead to unstable learning (Damoulas and Girolami [2009]). A convex combination could also be used but it would require the extra constraint that η_i sums to unity, which is unnecessary.

In contrast to the case where one is concerned with heterogeneous data sources, we want $\boldsymbol{\eta}$ to be sparse for our purpose, in order to select only a few of a set of possible kernel widths. To implement this idea we use a generalised LASSO-like approach, positing independent exponential priors to enforce this preference on $\boldsymbol{\eta}$. This results in adding a

[1] an extensive survey in recent advances of MKL is given in Gönen and Alpaydin [2011]

new regulariser to the objective in Eq.(3) to accommodate the MKL framework:

$$\sum_{n=1}^{N} \mathbb{1}(\tilde{y}_n = 1) \log \tilde{P}_n^1 + \mathbb{1}(\tilde{y}_n = 0) \log \tilde{P}_n^0 - \zeta \sum_{i=1}^{N} w_i^2 - \sum_{i=1}^{S} \xi_i \eta_i \tag{10}$$

To ensure positivity, we reparametrise $\eta_i = u_i^2$, and optimise for u_i using conjugate gradients method. The derivative of the objective, Eq.(10), w.r.t. u_i is given by:

$$\sum_{n=1}^{N} \Bigg[\left(\frac{\mathbb{1}(\tilde{y}_n = 1)(\gamma_{11} - \gamma_{01})}{\tilde{P}_n^1} + \frac{\mathbb{1}(\tilde{y}_n = 0)(\gamma_{10} - \gamma_{00})}{\tilde{P}_n^0} \right) \\ \times \sigma(\mathbf{w}^T \kappa(\cdot, \mathbf{x}_n))(1 - \sigma(\mathbf{w}^T \kappa(\cdot, \mathbf{x}_n))) \times (\mathbf{w}^T \kappa_i(\cdot, \mathbf{x}_n)) \Bigg] - 2\xi_i u_i \tag{11}$$

We later recover η_i by squaring the optimised u_i.

6.1.2 Choosing the regularisation parameters by Bayesian regularisation

As discussed earlier, everything that involves cross validation is questionable in the presence of labelling errors. This includes the selection of the regularisation hyper-parameters. To circumvent the problem, we adopt a Bayesian regularisation technique to automatically determine good values of ζ and $\boldsymbol{\xi} := (\xi_1, ..., \xi_S)$. For this, we consider a Bayesian interpretation of Eq.(10).

Consider the terms that depend on the parameter $\mathbf{w}$ and ζ first. The posterior probability of $\mathbf{w}$ can be expressed as:

$$p(\mathbf{w}|\mathcal{D}, \zeta) \propto p(\mathcal{D}|\mathbf{w}, \boldsymbol{\xi}) p(\mathbf{w}|\zeta) \tag{12}$$

The first term on the r.h.s. corresponds to the data likelihood while the second term is our regularisation term for $\mathbf{w}$. By taking logarithm on both sides of Eq.(12), $\log p(\mathbf{w}|\mathcal{D}, \zeta) =$

$\log p(\mathcal{D}|\mathbf{w}, \boldsymbol{\xi}) + \log p(\mathbf{w}|\zeta) + const.$, we see that the regularisation term is simply the negative logarithm of the prior distribution conditioned on ζ, the regularisation parameter. Therefore, $p(\mathbf{w}|\zeta) = \mathcal{N}(0, 1/\zeta)$.

We want to eliminate ζ from the formulation, so we build the model further by putting a prior on ζ. We choose this to be an exponential distribution because the values of ζ must be positive: $p(\zeta|\beta) = \beta e^{-\beta\zeta}$. Here, β is a hyper-parameter, i.e. the inverse scale of the exponential. This encodes our uncertainty about ζ, and as such, it reflects our uncertainty about $\mathbf{w}$ at a higher level of inference. We used $\beta = 2$ in in the reported experiments to constrain the expected prior variance of $\mathbf{w}$.

With this hyper-prior in place, we can write the marginal prior distribution, $p(\mathbf{w})$ by integrating out ζ:

$$p(\mathbf{w}) = \int_0^\infty p(\mathbf{w}|\zeta)p(\zeta)\mathtt{d}\zeta \tag{13}$$

Completing the integration by the use of the Gamma integral $\int_0^\infty x^{\nu-1}e^{-\mu x}\mathtt{d}x = \frac{\Gamma(\nu)}{\mu^\nu}$, we obtain:

$$\begin{aligned} p(\mathbf{w}) &= \int_0^\infty \prod_{i=1}^M \left\{ \sqrt{\frac{\zeta}{2\pi}} e^{-\frac{\zeta}{2}w_i^2} \right\} \cdot \beta e^{-\beta\zeta}\mathtt{d}\zeta \\ &= \frac{\beta}{(2\pi)^{m/2}} \int_0^\infty \zeta^{(m/2+1)-1} e^{-\zeta(\frac{1}{2}\sum_{i=1}^M w_i^2 + \beta)}\mathtt{d}\zeta \\ &= \frac{\beta}{(2\pi)^{m/2}} \frac{\Gamma(\frac{m}{2}+1)}{(\frac{1}{2}\sum_{i=1}^M w_i^2 + \beta)^{(m/2+1)}} \end{aligned} \tag{14}$$

Going back to our objective function in Eq.(10), we now replace $\mathbf{w}$'s the regularisation term with the negative log of the newly derived marginal prior, and optimise this objective w.r.t. $\mathbf{w}$. Computing the gradient of this new regularisation term yields:

$$-\frac{\partial \log p(\mathbf{w})}{\partial \mathbf{w}} = \frac{\frac{m}{2}+1}{\frac{1}{2}\sum_{i=1}^M w_i^2 + \beta} \frac{\partial \sum_{i=1}^M w_i^2}{\partial \mathbf{w}} \tag{15}$$

and since this has the same form as the gradient of the original regularisation term would, we read off from Eq.(15) the regularisation parameter as,

$$\zeta = \frac{\frac{m}{2} + 1}{\frac{1}{2}\sum_{i=1}^{M} w_i^2 + \beta} \tag{16}$$

Next we proceed to treat ξ_i using the same technique of Bayesian regularisation. This time we are looking for ξ_i that produces a sparse $\boldsymbol{\eta}$, in order to select just a very few kernel widths. We will employ a regularisation on each component of $\boldsymbol{\eta}$. The Bayesian interpretation of Eq.(10) with respect to $\boldsymbol{\eta}$ and $\boldsymbol{\xi}$ is given by,

$$p(\boldsymbol{\eta}|\mathcal{D}, \mathbf{w}, \boldsymbol{\xi}) \propto p(\mathcal{D}|\boldsymbol{\eta}) \prod_{i=1}^{S} p(\eta_i|\xi_i) \tag{17}$$

where we employed independent priors distributions on each η_i. Recall that we constrained η_i to be non-negative, so a natural choice is to use independent exponential distributions $p(\eta_i|\xi_i) = \xi_i e^{-\xi_i \eta_i}$, and ξ_i denote the inverse scale parameters of these. These are hyperparameters that correspond to the regularisation parameters in the last term of our objective function Eq.(10).

Again, we want to integrate out the ξ_i from the formulation, so we build this model further, positing a hyper-prior on all ξ_i. These also need to be positive, hence we use the exponential distribution one more, $p(\xi_i) = \psi e^{-\psi \xi_i}$, and set $\psi = 10^{-100}$ to a non-informative hyperprior that will encourage a sparse solution. We obtain the marginal prior by integration, which gives:

$$\begin{aligned} p(\eta_i) &= \int_0^\infty \xi_i e^{-\xi_i \eta_i} \cdot \psi e^{-\psi \xi_i} \mathtt{d}\xi_i \\ &= \psi \int_0^\infty \xi_i^{(1+1)-1} e^{-\xi_i(\eta_i + \psi)} \mathtt{d}\xi_i = \psi \frac{\Gamma(2)}{(\eta_i + \psi)^2} \end{aligned} \tag{18}$$

Finally, replacing the negative log of this in place of our original regularisation term

in Eq.(10), re-parametrising $u_i = \sqrt{\eta_i}$ and taking derivative of the log of Eq.(18) w.r.t. u_i we have,

$$-\frac{\partial \log p(\eta_i)}{\partial u_i} = \frac{2}{(\eta_i + \psi)} \frac{\partial \eta_i}{\partial u_i} \tag{19}$$

From here we read off that

$$\xi_i = \frac{2}{\eta_i + \psi} \tag{20}$$

Algorithm 7 summarises the steps to learn our novel "robust Multiple Kernel Logistic Regression" (rMKLR) model.

Algorithm 7 Optimisation of rMKLR

Input: Set of predefined kernels $\kappa_{i=1:S}$, Γ
  Initialise $\mathbf{w} \leftarrow 0$, $\boldsymbol{\eta} \leftarrow 1$, $\zeta \leftarrow 0$, $\boldsymbol{\xi} \leftarrow 0$
  while Iteration $<$ MaxIteration **do**
    Update $\mathbf{w}$ using Eq.(5)
    Update ζ using Eq.(16)
    Update η_i by optimising u_i using Eq.(11) and set $\eta_i = u_i^2$
    Update $\boldsymbol{\xi}$ using Eq.(20)
    Update Γ using Eq.(7) and Eq.(8)
  end while
Output: Optimised weight vector, $\mathbf{w}$. Optimised Γ.

6.2 Empirical evaluation

We conducted extensive experiments to answer three main research questions.

- Firstly, we ask if rKLR improves KLR in terms of robustness against labelling errors as measured via classification performance. To answer this question, we also study the relative harm of two common types of label noise: symmetric and asymmetric noise in non-linear problems.

- Secondly, we ask if MKL can be used to find a suitable kernel parameter in noisy settings. To answer the second research question we first show that our proposed MKL for kernel width selection works in a noise-free scenario. We then progress

to show the comparative performance of rKLR where its kernel width was selected using 1) CV with a trusted validation set, 2) CV without a trusted validation set and 3) MKL framework in a noisy setup.

- Thirdly, we ask how the proposed rMKLR compares to robust Kernel Fisher Discriminant (rKFD) (Lawrence and Schölkopf [2001]), to the gold-standard Support Vector Machine (SVM) and to the Stochastic Programming for Multiple Kernel Learning (StPMKL) (Yang et al. [2012]). The recently proposed StPMKL is designed to learn from noisy labels by relaxing a deterministic constraint in MKL into a chance constraint using a binary random variable for each example that indicates if the class assignment of the example is correct. It has been shown to outperform state-of-the-art MKL algorithms in noisy setups [1] and MKL formulation of the robust SVM (Xu et al. [2006]), and hence can be regarded as one of the best MKL algorithms for noisy labels. SVM is an established classifier which incorporates slack variables, hence it should be robust to label noise to some extent [2]. The rKFD was included in this comparison because it has been previously found to be effective in a wide range of noisy non-linear problems (Li et al. [2007]). It is a generative classifier as opposed to our rMKLR – which is a discriminative classifier – and it is interesting to compare their performance.

6.2.1 Experimental protocol

In answering these questions, we train the proposed classifier on data where label noise is created artificially so as to gain better understanding of its effects. We train the model on the corrupted dataset and evaluate the learnt model using clean test sets. We consider label noise contamination ranging from 10% up to 40%. We should note that label noise over 40% is very unlikely to occur in practice as it would mean a very poor labelling close

[1] StPMKL was compared to Simple MKL (Xu et al. [2010])
[2] We use LIBSVM (Chang and Lin [2011]) in our reported experiments

Data set	Training samples	Test samples	Pos. samples	Neg. samples	Dimensionality
Banana	400	4900	44.83%	55.17%	2
B.Cancer	200	77	29.28%	70.72%	9
Diabetes	468	300	34.90%	65.10%	8
German	700	300	30.00%	70.00%	20
Heart	170	100	44.44%	55.56%	13
Image	1300	1010	56.95%	43.05%	18
Ringnorm	400	7000	49.51%	50.49%	20
S.Flare	666	400	65.28%	34.72%	9
Splice	1000	2175	44.93%	55.07%	60
Thyroid	140	75	30.23%	69.77%	5
Titanic	150	2051	58.33%	41.67%	3
Twonorm	400	7000	50.04%	49.96%	20
Waveform	400	4600	32.94%	67.06%	21

Table 6.1: Characteristics of the non-linear benchmark datasets.

to random class assignment.

We use 13 UCI benchmark datasets (Rätsch et al. [2001]) in our controlled experiments. Each problem has been split into 100 train/test realisations except the *Image* and *Splice* datasets where 20 realisations are provided. The characteristics of the datasets used are summarised in Table 6.1. We later use crowdsourcing and cheaply annotated datasets to demonstrate real applications of the algorithm in learning from unreliable data sources.

For experiments where cross-validation (CV) is needed, we adopt the 'best practice' method described in Cawley and Talbot [2010], where model selection is seen as part of the learning, and that 5-folds CV is performed independently on each split of the data. When needed, we set aside 10% of the training data as a trusted validation set, in which all labels are perfect. We employed a Gaussian Radial Basis Function (RBF) kernel defined as,

$$\kappa(\mathbf{x}, \mathbf{x}_n) = \exp\left(-\frac{||\mathbf{x} - \mathbf{x}_n||^2}{\sigma}\right) \tag{21}$$

in all of our experiments except in the textual entailment recognition task where we used a linear kernel. For MKL, our multiple kernels set is composed of 21 RBF base kernels with widths σ in the set $\{2^{-10}, 2^{-9}, \ldots, 2^{10}\}$. This set has a comprehensive coverage of the range of possible values and we found this level of granularity works well in practice. An assessment of the sensitivity to this choice will be made in a later section. We also

use this set of parameter values in the CV experiments, for searches for both the kernel width and the C parameter in the case of SVM.

6.2.2 KLR versus rKLR

Symmetric versus Asymmetric noise

We start with an illustrative experiment, in which we are interested in finding out which kind of random noise is more detrimental to the traditional kernel learning. We have seen that asymmetric noise is more harmful in linear cases. Here we shall find out if the phenomenon can still be observed in non-linear cases. We train traditional KLR on data with 30% symmetric and 30% asymmetric noise and present the average classification errors and standard deviations over 100 repetitions in Figure 6.2. Taking the performance on clean data to be the baseline result, we see that in 5 datasets symmetric label noise is more detrimental while for the rest of the datasets asymmetric noise perturbs the classifier more. Hence, as we see, even symmetric label noise is not always harmless. It is worth noting also that the effect of label noise is more significant in artificial data that has a low Bayes error (e.g., *Ringnorm* and *Twonorm*) than real world dataset (e.g., *S.Flare*) that might possibly already have some inherent label noise.

The advantage of modelling the label noise

Having seen that traditional KLR is not suitable for learning from data with noisy labels, we are now interested to see if incorporating a label noise model helps to improve classification performance. To this end, we compare KLR to the proposed rKLR. To eliminate any other factors which could affect the assessment, we defer the use of MKL with Bayesian regularisation (hereinafter referred to as "the MKL") until the next subsection, and instead here we use CV with a trusted validation set containing correct labels to select the kernel width parameter for both algorithms. Since we have seen that both types of label noise are detrimental to the classifiers, we performed this test for both types

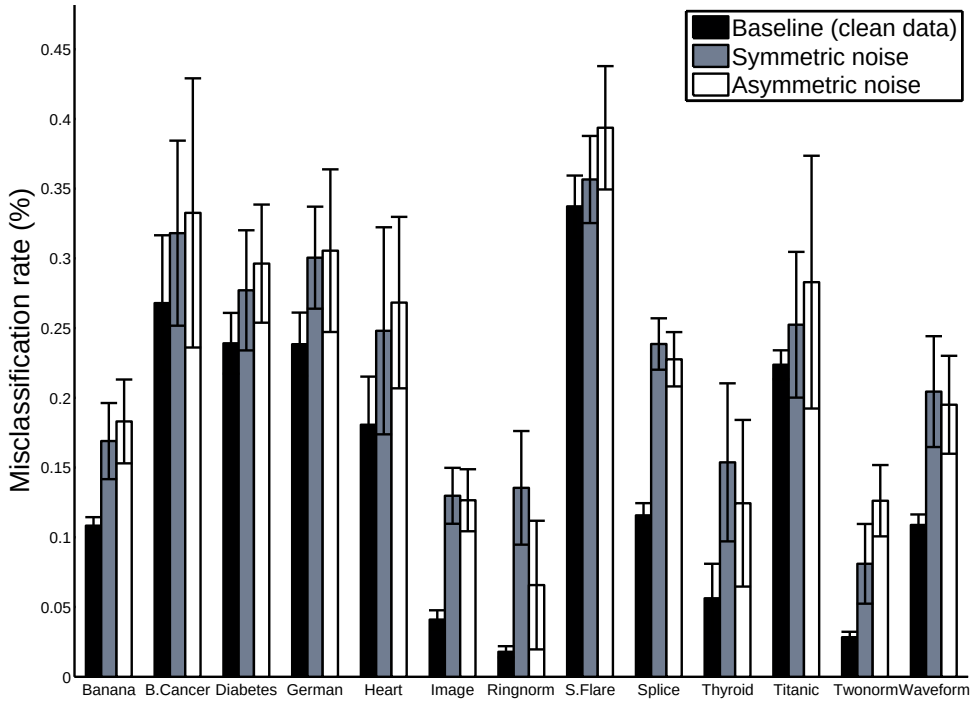


Figure 6.2: Effect of 30% symmetric and asymmetric noise to traditional KLR, compared against clean baseline.

of noise, and pulled together the misclassification rates computed from 100 independent splits of the data contaminated with symmetric noise as well as the 100 splits of data contaminated with asymmetric noise. Table 6.2 summarises the average misclassification rates, standard deviations and p-values.

We see that KLR is quite robust at low noise case, i.e. 10%. However, as level of noise increases we see that rKLR substantially improves upon KLR in most of the data sets used. We observe also that as the degree of mislabelling becomes more severe, the performance gaps are getting larger. This can be seen in the 30%-40% noise settings. The improvements are also statistically significant as tested using the Wilcoxon ranksum test at the 5% level. We may conclude on the basis of these results in Table 6.2 that label

Dataset	Noise level					
	10%			20%		
	KLR	rKLR	p-value	KLR	rKLR	p-value
Banana	**12.28 ± 1.49**	12.80 ± 2.10	0.03	14.58 ± 1.96	**12.91 ± 2.32**	$3.21e-22$
B.Cancer	**28.96 ± 5.62**	31.02 ± 6.20	$6.51e-4$	**30.19 ± 7.04**	32.16 ± 7.82	0.02
Diabetes	**24.86 ± 3.09**	26.00 ± 3.48	$3.05e-4$	26.17 ± 3.17	26.69 ± 4.28	0.79
German	**25.07 ± 3.23**	26.90 ± 3.46	$7.62e-8$	**26.64 ± 3.58**	27.78 ± 3.66	$3.27e-3$
Heart	19.84 ± 6.57	20.34 ± 5.73	0.17	22.31 ± 5.10	**20.63 ± 4.76**	$9.91e-4$
Image	6.58 ± 1.13	6.19 ± 1.52	0.12	8.06 ± 1.12	**6.80 ± 1.00**	$2.79e-6$
Ringnorm	4.51 ± 2.23	**3.11 ± 1.78**	$5.91e-13$	4.94 ± 3.23	**3.29 ± 1.86**	$1.87e-6$
S.Flare	34.45 ± 2.33	34.81 ± 2.83	0.24	35.87 ± 2.70	36.00 ± 2.92	0.96
Splice	14.90 ± 1.47	14.90 ± 1.69	0.89	17.33 ± 1.67	16.82 ± 1.63	0.14
Thyroid	9.25 ± 4.00	**7.76 ± 4.28**	$6.194e-5$	9.56 ± 4.55	**8.49 ± 4.60**	$9.86e-3$
Titanic	22.85 ± 1.33	22.88 ± 1.90	0.63	23.57 ± 2.55	23.30 ± 2.39	0.16
Twonorm	4.69 ± 1.16	**3.79 ± 0.78**	$1.51e-19$	7.82 ± 1.88	**4.38 ± 1.20**	$5.33e-54$
Waveform	12.91 ± 1.52	**12.21 ± 1.15**	$2.22e-6$	15.04 ± 2.18	**12.81 ± 1.54**	$1.05e-33$
Dataset	Noise level					
	30%			40%		
	KLR	rKLR	p-value	KLR	rKLR	p-value
Banana	17.60 ± 2.95	**16.13 ± 3.99**	$7.38e-7$	25.63 ± 6.01	**23.08 ± 10.49**	$1.85e-5$
B.Cancer	32.54 ± 8.29	32.92 ± 9.26	0.87	36.89 ± 10.39	35.52 ± 10.36	0.15
Diabetes	28.67 ± 4.37	**27.42 ± 4.56**	$2.60e-492$	33.77 ± 6.19	**31.14 ± 7.03**	$7.70e-6$
German	30.30 ± 4.86	**28.81 ± 4.69**	$9.78e-4$	33.86 ± 8.42	**30.11 ± 4.69**	$2.19e-4$
Heart	25.82 ± 6.87	26.64 ± 8.15	0.62	34.99 ± 8.76	**30.98 ± 11.65**	$2.34e-5$
Image	12.82 ± 2.10	**10.45 ± 3.21**	$1.20e-3$	20.29 ± 3.78	**15.98 ± 7.24**	$8.48e-3$
Ringnorm	10.06 ± 5.57	9.57 ± 6.00	0.43	16.92 ± 8.78	15.78 ± 9.67	0.11
S.Flare	37.51 ± 4.25	36.82 ± 3.63	0.13	41.04 ± 4.71	**38.61 ± 4.37**	$1.50e-7$
Splice	23.31 ± 1.95	**21.20 ± 4.06**	0.03	31.20 ± 3.85	**26.74 ± 8.49**	0.04
Thyroid	13.91 ± 5.99	13.76 ± 8.10	0.24	22.44 ± 11.01	**19.16 ± 13.41**	$8.82e-5$
Titanic	26.77 ± 7.54	**25.19 ± 5.55**	0.03	34.64 ± 12.49	**29.47 ± 11.39**	$4.18e-9$
Twonorm	10.36 ± 3.53	**9.60 ± 6.54**	$1.17e-4$	22.00 ± 6.16	**17.14 ± 13.42**	$6.19e-5$
Waveform	19.97 ± 3.77	**17.37 ± 5.24**	$4.11e-10$	27.50 ± 5.81	**22.86 ± 9.86**	$3.74e-8$

Table 6.2: The relative performance of KLR and the proposed rKLR. Average errors, standard deviations, and p-values.

noise modelling is indeed advantageous in general.

6.2.3 Cross Validation versus MKL with Bayesian regularisation

Results on clean data

In the literature MKL has been used to combine different sources of data. However, here we adopted the MKL framework to automatically determine a good kernel width. Before proceeding to our noisy scenarios, we will first establish that the MKL works on clean data. We present in Table 6.3 the comparative classification results between CV and the MKL on the clean benchmark datasets.

Data set	Cross validated rKLR	rMKLR	p-value
Banana	10.96 ± 0.81	10.72 ± 0.52	0.06
B.Cancer	29.94 ± 4.65	**27.73 ± 4.19**	$9.17e-4$
Diabetes	24.53 ± 2.21	24.24 ± 1.85	0.47
German	25.38 ± 2.63	**23.52 ± 2.26**	$2.09e-7$
Heart	18.63 ± 4.00	**16.30 ± 3.39**	$7.48e-5$
Image	**3.73 ± 0.71**	5.65 ± 0.96	$2.78e-6$
Ringnorm	1.80 ± 0.43	**1.48 ± 0.10**	$4.16e-12$
S.Flare	**33.50 ± 2.15**	34.33 ± 1.75	$1.05e-3$
Splice	**11.47 ± 0.82**	13.30 ± 1.13	$7.52e-6$
Thyroid	5.96 ± 2.78	5.91 ± 2.70	0.95
Titanic	**22.26 ± 0.94**	22.73 ± 0.83	$4.28e-5$
Twonorm	2.86 ± 0.36	**2.47 ± 0.16**	$4.58e-17$
Waveform	10.78 ± 0.85	**10.58 ± 0.45**	0.03

Table 6.3: Comparison between standard cross validation and MKL with Bayesian regularisation technique on clean datasets.

From Table 6.3, we see that the MKL is effective in choosing a good representation of the data, i.e. a good kernel width. We see that the difference between CV and Bayesian regularisation – albeit statistically significant in favour of the latter in 6 out of 13 cases – is small (mostly within <1%) in all cases. This clearly confirms that the MKL with Bayesian regularisation is a both effective and efficient way to automate the process of kernel width selection. It is worth noting that the results from these two robust classifiers are slightly worse than the traditional KLR (c.f. Cawley and Talbot [2008]) when the labels are actually perfect. This is due to the presumption of label noise of the robust model.

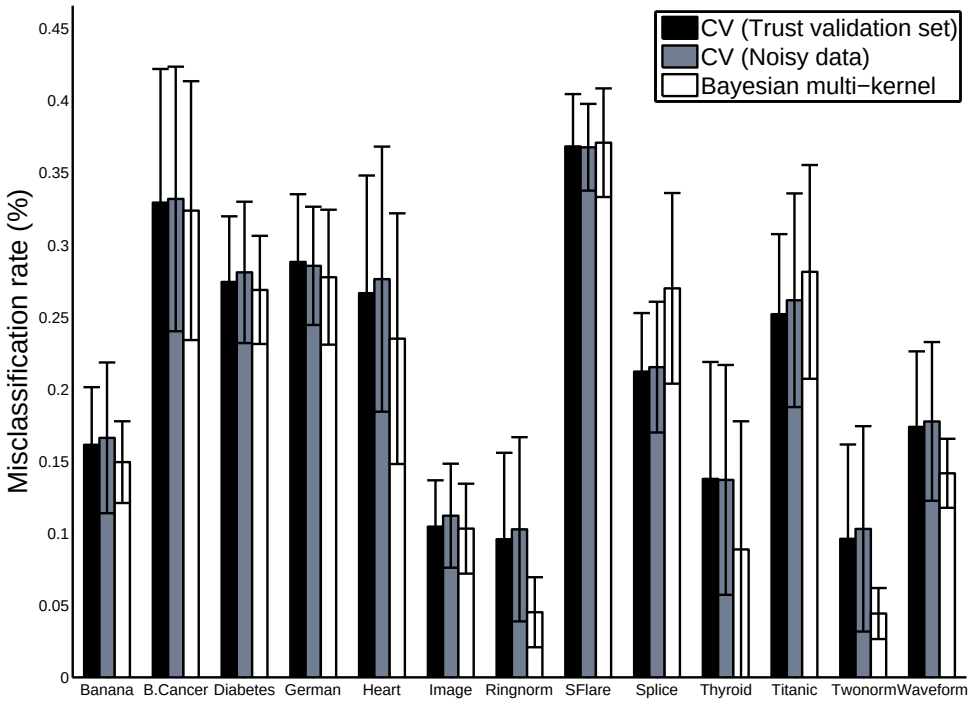


Figure 6.3: Cross validation for kernel width selection on different validation sets versus MKL with Bayesian regularisation.

Results on noisy data

We now move on to more challenging noisy settings. We compare the MKL against CV in a scenario where label errors are present. We shall focus on two aspects, firstly how label noise affects CV based model selection, and secondly how does the MKL compare to CV in this setup. We artificially inject 30% random symmetric and asymmetric noise into the training sets, while keeping the test sets clean. We compare the performance of the proposed model in which the kernel widths were selected using: (1) CV on a trusted validation set (2) CV on noisy validation set (3) MKL with Bayesian regularisation. Figure 6.3 reports the mean errors and standard deviations from 100 repetitions.

Interestingly, we observe negligible difference between doing CV on the originally noisy

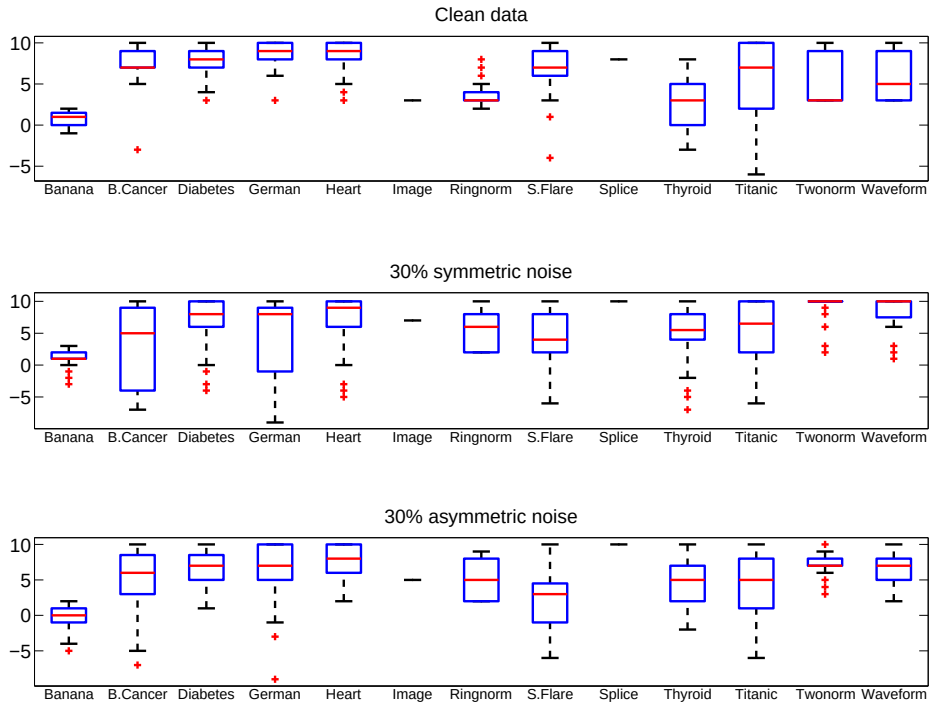


Figure 6.4: Comparison of the medians of the kernel widths selected using clean data and two types of label noise at 30% level, averaged over all data splits. Cross validation was done using noisy validation set

validation versus doing CV on a trusted validation set. More surprisingly we notice that CV on noisy data sometimes produces better results than on the trusted validation set – for example in the *S.Flare* dataset. We conjecture these datasets might originally already contain some label noise. However, the MKL performs better than the others in general. To better understand why CV on noisy validation set is still as good as CV on trusted validation set, we show in Figure 6.4 plots of kernel widths for each dataset from 100 random data splits, where all labels are clean (top), symmetric noises are presented (middle) and asymmetrically noises are presented (bottom). In the noisy scenarios we used noisy validation set to select kernel widths.

Figure 6.4 reveals that in the case of asymmetric noise the medians of the kernel widths

tend to be larger than the ones chosen using clean data while in the symmetric case the widths are mostly in the same proximity as the widths from the clean data. Having a larger width means a wider Gaussian basis function, that is a slight underfitting effect. In the case of rKLR, as tested, it is still legitimate to have a slightly wider width as those points with suspicious labels will likely be flagged as wrong label samples. Consequently CV on noisy labels should not deteriorate classification performance much compared to an idealised CV on trusted validation set – although CV is of course computationally more costly than MKL, as demonstrated in the sequel.

Comparison of the computation time of MKL vs CV

To assess the relative computation time of these methods, we report in Table 6.4 the average running times for a single training/testing split for each data. As we can see, MKL with Bayesian regularisation is approximately 5 to 10 times faster than standard CV.

Dataset	CPU time (seconds)		Dataset	CPU time (seconds)	
	rKLR	rMKLR		rKLR	rMKLR
Banana	48.44 ± 6.13	**10.16 ± 0.28**	*S.Flare*	1033.25 ± 66.55	**26.73 ± 0.74**
B.Cancer	26.61 ± 0.90	**2.96 ± 0.25**	*Splice*	245.61 ± 1.68	**64.12 ± 1.36**
Diabetes	151.78 ± 44.26	**14.84 ± 0.42**	*Thyroid*	23.04 ± 1.25	**1.64 ± 0.29**
German	192.84 ± 54.27	**31.63 ± 3.34**	*Titanic*	15.81 ± 1.74	**1.68 ± 0.12**
Heart	24.23 ± 0.93	**2.15 ± 0.09**	*Twonorm*	49.19 ± 0.81	**10.75 ± 0.36**
Image	1218.50 ± 451.54	**106.99 ± 1.41**	*Waveform*	49.69 ± 1.03	**10.07 ± 1.14**
Ringnorm	51.21 ± 0.78	**10.54 ± 0.34**			

Table 6.4: Running times on a 2.67GHz Intel Core i5 CPU averaged over 10 random splits. The MKL (rMKLR) is 5 to 10 times faster than the traditional CV approach.

Assessing the sensitivity of rMKLR to the number of kernel width choices

Throughout we have used the set of 21 kernel width values to select from in the above experiments. It is then interesting to see how the number of kernels in this set would affect the performance of the proposed technique. To this end, we fix the range of the kernel width choice to the interval $[2^{-10}, 2^{10}]$ as before, but vary the number of available kernel width choices within this range from 11 up to 81. We inject a mix of symmetric

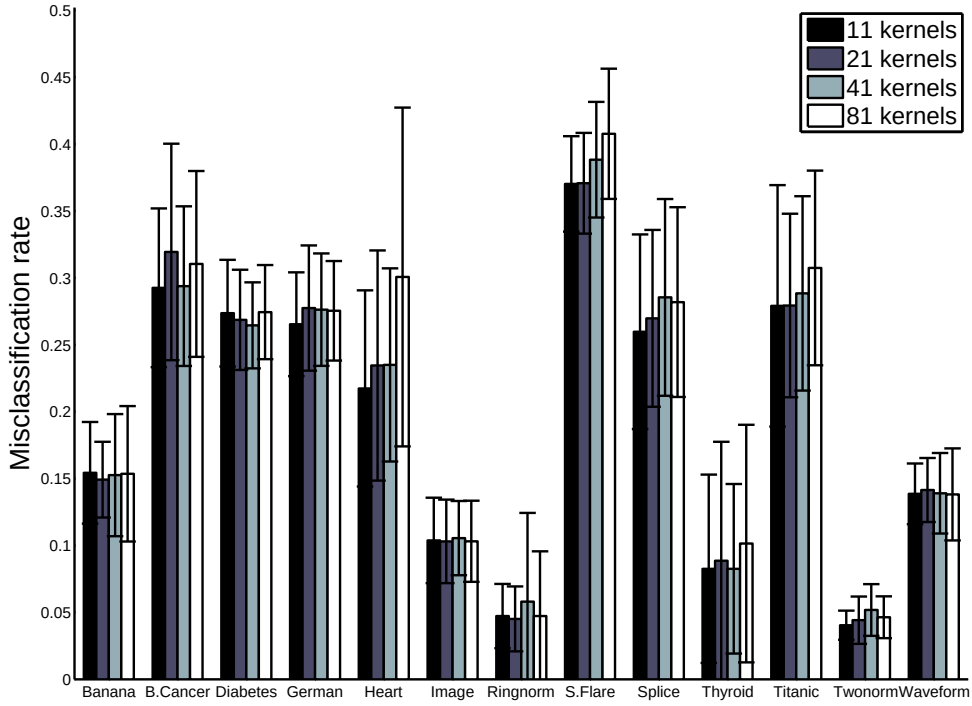


Figure 6.5: Comparison of the different kernel set size at 30% level, averaged over 100 random runs.

and asymmetric noise at 30% level into the training sets and validate the model on clean test sets. The results are given in Figure 6.5 and show that although the classification performances vary somewhat as the size of the set of kernel width choices varies, the differences in most cases are marginal, with the standard error bars are substantially overlapping. This clearly demonstrates that the number of kernels in the kernel width set has a small effect on the classification performance and we are free to choose any reasonable configuration. However one has to bear in mind that having larger kernel set means a longer training time.

6.2.4 Comparisons with state-of-the-art classifiers

In our final controlled experiment, we compare rMKLR to three state-of-the-art classifiers: rKFD , the SVM and the model-free method of StPMKL. The comparison with rKFD allows us to see comparative performance between generative and discriminative model in noisy settings. We compare with SVM to find out to what extent class label noise could be considered to be a normal part of any classification problem and conversely, to what extent it actually needs the special treatment that we developed in the previous sections. The comparison with StPMKL will give insight into how does our simple modelling assumptions about the label flipping process compare with the model-free approach in StPMKL.

rMKLR versus rKFD versus SVM

To make a fair comparison of our rMKLR to its generative counterpart rKFD, as well as the 'gold standard' SVM, the kernel widths for rKFD and SVM (and additionally SVM's C parameter) were selected using CV without a trusted validation set. We perform 100 independent repeated experiments for symmetric and asymmetric noise which ultimately gives us 200 repetitions in total, and we do this at 10% and 30% levels of label noise contamination. Table 6.5 summarises our findings. We observe rMKLR substantially outperformed the rKFD in 5 out of 13 data sets at 10% noise and constantly dominates as noise level increases. The finding is also statistically significant as tested by Friedman + Nemenyi post-hoc test. We observe that the difference between the rank of the two is larger than the critical difference value. It is thus apparent that, as far as classification is concerned, a discriminative classifier such as rMKLR has an edge over rKFD. The SVM is doing very well as a straight-out-of-the-box classifier. There is no doubt that slack variables play an important role in SVM's robustness. Even though the difference between rMKLR and SVM is not statistically significant, rMKLR seems to perform better

in higher noise case.

Dataset	10 % noise			30 % noise		
	rKFD	SVM	rMKLR	rKFD	SVM	rMKLR
Banana	12.39 ± 1.13	11.55 ± 1.07	11.39 ± 0.79	20.42 ± 6.07	17.63 ± 5.21	14.92 ± 2.83
B.Cancer	28.71 ± 4.81	27.90 ± 5.05	27.93 ± 4.50	33.50 ± 8.21	32.95 ± 8.39	32.36 ± 8.98
Diabetes	27.15 ± 2.51	24.21 ± 2.07	24.56 ± 2.00	34.47 ± 4.77	29.60 ± 3.94	26.87 ± 3.75
German	26.93 ± 2.64	24.73 ± 2.57	24.12 ± 2.38	32.34 ± 4.56	29.80 ± 4.11	27.75 ± 4.68
Heart	18.96 ± 4.10	17.60 ± 3.92	17.27 ± 3.48	26.50 ± 9.18	25.31 ± 8.21	23.49 ± 8.69
Image	5.25 ± 0.88	4.95 ± 0.85	6.09 ± 1.19	12.41 ± 3.00	10.24 ± 2.33	10.31 ± 3.12
Ringnorm	2.32 ± 0.44	1.84 ± 0.49	2.20 ± 0.48	6.88 ± 2.33	3.24 ± 1.95	4.51 ± 2.43
S.Flare	35.37 ± 1.91	34.25 ± 2.24	34.93 ± 1.96	38.51 ± 4.12	38.65 ± 4.21	37.08 ± 3.77
Splice	15.09 ± 1.47	13.12 ± 1.14	15.11 ± 1.80	29.89 ± 5.52	21.46 ± 2.34	26.98 ± 6.61
Thyroid	7.07 ± 3.96	6.03 ± 3.13	6.15 ± 2.75	15.93 ± 8.76	12.65 ± 7.99	8.87 ± 8.89
Titanic	24.22 ± 2.23	22.88 ± 1.27	23.00 ± 1.12	29.25 ± 8.86	27.80 ± 8.36	28.12 ± 7.41
Twonorm	2.61 ± 0.29	2.88 ± 0.52	2.91 ± 0.36	3.08 ± 1.48	5.38 ± 2.57	4.42 ± 1.77
Waveform	12.82 ± 1.40	11.12 ± 1.06	10.93 ± 0.76	19.75 ± 3.38	16.40 ± 3.33	14.15 ± 2.39
Average rank	2.6923	1.3846	1.9231	2.7692	1.8462	1.3846
p-value	0.0036			0.0016		
Nemenyi CD	0.9190					

Table 6.5: Comparative performance of rKFD and SVM by CV and the proposed rMKLR. Average errors, standard deviations and average ranks from Friedman test at 5% level are reported.

rMKLR versus StPMKL

As a final experiment, before diving into real applications, we compare our rMKLR to the model-free multiple kernel learning algorithm for noisy labels called StPMKL. We follow the experimental protocol discussed in Xu et al. [2010] and Yang et al. [2012] to generate multiple RBF kernels with 10 different widths $\{2^{-3}, 2^{-2}, \ldots, 2^{6}\}$ for individual features as well as for all features, leading to $S = 10(m + 1)$ kernels in total for each dataset – where m is the dimensionality of the data. In addition to the *Heart* dataset from the 13 benchmark datasets we analysed before, we use *Ionosphere* and *Australia* datasets from UCI repository in this experiment so that we can compare directly with quoted results for StPMKL. We perform 20 repetitions using 80% train and 20% test random split. The statistics of the datasets used in this experiment are summarised in Table 6.6.

Figure 6.6 shows the classification accuracy of the algorithms with label noise levels varied from 0% to 40% on the three datasets. The results for StPMKL are quoted from Yang et al. [2012]. We observe that the performance of rMKLR is similar to that of

Data set	# of Examples	Dimensionality	# of Kernels
Ionosphere	351	34	350
Heart	270	13	140
Australia	690	14	150

Table 6.6: Characteristics of the datasets used in the comparison between the MKL algorithms rMKLR and StPMKL.

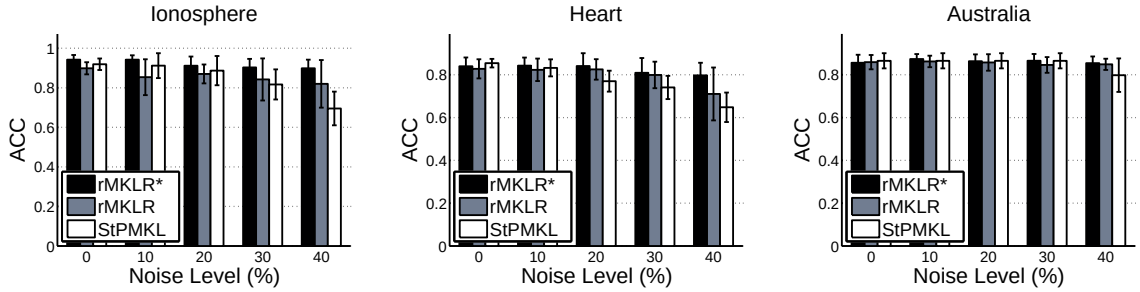


Figure 6.6: Comparison of classification accuracy (ACC) with noise level ranging from 0% to 40%.

StPMKL when there is either (i) no label noise or (ii) mild label noise, but StPMKL tends to perform better when the label noise level is high. However, we suspect that the experimental procedure that generates multiple kernels for each feature is not particularly suitable for rMKLR due to the sparsity promoting regularisation that the algorithm used. We then repeat the experiment with multiple kernels generated from full-length input vectors (not each feature individually). The results, denoted as 'rMKLR*', turn out to be very interesting. We observe significant boost in classification accuracy and see that rMKLR* outperforms StPMKL in almost all cases. The results also demonstrate convincingly that a model-based approach does not over-simplify the label noise problem and it is practically useful.

6.2.5 Real applications

Recognising Textual Entailment

For the first real world problem, we test the proposed method on a variant of the PASCAL2 competition data discussed in Snow et al. [2008] and Raykar et al. [2010]. The dataset

contains 800 sentence pairs. An annotator was asked if the second sentence follows from the first sentence. There are 164 distinct annotators in total, of which only one annotator has labelled all sentence pairs. On average an annotator has completed 53 out of 800 pairs which results in a sparse 800×164 matrix. Apart from that, the actual ground truths are also given. The task is to estimate and predict the ground truths using a varying number of annotators. For this type of task, majority voting has long been a standard approach but Raykar et al. [2010] has already demonstrated that we can do better. This is apparent in our results too. We measure the accuracy of the estimated ground truth while varying the number of annotators, at which point we perform 100 independent random repetitions. The overall results together with those quoted from Raykar et al. [2010] are summarised in Figure 6.7.

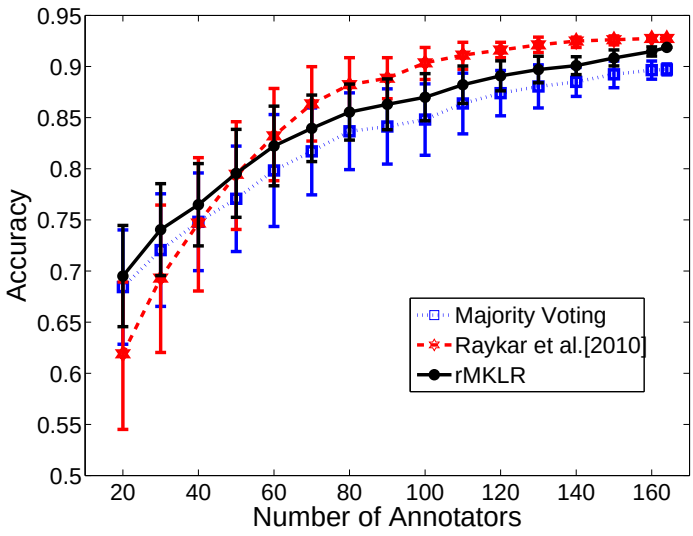


Figure 6.7: Accuracy versus number of annotators in Textual Entailment recognition task. Each result is obtained by 100 independent draws without replacement from the total of 164 annotators.

We find that rMKLR uses fewer annotations to achieve the same accuracy as the majority voting. We further observe that rMKLR outperforms the EM-algorithm based approach (as employed in Raykar et al. [2010]) when limited annotations are provided.

This is because the model discussed in Raykar et al. [2010] is formulated such that each annotator has their own gamma matrix whereas ours employs a single gamma matrix. The advantage of having separate gamma matrices is to be able to assess the quality of each annotator separately, but the algorithm inevitably requires more labels in order to perform well. A single gamma matrix, however, suffers less from scarcity of labels but has to pay the price when more labels become available.

Image classification using cheaply acquired labelled data

In this part of the experiment we will use the raw data from *Websearch* dataset introduced in Chapter 4. We shall restate the problem for the sake of readability and completeness. Suppose we were to train a classifier to recognise images that contain a bike. The standard machine learning approach is to collect training images representing 'bike', as well a counterexamples, and laboriously label each of them. Here, we suggest that we could reduce human intervention and obtain the training data cheaply using annotated data from search engines. By searching for images using the keyword 'bike' and 'not bike', we obtain a set of images that are *loosely* categorised into 'bike' class versus 'not bike' class. This allows us to acquire a large number of training data quickly and cheaply. The problem is of course that the annotations returned by the search engine are somewhat unreliable. Here we demonstrate that the proposed model is useful in such circumstances. We collected 515 images using the keyword 'bike' and 515 images using the keyword 'not bike' from Google. We also manually labelled all images, but will not use these labels for training. The manual labels were determined as the following: a 'bike' image is one that contains a bike as its main object and we make no distinction between a bicycle and a motorbike. Everything else is labelled as 'not bike'. This reveals 83 flips from 'bike' to 'not bike' images and 100 flips from 'not bike' to 'bike' relative to the labels from the search engine. The manually labelled set is only used for testing purposes.

The images are passed through a series of pre-processing steps, which have some subtle

differences from the steps used in Chapter 4. Here, we engineer the preprocessing steps to take the full advantage of the MKL. We extracted a meaningful visual vocabulary using dense SIFT (Lowe [1999]), then extracted texture information using Local Binary Pattern (LBP) (Ojala et al. [2002]), and finally extracted Pyramid Histogram of Oriented Gradients (PHOG) descriptors (Dalal and Triggs [2005]). Having three distinct types of features allows us to exploit the original idea behind MKL where heterogeneous data are combined. We construct 21 RBF base kernels for each types of feature, which results in 63 base kernels in total. We employ rMKLR to learn logistic regression parameters as well as the combination of kernels.

We repeated 100 independent bootstrap classification experiments using 80/20 random splits and employed KLR, rMKLR and additionally linear rLR to perform the task comparatively. The rMKLR attains an average generalisation error of 14.19% $\pm$ 0.02 while traditional KLR and linear rLR lag behind. This result is summarised in Table 6.7, and highlights the advantage of our new robust kernel machine and how badly KLR was affected by label noise. A subset of classification results from rMKLR are depicted in Figure 6.8 and Figure 6.9 for visual inspection. We see that rMKLR is able to detect mislabelled instances effectively. On the basis of these results we believe that there is high potential for learning from unreliable data from the Internet using the label-noise robust algorithm proposed.

Classifier	rLR	KLR	rMKLR
Error rate	18.17% $\pm$ 0.02	21.44% $\pm$ 0.03	**14.19% $\pm$ 0.02**

Table 6.7: Comparative results between rMKLR, KLR and linear rLR on the noisy label image classification task. The proposed rMKLR is the best performer. Interestingly, linear rLR also outperforms the traditional KLR.

Figure 6.8: Examples of positive class ('Bike') predictions sorted by their posterior probability. Boxed images illustrate disagreement between the classifier (denoted as P) and the provided labels from Google (denoted as L).

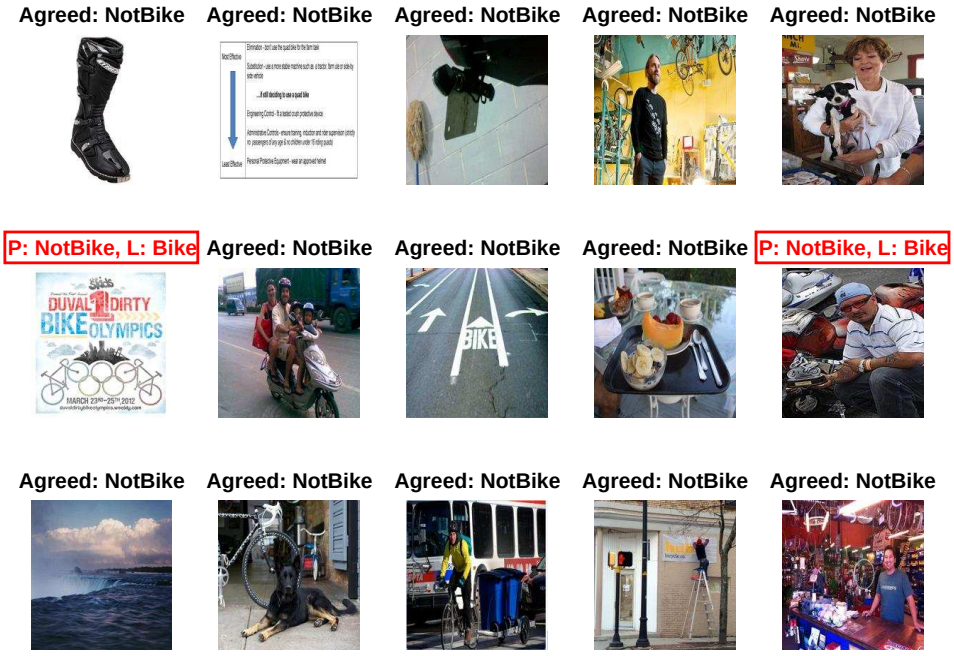


Figure 6.9: Examples of negative class ('NotBike') predictions sorted by their posterior probability. Boxed images illustrate disagreement between the classifier (denoted as P) and the provided labels from Google (denoted as L).

6.3 Extension to multi-class problems

The proposed multi-kernel approach with Bayesian regularisation technique can be straightforwardly extended to a multi-class problem. In multi-class setting, where $\tilde{y} \in \{1, \ldots, K\}$, the class posterior of the true label is typically modelled by the softmax function

$$p(y = k|\kappa(\cdot, \mathbf{x}_n), \mathbf{w}_k) = \frac{\exp(\mathbf{w}_k^T \kappa(\cdot, \mathbf{x}_n))}{\sum_{j=1}^{K} \exp(\mathbf{w}_j^T \kappa(\cdot, \mathbf{x}_n))} \tag{22}$$

Using this we can write the likelihood of the observed label as the following:

$$p(\tilde{y} = k|\kappa(\cdot, \mathbf{x}_n), \Theta) = \sum_{j=1}^{K} \gamma_{jk} p(y = j|\kappa(\cdot, \mathbf{x}_n), \mathbf{w}_j) \tag{23}$$

which brings us to the objective of the 'robust Multi-Class Multi-Kernel Logistic Regression'.

$$\sum_{n=1}^{N} \sum_{k=1}^{K} \mathbb{1}(\tilde{y}_n = k) \log \tilde{P}_n^k - \sum_{k=1}^{K} \zeta_k \sum_{n=1}^{N} w_{nk}^2 - \sum_{i=1}^{S} \xi_i \eta_i \tag{24}$$

The optimisation of the objective proceeds in the same way as in the binary case by using a conjugate gradient method. The gradient of the objective w.r.t. $\mathbf{w}_c$ is given by:

$$\mathbf{g}_{\mathbf{w}_c} = \sum_{n=1}^{N} \sum_{k=1}^{K} \frac{\mathbb{1}(\tilde{y}_n = k)}{\tilde{P}_n^k} \frac{\left(\sum_{j=1}^{K} (\gamma_{ck} - \gamma_{jk}) e^{(\mathbf{w}_j^T \kappa(\cdot, \mathbf{x}_n))}\right) e^{(\mathbf{w}_c^T \kappa(\cdot, \mathbf{x}_n))} \cdot \kappa(\cdot, \mathbf{x}_n)}{\left(\sum_{l=1}^{K} e^{(\mathbf{w}_l^T \kappa(\cdot, \mathbf{x}_n))}\right)^2} - \zeta_c \sum_{n=1}^{N} w_{nc}^2 \tag{25}$$

And w.r.t. $u_i = \sqrt{\eta_i}$:

$$\mathbf{g}_{u_i} = \sum_{n=1}^{N} \sum_{c=1}^{K} \sum_{k=1}^{K} \frac{\mathbb{1}(\tilde{y}_n = k)}{\tilde{P}_n^k} \frac{\left(\sum_{j=1}^{K} (\gamma_{ck} - \gamma_{jk}) e^{(\mathbf{w}_j^T \kappa(\cdot, \mathbf{x}_n))}\right) e^{(\mathbf{w}_c^T \kappa(\cdot, \mathbf{x}_n))} (\mathbf{w}_c^T \kappa_i(\cdot, \mathbf{x}_n))}{\left(\sum_{l=1}^{K} e^{(\mathbf{w}_l^T \kappa(\cdot, \mathbf{x}_n))}\right)^2} - 2\xi_i u_i \tag{26}$$

Since we treat weight vectors of each class separately, the regularisation parameters can then be determined using Eq.(20) and Eq.(11) without the need of modification. Further, the estimates of the elements of the flip matrix γ_{jk} can be obtained by efficient multiplicative update equations:

$$\gamma_{jk} = \frac{1}{C} \times \gamma_{jk} \sum_{n=1}^{N} \frac{\mathbb{1}(\tilde{y}_n = k)}{\tilde{P}_n^k} \frac{e^{(\mathbf{w}_j^T \mathbf{x}_n)}}{\sum_{l=0}^{K-1} e^{(\mathbf{w}_l^T \mathbf{x}_n)}} \tag{27}$$

where the constant term C equals $\sum_{k=0}^{K-1} \gamma_{jk} \sum_{n=1}^{N} \frac{\mathbb{1}(\tilde{y}_n=k)}{\tilde{P}_n^k} \frac{e^{(\mathbf{w}_j^T \mathbf{x}_n)}}{\sum_{l=0}^{K-1} e^{(\mathbf{w}_l^T \mathbf{x}_n)}}$.

6.4 Summary

We proposed a novel algorithm to learn a label-noise robust Kernel Logistic Regression model. The essence of the algorithm is a novel model selection approach where the optimal hyper-parameters are automatically determined using multiple kernel learning and Bayesian regularisation techniques. The experimental results show that the latent variable model used is robust against mislabelling while the proposed learning algorithm is faster and has superior predictive abilities than traditional approaches. In comparisons with three state-of-the-art kernel machines in controlled settings we observed significant improvements over the previously existing Kernel Fisher Discriminant classifier and even the Multiple Kernel Learning algorithm developed specifically for noisy labels. Finally, we demonstrated real-world applications to learning from crowd-sourcing data, learning from cheaply obtained but unreliable annotated data.

CHAPTER 7

Ensemble of Robust Classifiers

So far, we have only studied the performance of a single robust classifier. In this chapter we will take a step forward to study the performance of the proposed robust classifiers collectively. We will employ a boosting-type ensemble method called AdaBoost in this study. It is well known to practitioners that boosting is sensitive to label noise. The issue stems directly from the fundamental concept of boosting in that the effort is directed towards classifying the difficult samples. In fact, the complexity of traditional boosting is very high, so much so that for a dataset with any configuration of its labels, it is possible to draw a decision boundary with zero training error. This seems to be a good approach to the classification problem if the difficult samples are not mislabelled samples in the first place.

We shall investigate the solution to boosting in the presence of label noise at two different levels. At the lower level we study the robust committee where robust classifiers are combined and boosted using existing AdaBoost algorithm. At the higher level, we propose a new robust boosting algorithm that we call 'rBoost' where the objective function is a convex combination of two exponential losses. The coefficients of the combination represent uncertainty in the observed labels. The new boosting algorithm is closely related to

AdaBoost and requires a relatively minor modification to the existing algorithm. Moreover, the study of the robust boosting will enable us to verify our hypothesis regarding the limitation of the robustification technique.

7.1 A robust base learner

In recent years many classifiers have been introduced to tackle the problem of learning in the presence of label noise. To date, there are a number of classifiers developed specifically for dealing with label noise: robust logistic regression, robust Fisher discriminant analysis, robust Gaussian Process (Hernández-Lobato et al. [2011]) or robust Nearest Neighbours (Barandela and Gasca [2000]). All of these can potentially be used as a base classifier, and it is then interesting to see how would such classifiers behave collectively in an ensemble. One way to construct a robust classifier is through a probabilistic latent variable model. Under the latent model, a robust classifier attempts to learn a posterior probability of the true labels via the likelihood of the observed labels.

Recall from previous chapters that the likelihood of the observed label $\tilde{y}$ of a point $\mathbf{x}_n$ given the current parameter setting is defined as the following:

$$\tilde{P}_n^k = p(\tilde{y} = k|\mathbf{x}_n, \theta, \{\omega_{jk}\}_{j,k=0}^{1}) = \sum_{j=0}^{1} \omega_{jk} p(y = j|\mathbf{x}_n, \theta) \tag{1}$$

that is, a linear combination of the 'true' class posteriors. We will use ω to represent label flipping probability of a base learner and reserve γ for representing label flipping coefficient in the new robust boosting that will be introduced shortly. From this assumption the modified log-likelihood is given by

$$\mathcal{L}(\theta, \Omega) = \sum_{n=1}^{N} \sum_{k=0}^{1} \mathbb{1}(\tilde{y}_n = k) \log(\tilde{P}_n^k) \tag{2}$$

Note that any probabilistic classifier yielding class posterior probability will fit the frame-

work and can be converted into a robust classifier using the technique shown. For the sake of concreteness we will employ logistic regression with parameter $\theta = \boldsymbol{\beta}$ in this study. Recall from Chapter 4 that the likelihood of $\tilde{y} = 1$ is defined as:

$$\tilde{P}_n^1 = \omega_{11}\sigma(\boldsymbol{\beta}^T\mathbf{x}_n) + \omega_{01}(1 - \sigma(\boldsymbol{\beta}^T\mathbf{x})_n) \tag{3}$$

Here, $\boldsymbol{\beta}$ is the weight vector orthogonal to the decision boundary and it determines the orientation of the separating plane and $\sigma(a) = 1/(1 + \exp(-a))$ is the sigmoid function. Learning robust logistic regression model involves estimating $\boldsymbol{\beta}$ and well as ω_{jk}. We will follow the steps presented in Chapter 4 where the conjugate gradient method is used to optimise $\boldsymbol{\beta}$. The gradient of the log-likelihood w.r.t. the weight vector is

$$\sum_{n=1}^{N}\left[\left(\frac{\tilde{y}_n(\omega_{11} - \omega_{01})}{\tilde{P}_n^1} + \frac{(1 - \tilde{y}_n)(\omega_{10} - \omega_{00})}{\tilde{P}_n^0}\right)\sigma(\boldsymbol{\beta}^T\mathbf{x}_n)(1 - \sigma(\boldsymbol{\beta}^T\mathbf{x}_n))\mathbf{x}_n\right] \tag{4}$$

The following multiplicative updates are then used to estimate γ_{jk}:

$$\omega_{10} = \frac{g_{10}}{g_{10} + g_{11}}, \quad \omega_{11} = \frac{g_{11}}{g_{10} + g_{11}} \tag{5}$$

$$\omega_{00} = \frac{g_{00}}{g_{00} + g_{01}}, \quad \omega_{01} = \frac{g_{01}}{g_{00} + g_{01}} \tag{6}$$

where

$$g_{11} = \omega_{11}\sum_{n=1}^{N}\left(\frac{\tilde{y}_n\sigma(\boldsymbol{\beta}^T\mathbf{x}_n)}{\tilde{P}_n^1}\right)$$

$$g_{10} = \omega_{10}\sum_{n=1}^{N}\left(\frac{(1 - \tilde{y}_n)\sigma(\boldsymbol{\beta}^T\mathbf{x}_n)}{\tilde{P}_n^0}\right)$$

$$g_{01} = \omega_{01}\sum_{n=1}^{N}\left(\frac{\tilde{y}_n(1 - \sigma(\boldsymbol{\beta}^T\mathbf{x}_n))}{\tilde{P}_n^1}\right)$$

$$g_{00} = \omega_{00} \sum_{n=1}^{N} \left(\frac{(1-\tilde{y}_n)(1-\sigma(\boldsymbol{\beta}^T \mathbf{x}_n))}{\tilde{P}_n^0} \right)$$

7.2 The robust boosting

Suppose we have a training set with corrupted labels $S = \{(\mathbf{x}_n, \tilde{y}_n)\}_{n=1}^N$, where $\mathbf{x}_n \in \mathbb{R}^m$ and $\tilde{y}_n \in \{+1, -1\}$. Let a base hypothesis be a decision function $h : \mathbf{x} \to \tilde{y}$. Under the boosting framework, a final hypothesis is a linear combination of the base hypotheses and it takes the following additive form:

$$H(x) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}) \tag{7}$$

In boosting, the 0/1 misclassification loss incurred by the final hypothesis is measured by the exponential loss:

$$\sum_{n=1}^{N} \mathbb{1}(\tilde{y}_n = 1) e^{-H(\mathbf{x}_n)} + \mathbb{1}(\tilde{y}_n = -1) e^{H(\mathbf{x}_n)} \tag{8}$$

This forms a boosting objective that has to be optimised. However, in the situation where labels are contaminated the loss in Eq.(8) is not ideal, for obvious reasons. Instead, we form a new objective which explicitly takes into account uncertainties in labels:

$$\sum_{n=1}^{N} \mathbb{1}(\tilde{y}_n = 1) \Big\{ \gamma_{00} e^{-H(\mathbf{x}_n)} + \gamma_{01} e^{H(\mathbf{x}_n)} \Big\} + \mathbb{1}(\tilde{y}_n = -1) \Big\{ \gamma_{11} e^{H(\mathbf{x}_n)} + \gamma_{10} e^{-H(\mathbf{x}_n)} \Big\} \tag{9}$$

Here, $\gamma_{jk} = p(\tilde{y} = k | y = j)$ are probabilistic factors representing uncertainties in labels. Intuitively, the loss is weighed up or down depending on the gamma parameters γ_{jk}. For example, $\gamma_{01} = 0.3$ and $\gamma_{10} = 0$ indicates the situation where labels in the negative class (or class 0) are all correct – because no flipping from positive to negative occurred – but labels in the positive class are contaminated. Accordingly, the new loss accounts for this

by adjusting the loss for the positive class (class 1) to: $0.7 * e^{-H} + 0.3 * e^{H}$. This is a hyperbolic cosine with the two tails adjusted and it represents the modified loss associated with the positive class. The shapes of such modified loss functions are depicted in Figure 7.1. From the figure we see that the classification that is 'too correct' will be penalised, hence reducing the overfitting problem. Meanwhile the loss of the negative class (class 0), which is $e^{H} + 0 * e^{-H} = e^{H}$, reduces to traditional boosting. It may be interesting to note that a similar shape of the loss can also be obtained by truncating the Taylor expansion of the exponential function to some finite degree. This could also be used to implement the same idea, although it would not have the transparent formulation given above.

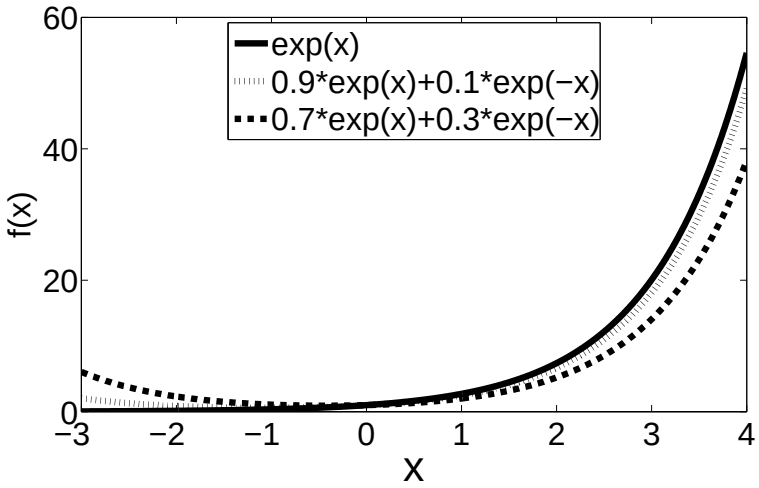


Figure 7.1: Various setups of the Γ and theirs associated loss shape.

7.2.1 Adding a new base learner

Consider the case when $\tilde{y} = 1$, define $d_{00} = e^{-H(\mathbf{x})}$ and $d_{01} = e^{H(\mathbf{x})}$. Likewise, when $\tilde{y} = -1$ define $d_{11} = e^{H(\mathbf{x})}$ and $d_{10} = e^{-H(\mathbf{x})}$ to be an unnormalised distribution of the data $(\mathbf{x}_n, \tilde{y}_n)$. It can be shown that at the iteration t of boosting, minimising the loss in Eq.(9) w.r.t. the new $h_t(\mathbf{x})$ is equivalent to minimising the following (derivation details are given in the Appendix A.1):

$$\begin{aligned}\underset{h,\alpha}{\arg\min}\ \ & 2\sinh(\alpha)\sum_{n=1}^{N}\Big\{w_n\mathbb{1}(h(\mathbf{x}_n)\neq\tilde{y}_n)\Big\}\\ &+e^{-\alpha}\sum_{n=1}^{N}\Big\{\mathbb{1}(\tilde{y}_n=1)w_{00}+\mathbb{1}(\tilde{y}_n=-1)w_{11}\Big\}\\ &+e^{\alpha}\sum_{n=1}^{N}\Big\{\mathbb{1}(\tilde{y}_n=1)w_{01}+\mathbb{1}(\tilde{y}_n=-1)w_{10}\Big\}\end{aligned}\tag{10}$$

where

$$w_n=\begin{cases}(w_{00}-w_{01}), & \text{if } \tilde{y}_n=+1.\\ (w_{11}-w_{10}), & \text{if } \tilde{y}_n=-1.\end{cases}\tag{11}$$

and

$$w_{jk}=\gamma_{jk}\cdot d_{jk}\tag{12}$$

From this, it is immediate to see that in order to minimise the loss we have to seek for $h_t(\mathbf{x})$ that minimises the misclassification error $\epsilon_t=\sum_{n=1}^{N}w_n\mathbb{1}(\tilde{y}_n\neq h(\mathbf{x}))$. The step is identical to the traditional AdaBoost except that the misclassification error of the current classifier is measured against different weighting factors which take into account the uncertainty of the observed noisy label as indicated by γ_{jk}. Note that the expression is fully compatible with the traditional AdaBoost such that the rBoost reduces to the original AdaBoost when $\gamma_{01}=0$ and $\gamma_{10}=0$. We emphasise that the weights in rBoost need not be normalised. In fact, in the original AdaBoost the normalisation simply facilitates the algebra in deriving a closed-form update for α_t.

7.2.2 Updating the weight of base learner

Now in our case, to get the update for α_t we take derivative of Eq.(10) w.r.t. α_t, equate it to zero:

$$\begin{aligned} & 2\cosh(\alpha)\sum_{n=1}^{N}\Big\{w_n\mathbb{1}(h(\mathbf{x}_n)\neq\tilde{y}_n)\Big\} \\ & -e^{-\alpha}\sum_{n=1}^{N}\Big\{\mathbb{1}(\tilde{y}_n=1)w_{00}+\mathbb{1}(\tilde{y}_n=-1)w_{11}\Big\} \\ & +e^{\alpha}\sum_{n=1}^{N}\Big\{\mathbb{1}(\tilde{y}_n=1)w_{01}+\mathbb{1}(\tilde{y}_n=-1)w_{10}\Big\}=0 \end{aligned} \tag{13}$$

Unfortunately, this equation cannot be solved in closed form. We thus resort to numerical optimisation to solve for the α_t. Note that the term which is multiplied by $2\cosh(\alpha)$ is nothing but our error ϵ_t defined earlier.

7.2.3 Updating the weight of data point

Next, to derive the update for the weight vectors, recall that we define $w_{jk}=\gamma_{jk}e^{-\tilde{y}_nH(\mathbf{x}_n)}$. It follows, for example, that the update for w_{00} can be written as:

$$\begin{aligned} w_{00}^{t+1} &= \gamma_{00}e^{-\tilde{y}_n(H+\alpha h)} \\ &= \gamma_{00}e^{-\tilde{y}_nH}\cdot e^{-\tilde{y}_n\alpha h} \\ &= \gamma_{00}d_{00}^{t}\cdot e^{\alpha(2\mathbb{1}(h(\mathbf{x}_n)\neq y_n)-1)} \\ &= \gamma_{00}d_{00}^{t}\cdot e^{2\alpha\mathbb{1}(h(\mathbf{x}_n)\neq 1)}\cdot e^{-\alpha} \\ &\propto \gamma_{00}d_{00}^{t}\cdot e^{2\alpha\mathbb{1}(h(\mathbf{x}_n)\neq 1)} \end{aligned} \tag{14}$$

Since $e^{-\alpha}$ are shared among all w_{jk} it does not affect the optimisation. We also used a simple trick: $-\tilde{y}h = 2\mathbb{1}(h(\mathbf{x}) \neq \tilde{y}) - 1$. Similarly for the rest of the weight vectors we get:

$$w_{01}^{t+1} = \gamma_{01} d_{01}^{t} \cdot e^{2\alpha \mathbb{1}(h(\mathbf{x}_n) \neq -1)} \tag{15}$$

$$w_{11}^{t+1} = \gamma_{11} d_{11}^{t} \cdot e^{2\alpha \mathbb{1}(h(\mathbf{x}_n) \neq -1)} \tag{16}$$

$$w_{01}^{t+1} = \gamma_{10} d_{10}^{t} \cdot e^{2\alpha \mathbb{1}(h(\mathbf{x}_n) \neq 1)} \tag{17}$$

One way to implement this is to keep the distribution d_{jk} separately and multiply it by γ_{jk} to get a new w_{jk} in each iteration.

7.2.4 Updating the label flipping probabilities

At last, we would also like to estimate the label flipping coefficients, γ_{jk}. We could take derivative of the loss incurred by the current ensemble, Eq.(9), w.r.t. each gamma and try to solve this directly. This did not yield satisfactory results in our experience, most likely because the loss lacks probabilistic semantics. The workaround is to convert the output of boosting i.e. H into a probability. There are three popular approaches to do that: 1) Logistic calibration $p(y = 1|\mathbf{x}, H) = 1/(1 + \exp(-H))$ (Friedman et al. [1998]), 2) Platt's calibration $p(y = 1|\mathbf{x}, H) = 1/(1 + \exp(AH + B))$ where A and B need to be learnt (Platt [1999]), and 3) Isotonic Regression (Robertson et al. [1988]). Niculescu-Mizil and Caruana [2005] empirically shows that Platt's technique and Isotonic regression are superior to a simple logistic transform. In addition, Platt's method has a slight advantage over IsoReg on small sample size. Hence, in this study, we will employ Platt's method to get calibrated posterior probabilities.

By converting H to $p(y = 1|\mathbf{x}, H)$, we can estimate the gamma from the following binomial log-loss, or cross-entropy. Using the notation $P(\mathbf{x}) = p(y = 1|\mathbf{x}, H)$ and $\bar{P}(\mathbf{x}) =$

$1 - P(\mathbf{x})$, this is:

$$-\sum_{n=1}^{N} \mathbb{1}(\tilde{y}_n = 1) \log \left\{ \gamma_{11} P(\mathbf{x}_n) + \gamma_{01} \bar{P}(\mathbf{x}_n) \right\} + \mathbb{1}(\tilde{y}_n = -1) \log \left\{ \gamma_{00} \bar{P}(\mathbf{x}_n) + \gamma_{10} P(\mathbf{x}_n) \right\} \tag{18}$$

Following the Lagrangian method which imposes $\gamma_{00} + \gamma_{01} = 1$ and $\gamma_{11} + \gamma_{10} = 1$, and following the derivation in Chapter 4, the multiplicative updates for γ_{jk} are found to be:

$$\gamma_{10} = \frac{g_{10}}{g_{10} + g_{11}}, \quad \gamma_{11} = \frac{g_{11}}{g_{10} + g_{11}} \tag{19}$$

$$\gamma_{00} = \frac{g_{00}}{g_{00} + g_{01}}, \quad \gamma_{01} = \frac{g_{01}}{g_{00} + g_{01}} \tag{20}$$

where

$$g_{11} = \gamma_{11} \sum_{n=1}^{N} \left(\frac{\mathbb{1}(\tilde{y}_n = 1) P_n}{\gamma_{11} P_n + \gamma_{01} \bar{P}_n} \right)$$

$$g_{10} = \gamma_{10} \sum_{n=1}^{N} \left(\frac{\mathbb{1}(\tilde{y}_n = -1) P_n}{\gamma_{10} P_n + \gamma_{00} \bar{P}_n} \right)$$

$$g_{01} = \gamma_{01} \sum_{n=1}^{N} \left(\frac{\mathbb{1}(\tilde{y}_n = 1) \bar{P}_n}{\gamma_{11} P_n + \gamma_{01} \bar{P}_n} \right)$$

$$g_{00} = \gamma_{00} \sum_{n=1}^{N} \left(\frac{\mathbb{1}(\tilde{y}_n = -1) \bar{P}_n}{\gamma_{10} P_n + \gamma_{00} \bar{P}_n} \right)$$

The new rBoost algorithm is summarised in Algorithm 8.

Worth noting that rBoost algorithm has some analogies with cost-sensitive boosting (Fan and Stolfo [1999], Masnadi-Shirazi and Vasconcelos [2011]). One major difference is that the weighting factors in our case are outside of the exponential, whereas they are inside the exponent in the mentioned works. In Fan and Stolfo [1999] the author did briefly discuss the possibility of having the weighting factors outside of the exponential, however their update of the weight is different from ours. Besides, the goal of cost-

sensitive methods is different from ours. In cost-sensitive framework the cost is assumed to be known or given by the expert, and there is no implication of labelling errors.

Algorithm 8 The robust Boosting, 'rBoost', algorithm

Input: data $\{\mathbf{x}_n, \tilde{y}_n\}_{n=1}^N$, boosting round T
Initialize $w_{jk} = \gamma_{jk}$
for $t = 1$ **to** T **do**
(1)$h_t = \arg\max_{\boldsymbol{\beta}}$ Eq.(2) weighted by w_n.
(2)Calculate the error w.r.t. w_n defined in Eq.(11)
$\epsilon_t = \sum_{n=1}^N w_n \mathbb{1}(\tilde{y}_n \neq h_t(\mathbf{x}_n))$
(3)Optimise α_t numerically using the gradient in Eq.(13).
(4)Update w_{jk} according to Eq.(14)–(17).
(5)Calculate $p(y = 1|\mathbf{x}, H)$ using Platt's method.
(6)Update γ_{jk} using Eq.(19)-(20).
end for
Output the final classifier $sign(\sum_{t=1}^T \alpha_t h_t)$.

7.3 Empirical evaluation

This section will investigate the performance of our robust boosting methods in practice. In addition, our new rBoost algorithm will be compared to the standard AdaBoost, GentleAdaBoost and ModestAdaBoost.

7.3.1 Methodology

We will study 4 configurations of base-learner and booster pairs: 1) LR + AdaBoost, 2) rLR + AdaBoost, 3) LR + rBoost and 4) rLR + rBoost. These four combinations will shed light on whether 1) a robust committee is robust against label errors?, 2) the new rBoost can counteract the bad effects of label noise? and finally 3) What can we get from pairing them together? We set our baseline to be the GentleAdaBoost and ModestAdaBoost where the base learner is a decision tree with maximum node splits of 2. For LR to serve as a weak learner, we employ random subsampling to create diversity in the ensemble. Further, we create two types of training sets by artificially injecting symmetric and asymmetric label noise at 10% rate, as well as at 30% contamination rate

Data set	Training samples	Test samples	Pos. samples	Neg. samples	Dimensionality
Banana	400	4900	44.83%	55.17%	2
B.Cancer	200	77	29.28%	70.72%	9
Diabetes	468	300	34.90%	65.10%	8
German	700	300	30.00%	70.00%	20
Heart	170	100	44.44%	55.56%	13
Image	1300	1010	56.95%	43.05%	18
Ringnorm	400	7000	49.51%	50.49%	20
S.Flare	666	400	65.28%	34.72%	9
Splice	1000	2175	44.93%	55.07%	60
Thyroid	140	75	30.23%	69.77%	5
Titanic	150	2051	58.33%	41.67%	3
Twonorm	400	7000	50.04%	49.96%	20
Waveform	400	4600	32.94%	67.06%	21

Table 7.1: Characteristics of the datasets used.

into the training data. We train on the corrupted training set and validate the performance of the ensemble on a clean test set. We report the average and standard deviation of the misclassification rates from 10 independent random repetitions of 150 rounds of boosting each.

7.3.2 Datasets

Again, we use 13 UCI benchmark datasets (Rätsch et al. [2001]) in our controlled experiments. Each problem has been split into 100 train/test realisations except the *Image* and *Splice* datasets where 20 realisations are provided. The characteristics of the datasets used are summarised in Table 7.1.

7.3.3 Results and discussion

We first investigate the behaviour of the robust classifiers as weak learners within the original AdaBoost algorithm. From the leftmost column of Tables 7.2-7.5, we see that when a robust classifier is used as a base learner the generalisation error of the ensemble is already lower compared to the original non-robust AdaBoost in 4 out of 7 datasets. The finding is consistent across all noise levels. It very interesting to observe this because even though the base classifier is robust, it is still under the control of the original AdaBoost. Namely, the boosting will still guide the classifiers to focus on the more difficult parts of the dataset (which of course are likely to contain the points whose labels are wrong).

Why is then this committee of robust classifiers more accurate? Lower error can come from two different sources: Either the robust committee is indeed robust against labelling errors, or it simply converges faster. To check this we run both configurations for more rounds to see the dynamics of the ensemble. Plotted in Figure 7.2 and Figure 7.3 are the training and test errors of AdaBoost using the robust classifiers (rLR) as well as using the traditional classifiers (LR) on selected datasets. Superimposed for reference are ModestAdaBoost and rLR + rBoost.

It turns out that the robust committee converges much quicker than the non-robust committee. However when boosted long enough we are starting to see that their classification performances become very similar. This answers our first research question. The robust classifiers as weak learners introduce what is understood to be a 'good diversity' in the ensemble, and drives the ensemble to convergence much more quickly than the non-robust committee. Unfortunately however, the robustness of the base learner is not enough to withstand the effect of labelling errors.

Now we see that having rLR as a base learner alone is not enough to counteract the bad effects of mislabelling. We investigate further if we can pair rLR, which has a fast convergence rate with our new rBoost algorithm.

Before proceeding, we need to establish that rBoost is superior to original AdaBoost when there is label noise. To this end, we consider two combinations: 1) AdaBoost+LR and 2)rBoost + LR in Tables 7.2-7.5. From the tables we see that rBoost+LR performs comparably to its non-robust booster counterpart when the noise rate is relatively low, and in the case of symmetric label case (i.e. the easy cases in terms of label noise). However when the noise is asymmetric and more severe (Table 7.5), rBoost substantially outperforms the original AdaBoost in general as test using Friedman test + Nemenyi post-hoc test. This answers our second research question. That is, rBoost improves over the original AdaBoost in terms of classification performance especially in higher label

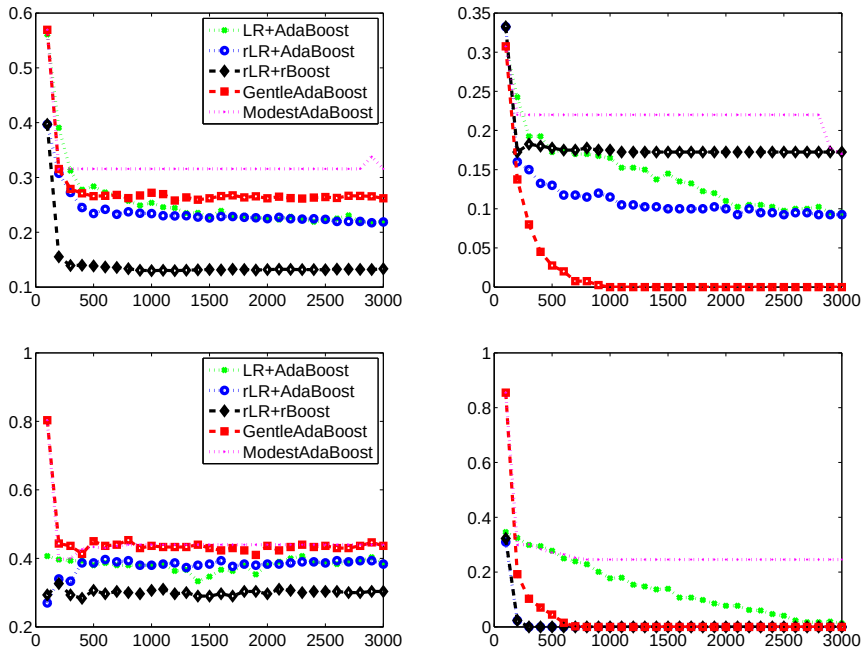


Figure 7.2: Test error(left) and training error(right) for *Banana* and *Diabetes* datasets. The x-axis indicates boosting rounds while the y-axis shows classification errors.

contamination rate conditions and in asymmetric label noise conditions (i.e. the difficult cases).

Next, we equip our rBoost method with the robust base classifiers that enjoy fast convergence to obtain our final robust boosting algorithm. These results are shown in the fourth column of Table 7.5. The superior performance of this approach is most apparent, and we also give an illustrative example of the working of our rBoost on the *Banana* dataset in Figure 7.4. We see that the original AdaBoost generated a patchy decision boundary as a result of label noise, while our rBoost returned a smoother and more appropriate decision boundary.

Further, we validate our approach for estimating the flip probabilities γ_{jk} using the multiplicative updates given in Eq.(19) and Eq.(20). Disappointingly, we see that the results (5th and 6th column of Tables 7.2-7.5) are not as good as the ideal setting where the γ_{jk} are fixed to the true value (rBoost-Fixed gamma). The reason is that the estimated

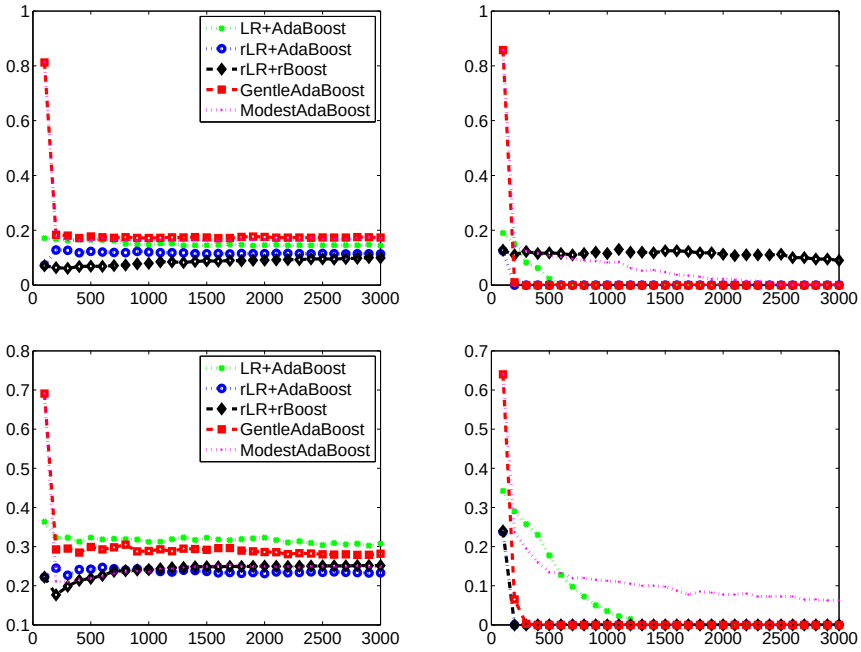


Figure 7.3: Test error(left) and training error(right) for *Twonorm* and *Waveform* datasets. The x-axis indicates boosting rounds while the y-axis shows classification errors.

gamma table tends to converge to the identity (suggesting that there is no label error). This is an indication that the classifier is so powerful that even the robustification cannot prevent it from overfitting the label noise. However, and more interestingly, we observe that the quality of the estimated gammas depends highly on the quality of the calibrated probability used in the update. Assuming that we have a trusted validation set that we can use to obtain a more accurate calibrated probability, we ask how well can we estimate the gammas? We hold out a small subset of the dataset, where all of the labels are clean. This will be our trusted validation set, and we took this set as tiny as 20 points only. We feed this small trusted dataset into the Platt's calibration method. We carried out this experiment on *Banana*, *Image* and *Twonorm*. The classification error from 10 repeated runs of our rBoost algorithm with the use of the trusted validation set as a source for calibrating the probability is 15.74±0.23% on *Banana* at 30% asymmetric noise, compared to 23.83% without. This is taken from the sixth column of Table 7.5. On *Image* at 30%

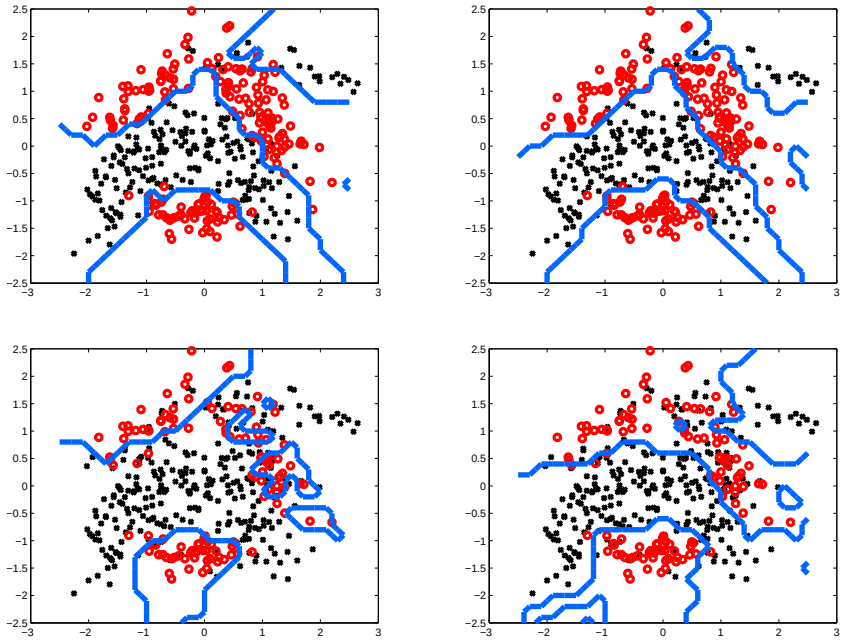


Figure 7.4: Comparison of the decision boundaries obtained from AdaBoost(left) and rBoost(right) in noise-free case(top) and 30% asymmetric noise case(bottom) on *Banana* dataset.

asymmetric noise the error is as low as 7.61±0.19% and on *Twonorm* it is 9.73±0.31%. Intriguingly, a tiny trusted set of 20 points is able to improve the situation even for the *Image* data, where the training set size is as large as 1300 (80% of total number of samples in *Image*). Thus we can conclude that the trusted validation set approach may be seen as a technique to effectively and efficiently incorporate extra knowledge about the labels into the rBoost algorithm. We should note, this differs from simply including the trusted samples into the training set, since the latter would simply make a slight reduction of the noise rate. Of course, the larger the trusted validation set for calibration, the better probability calibration we can expect, and consequently this should lead to more accurate estimates of the gammas (γ_{jk}), and hence to better classification performance.

Dataset	AdaBoost		rBoost-Fixed gamma		rBoost		Gentle	Modest
	LR	rLR	LR	rLR	LR	rLR	Boost	Boost
Banana	18.53±1.0	13.13±1.1	17.53±1.8	12.94±0.9	17.44±1.8	12.96±0.9	16.09±1.6	21.87±3.4
B.Cancer	33.12±4.8	33.12±3.7	31.04±3.1	31.43±4.4	32.08±4.2	35.58±3.8	32.60±5.7	29.35±4.1
Diabetes	24.00±2.7	25.80±2.3	24.10±1.7	25.63±1.5	23.87±1.9	25.20±2.4	27.40±2.0	24.33±1.9
German	25.30±2.8	26.33±2.2	25.73±1.9	26.00±3.0	24.87±2.6	25.90±3.2	27.93±2.0	28.03±1.8
Heart	21.20±4.4	21.60±3.1	22.40±3.9	20.30±3.5	22.10±4.8	20.90±4.4	23.60±3.1	22.40±3.5
Image	14.61±1.3	4.08±1.0	15.29±1.5	4.49±0.8	13.51±1.4	4.12±0.8	4.43±0.7	15.91±3.1
Ringnorm	27.47±1.2	29.77±2.0	27.05±1.2	37.23±3.0	27.37±0.9	30.06±2.1	14.68±1.8	11.89±1.5
S.Flare	35.52±1.3	34.85±1.7	35.00±1.0	35.45±1.3	35.32±1.4	35.18±1.4	35.98±1.5	34.48±4.4
Splice	18.49±0.9	17.07±1.1	19.32±1.0	20.18±1.9	18.54±0.9	16.96±1.0	11.91±1.1	5.91±1.0
Thyroid	12.40±5.0	11.73±5.2	11.47±5.2	13.73±4.3	13.07±4.7	12.40±4.5	10.93±2.9	6.80±4.1
Titanic	22.76±1.3	22.32±1.1	22.97±1.4	22.37±1.1	22.76±1.2	22.37±1.4	22.30±1.7	23.28±1.4
Twonorm	5.78±0.8	4.30±0.8	5.75±0.7	4.42±0.9	5.72±1.0	4.41±0.7	9.65±1.0	7.21±0.5
Waveform	16.67±1.5	13.40±0.7	16.43±0.7	14.65±0.8	16.12±1.2	13.47±0.6	14.98±0.8	14.77±1.5
Average rank	5.2692	3.6538	4.8077	4.5769	4.5000	3.8462	4.8462	4.5000
p-value	0.7489							

Table 7.2: Average classification errors and standard deviations for AdaBoost and rBoost at 10% symmetric noise together with Friedman test at the 5% level.

Dataset	AdaBoost		rBoost-Fixed gamma		rBoost		Gentle	Modest
	LR	rLR	LR	rLR	LR	rLR	Boost	Boost
Banana	18.71±3.1	14.73±3.0	18.14±1.7	14.47±2.1	18.05±1.6	14.94±2.7	20.62±1.6	24.69±2.5
B.Cancer	35.58±3.4	39.22±5.1	34.94±6.5	32.99±4.8	35.58±6.2	39.48±5.5	39.09±7.7	33.38±4.6
Diabetes	25.37±2.3	27.47±1.9	24.90±2.3	29.57±2.4	25.23±2.7	28.57±2.3	30.60±2.9	26.67±2.3
German	30.67±1.8	32.60±3.5	31.37±2.9	29.50±2.9	30.33±3.1	34.73±3.4	35.37±4.0	30.93±3.1
Heart	22.10±5.7	21.50±4.0	22.70±4.0	22.60±6.5	22.90±4.9	21.90±4.3	30.00±5.5	24.80±3.7
Image	14.67±1.4	6.94±1.0	15.23±1.0	6.67±1.0	14.30±0.9	6.52±0.9	7.51±1.0	20.10±4.4
Ringnorm	33.54±1.1	33.76±2.4	37.58±2.5	45.75±1.5	33.78±1.7	33.71±1.7	28.46±1.8	18.39±2.2
S.Flare	35.65±1.9	35.50±2.3	47.33±3.7	47.47±4.8	35.00±1.9	35.48±2.6	36.48±2.3	35.00±3.9
Splice	27.53±2.0	28.65±1.6	35.05±1.4	39.01±1.4	27.07±2.2	28.23±2.2	27.09±2.5	13.77±2.6
Thyroid	16.27±3.1	24.53±6.9	21.20±7.0	22.13±5.7	18.80±4.6	24.53±6.9	24.40±7.4	17.60±7.2
Titanic	23.12±1.6	23.01±1.8	23.27±1.5	22.92±1.4	23.09±1.4	23.11±1.8	22.80±1.9	23.41±1.3
Twonorm	8.53±1.1	6.67±0.9	8.77±1.0	6.87±1.3	8.63±1.0	6.60±1.1	16.06±2.0	8.84±0.9
Waveform	21.02±2.1	16.88±1.8	20.80±2.4	18.16±2.0	20.41±2.2	16.96±1.6	20.26±2.2	15.57±0.9
Average rank	4.3462	4.1154	5.4615	4.1538	3.9231	4.2692	5.5385	4.1923
p-value	0.5363							

Table 7.3: Average classification errors and standard deviations for AdaBoost and rBoost at 30% symmetric noise together with Friedman test at the 5% level.

7.4 Summary

We presented a robust boosting algorithm based on the famous AdaBoost algorithm called rBoost. The rBoost contains good properties namely its objective is non-convex and it has label noise parameters that can be estimated efficiently using the proposed multiplicative update rules. The new algorithm is also appealing since it requires only a minor modification to the existing AdaBoost algorithm. We further demonstrated that the label noise parameters can be more accurately estimated by using a trusted validation

Dataset	AdaBoost		rBoost-Fixed gamma		rBoost		Gentle	Modest
	LR	rLR	LR	rLR	LR	rLR	Boost	Boost
Banana	17.85±2.7	13.54±1.3	16.78±1.7	12.55±1.1	17.81±2.4	13.65±1.2	16.81±1.5	22.27±2.8
B.Cancer	31.82±5.5	32.60±5.9	30.26±4.1	29.09±4.2	33.12±5.3	34.29±6.4	33.51±8.0	30.13±4.9
Diabetes	24.70±1.6	25.67±1.6	24.27±1.7	25.57±1.5	24.23±1.6	25.97±1.7	27.80±2.1	24.93±1.6
German	25.63±2.8	26.57±1.8	24.77±2.5	24.53±2.5	25.40±2.5	26.50±3.4	26.67±2.5	27.10±2.0
Heart	21.70±4.4	21.50±3.9	21.60±3.7	22.20±2.9	21.30±3.7	21.10±2.7	26.10±3.2	21.70±4.2
Image	15.88±1.0	4.52±1.0	15.20±1.6	4.06±0.9	15.15±1.5	4.40±1.2	4.45±0.7	24.12±1.9
Ringnorm	27.48±1.4	25.66±0.7	26.26±1.2	29.57±1.8	27.29±1.4	26.00±0.4	11.63±1.1	12.28±1.4
S.Flare	35.03±1.4	34.85±1.0	34.52±1.3	34.67±2.5	35.43±1.5	35.05±1.5	35.90±1.4	46.37±15.4
Splice	18.29±1.2	15.82±0.8	17.94±0.6	16.42±0.9	17.89±0.9	15.55±0.9	10.07±0.9	6.91±1.0
Thyroid	12.00±4.4	9.73±2.5	11.73±3.2	9.60±3.3	12.93±3.8	10.40±3.3	10.13±3.6	7.73±2.6
Titanic	22.87±1.3	23.06±1.3	22.65±1.3	22.23±1.1	23.58±1.6	22.44±1.0	22.83±1.5	23.63±1.5
Twonorm	7.02±1.6	5.21±1.1	6.16±1.4	4.51±0.8	6.56±1.4	5.34±1.1	9.19±1.0	7.71±1.2
Waveform	18.10±1.3	14.48±1.1	17.83±1.2	16.23±1.5	17.71±1.2	14.54±1.4	16.52±1.0	14.19±0.7
Average rank	5.8077	3.7692	4.1538	3.0000	5.1538	3.9231	5.3077	4.8846
p-value	0.0631							

Table 7.4: Average classification errors and standard deviations for AdaBoost and rBoost at 10% asymmetric noise together with Friedman test at the 5% level.

Dataset	AdaBoost		rBoost-Fixed gamma		rBoost		Gentle	Modest
	LR	rLR	LR	rLR	LR	rLR	Boost	Boost
Banana	31.45±5.2	23.53±4.7	27.31±4.5	14.27±1.0	32.39±3.9	23.83±4.1	25.38±2.7	33.04±6.8
B.Cancer	39.22±4.9	40.26±6.3	35.06±5.6	30.52±3.7	40.91±7.2	41.82±7.4	42.86±5.9	37.40±7.7
Diabetes	32.20±2.1	33.43±3.9	29.47±3.0	30.20±2.6	32.80±3.3	33.27±3.1	38.37±3.6	32.07±3.5
German	36.23±3.2	35.27±3.3	31.30±3.2	27.67±2.5	34.53±3.6	33.93±3.5	36.17±3.6	35.83±4.6
Heart	27.60±5.5	27.30±6.8	23.00±4.3	24.30±3.8	28.20±6.3	28.00±6.9	32.00±7.2	29.60±11.7
Image	22.48±1.6	10.70±0.9	16.96±1.8	5.47 ±1.0	20.53±1.6	9.82 ±1.5	11.94±1.1	26.44±1.3
Ringnorm	34.71±2.5	26.80±1.1	30.64±1.4	38.37±2.6	33.92±2.3	27.12±0.8	21.33±2.3	47.48±10.3
S.Flare	37.48±3.9	37.25±4.3	35.80±1.9	34.92±1.3	37.00±4.3	38.22±5.0	37.85±4.2	67.88±5.3
Splice	26.15±2.2	24.50±2.0	20.88±1.3	20.65±1.7	25.30±1.8	24.25±2.5	20.15±1.5	17.56±2.5
Thyroid	23.07±6.6	16.93±6.9	14.80±3.7	13.07±5.0	21.47±6.2	15.20±6.5	24.93±6.0	23.73±17.2
Titanic	32.60±8.4	31.21±8.7	23.88±1.8	22.14±1.5	33.17±9.3	30.73±8.7	32.94±9.0	33.49±13.7
Twonorm	16.02±2.4	12.07±2.0	8.89 ±1.5	6.51 ±1.3	14.68±2.3	12.19±2.0	17.85±1.8	16.62±3.1
Waveform	28.83±2.8	23.43±2.5	24.27±2.1	19.95±1.6	28.39±3.1	23.02±2.5	27.31±2.5	21.10±2.2
Average rank	5.9231	4.0000	2.8462	1.7692	5.6154	4.1538	5.8462	5.8462
p-value	3.2e-6							
Nemenyi CD	2.5845							

Table 7.5: Average classification errors and standard deviations for AdaBoost and rBoost at 30% asymmetric noise together with Friedman test at the 5% level.

set for Platt's calibration algorithm as a form of extra information. It shows good result close to the rBoost with label noise parameters fixed to the true values. In addition, we have empirically shown that simply employing a robust classifier as a base learner in AdaBoost does not help to alleviate the bad effect of label noise. However, rather interestingly, its effect is to speed up the boosting process. This could be advantageous in cases of low noise.

From the empirical results we also see that as the complexity of a classifier increases

its ability to estimate the label flipping probabilities is impaired. Eventually the classifier won't be able to distinguish between good and noisy training labels. Rather the classifier will simply learn to fit all the labels without questioning theirs correctness, unless there is extra information available.

CHAPTER 8

Conclusion and Outlook

The thesis presents approaches to incorporate a probabilistic label noise model into the classical probabilistic classifiers and investigates their performance empirically and theoretically.

We presented a multi-class robust normal discriminant analysis classifier (rNDA) in Chapter 3, the robust Logistic Regression (rLR) and robust multinomial logistic regression (rmLR) in Chapter 4, and the robust Bayesian Logistic Regression in Chapter 5. Empirical results on both synthetic datasets with artificial label noise and real-world datasets which genuinely contain label errors suggest that the robust classifiers outperform the classical approaches in terms of classification accuracy. The theoretical analysis also confirms that the new robust classifiers have an ability to obtain more accurate parameter estimates in noisy situations.

To broaden the applicability of our robust models, we presented the robust Kernel Logistic Regression (rkLR) and its multi-class extension in Chapter 6. The challenge is how to perform model selection in noisy label environment where the use of standard cross-validation technique is sub-optimal. The proposed solution adopts a multiple kernel learning framework to select good kernel parameters and the makes use of Bayesian reg-

ularisation technique to select good regularisation coefficients. This new approach shows superior performance in terms of classification accuracy and is significantly faster than cross-validation.

In addition to studying a single robust classifier, we discussed in Chapter 7 an ensemble of robust classifiers by extending AdaBoost at two different levels. At the lower level we studied the robust committee where robust classifiers are combined and boosted using existing AdaBoost algorithm. At the higher level, we proposed a new robust boosting algorithm utilising a more robust objective function. We showed that a committee of robust classifiers, although converging faster, is still susceptible to noise. Meanwhile, the new robust boosting is able to deal with label noise. Due to the unbounded complexity of the robust boosting, the robust boosting algorithm is not entirely immune to label noise and could eventually fail as its complexity increases, unless some extra information is provided. This finding suggests that the success of probabilistic noise modelling depends on the complexity of target classifier. Relating this finding back to the models in Chapters 3-6, we can understand that the positive results observed are because those classifiers are relatively constrained.

Throughout the thesis, we studied *random misclassification noise* which assumes that mislabelling arises randomly and independently from the observation features. This is different from a non-random noise where label flipping depends on the content of the observation. The difference in the nature of label noises suggests that each type of noise needs a separate treatment using an appropriate model. Having said that, however, we found that the random noise model works pretty well even in the situation where the randomness assumption does not strictly hold true – for example in our bike classification problem where the labeller might have used textual information around an image to determine the label of the image. Another example is microarray classification where it is very likely that the labelling process is not entirely independent of the feature content, and

yet the random noise model has been successfully applied. In such situations the random label noise model may be understood as a simplifying model assumption which trades some suboptimality for tractability to estimate the key parameters from limited amounts of data. These results demonstrate indeed that the random label noise assumption is useful in practice despite its simplicity even when there exist some dependencies between the label flipping and the input. Nonetheless, developing more specialised noise models for non-random class noise is an interesting avenue for future work.

APPENDIX A

Derivation details

A.1 Derivation of the robust boosting

This section shows derivation details of the rBoost algorithm. The loss of the rBoost is defined as:

$$\begin{aligned}\mathcal{L}(H) = &\sum_{n=1}^{N} \mathbb{1}(\tilde{y}_n = 1)\Big\{\gamma_{00} e^{-H(\mathbf{x}_n)} + \gamma_{01} e^{H(\mathbf{x}_n)}\Big\} \\ &+ \mathbb{1}(\tilde{y}_n = -1)\Big\{\gamma_{11} e^{H(\mathbf{x}_n)} + \gamma_{10} e^{-H(\mathbf{x}_n)}\Big\} \end{aligned} \tag{1}$$

Here, γ_{jk} are again probabilistic factors representing uncertainty in labels. We write out the form of $H(\mathbf{x}_n)$ for the next round of AdaBoost. Minimising this loss of eq. (1) in a step-wise manner is then equivalent to minimising the following:

$$\begin{aligned}&\arg\min_{h,\alpha} \\ &\Big(\sum_{n=1}^{N} \mathbb{1}(\tilde{y}_n = 1)\Big\{\gamma_{00} e^{-(H(\mathbf{x}_n)+\alpha h(\mathbf{x}_n))} + \gamma_{01} e^{H(\mathbf{x}_n)+\alpha h(\mathbf{x}_n)}\Big\} \\ &+ \mathbb{1}(\tilde{y}_n = -1)\Big\{\gamma_{11} e^{H(\mathbf{x}_n)+\alpha h(\mathbf{x}_n)} + \gamma_{10} e^{-(H(\mathbf{x}_n)+\alpha h(\mathbf{x}_n))}\Big\}\Big) \end{aligned} \tag{2}$$

$$\begin{aligned} = \arg\min_{h,\alpha} \sum_{n=1}^{N} \Big\{ &\mathbb{1}(\tilde{y}_n = 1)\gamma_{00} e^{-H(\mathbf{x}_n)} e^{-\alpha h(\mathbf{x}_n)} && (3)\\ &+ \mathbb{1}(\tilde{y}_n = 1)\gamma_{01} e^{H(\mathbf{x}_n)} e^{\alpha h(\mathbf{x}_n)} && (4)\\ &+ \mathbb{1}(\tilde{y}_n = -1)\gamma_{11} e^{H(\mathbf{x}_n)} e^{\alpha h(\mathbf{x}_n)} && (5)\\ &+ \mathbb{1}(\tilde{y}_n = -1)\gamma_{10} e^{-H(\mathbf{x}_n)} e^{-\alpha h(\mathbf{x}_n)} \Big\} && (6) \end{aligned}$$

Now consider each term in the sum.

$$
\begin{aligned}
(3) = & \sum_{i|h(\mathbf{x}_n)=\tilde{y}_n} \mathbb{1}(\tilde{y}_n = 1)\gamma_{00} e^{-H(\mathbf{x}_n)} e^{-\alpha} \\
& + \sum_{i|h(\mathbf{x}_n)\neq\tilde{y}_n} \mathbb{1}(\tilde{y}_n = 1)\gamma_{00} e^{-H(\mathbf{x}_n)} e^{\alpha} \\
= & \sum_{n=1}^{N} \Big(1 - \mathbb{1}(h(\mathbf{x}_n) \neq \tilde{y}_n)\Big) \mathbb{1}(\tilde{y}_n = 1)\gamma_{00} e^{-H(\mathbf{x}_n)} e^{-\alpha} \\
& + \sum_{i|h(\mathbf{x}_n)\neq\tilde{y}_n} \mathbb{1}(\tilde{y}_n = 1)\gamma_{00} e^{-H(\mathbf{x}_n)} e^{\alpha} \\
= & \sum_{n=1}^{N} \mathbb{1}(\tilde{y}_n = 1)\gamma_{00} e^{-H(\mathbf{x}_n)} e^{-\alpha} \\
& - \sum_{n=1}^{N} \mathbb{1}(\tilde{y}_n = 1)\mathbb{1}(h(\mathbf{x}_n) \neq \tilde{y}_n)\gamma_{00} e^{-H(\mathbf{x}_n)} e^{-\alpha} \\
& + \sum_{n=1}^{N} \mathbb{1}(\tilde{y}_n = 1)\mathbb{1}(h(\mathbf{x}_n) \neq \tilde{y}_n)\gamma_{00} e^{-H(\mathbf{x}_n)} e^{\alpha} \\
= & \Big(e^{\alpha} - e^{-\alpha}\Big) \sum_{n=1}^{N} \mathbb{1}(\tilde{y}_n = 1)\mathbb{1}(h(\mathbf{x}_n) \neq \tilde{y}_n)\gamma_{00} e^{-H(\mathbf{x}_n)} \\
& + \sum_{n=1}^{N} \mathbb{1}(\tilde{y}_n = 1)\gamma_{00} e^{-H(\mathbf{x}_n)} e^{-\alpha} \qquad (7)
\end{aligned}
$$

Using similar substitution and grouping, we also have the following:

$$
\begin{aligned}
(4) = & \Big(e^{-\alpha} - e^{\alpha}\Big) \sum_{n=1}^{N} \mathbb{1}(\tilde{y}_n = 1)\mathbb{1}(h(\mathbf{x}_n) \neq \tilde{y}_n)\gamma_{01} e^{H(\mathbf{x}_n)} \\
& + \sum_{n=1}^{N} \mathbb{1}(\tilde{y}_n = 1)\gamma_{01} e^{H(\mathbf{x}_n)} e^{\alpha} \qquad (8) \\
(5) = & \Big(e^{\alpha} - e^{-\alpha}\Big) \sum_{n=1}^{N} \mathbb{1}(\tilde{y}_n = -1)\mathbb{1}(h(\mathbf{x}_n) \neq \tilde{y}_n)\gamma_{11} e^{H(\mathbf{x}_n)} \\
& + \sum_{n=1}^{N} \mathbb{1}(\tilde{y}_n = -1)\gamma_{11} e^{H(\mathbf{x}_n)} e^{-\alpha} \qquad (9) \\
(6) = & \Big(e^{-\alpha} - e^{\alpha}\Big) \sum_{n=1}^{N} \mathbb{1}(\tilde{y}_n = -1)\mathbb{1}(h(\mathbf{x}_n) \neq \tilde{y}_n)\gamma_{10} e^{-H(\mathbf{x}_n)}
\end{aligned}
$$

$$+\sum_{n=1}^{N}\mathbb{1}(\tilde{y}_n=-1)\gamma_{10}e^{-H(\mathbf{x}_n)}e^{\alpha} \tag{10}$$

Summing all four expressions we have the objective:

$$\begin{aligned}
&\arg\min_{h,\alpha}(e^{\alpha}-e^{-\alpha})\sum_{n=1}^{N}\Big\{\mathbb{1}(\tilde{y}_n=1)\gamma_{00}e^{-H(\mathbf{x}_n)}\mathbb{1}(h(\mathbf{x}_n)\neq\tilde{y}_n)\\
&\quad+\mathbb{1}(\tilde{y}_n=-1)\gamma_{11}e^{H(\mathbf{x}_n)}\mathbb{1}(h(\mathbf{x}_n)\neq\tilde{y}_n)\Big\}\\
&+e^{-\alpha}\sum_{n=1}^{N}\Big\{\mathbb{1}(\tilde{y}_n=1)\gamma_{00}e^{-H(\mathbf{x}_n)}+\mathbb{1}(\tilde{y}_n=-1)\gamma_{11}e^{H(\mathbf{x}_n)}\Big\}\\
&-(e^{\alpha}-e^{-\alpha})\sum_{n=1}^{N}\Big\{\mathbb{1}(\tilde{y}_n=1)\gamma_{01}e^{H(\mathbf{x}_n)}\mathbb{1}(h(\mathbf{x}_n)\neq\tilde{y}_n)\\
&\quad+\mathbb{1}(\tilde{y}_n=-1)\gamma_{10}e^{-H(\mathbf{x}_n)}\mathbb{1}(h(\mathbf{x}_n)\neq\tilde{y}_n)\Big\}\\
&+e^{\alpha}\sum_{n=1}^{N}\Big\{\mathbb{1}(\tilde{y}_n=1)\gamma_{01}e^{H(\mathbf{x}_n)}+\mathbb{1}(\tilde{y}_n=-1)\gamma_{10}e^{-H(\mathbf{x}_n)}\Big\} \qquad (11)\\
&=\arg\min_{h,\alpha}2\sinh(\alpha)\times\\
&\sum_{n=1}^{N}\Big\{\mathbb{1}(\tilde{y}_n=1)\mathbb{1}(h(\mathbf{x}_n)\neq\tilde{y}_n)[\gamma_{00}e^{-H(\mathbf{x}_n)}-\gamma_{01}e^{H(\mathbf{x}_n)}]\Big\}\\
&+2\sinh(\alpha)\times\\
&\sum_{n=1}^{N}\Big\{\mathbb{1}(\tilde{y}_n=-1)\mathbb{1}(h(\mathbf{x}_n)\neq\tilde{y}_n)[\gamma_{11}e^{H(\mathbf{x}_n)}-\gamma_{10}e^{-H(\mathbf{x}_n)}]\Big\}\\
&+e^{-\alpha}\sum_{n=1}^{N}\Big\{\mathbb{1}(\tilde{y}_n=1)\gamma_{00}e^{-H(\mathbf{x}_n)}+\mathbb{1}(\tilde{y}_n=-1)\gamma_{11}e^{H(\mathbf{x}_n)}\Big\}\\
&+e^{\alpha}\sum_{n=1}^{N}\Big\{\mathbb{1}(\tilde{y}_n=1)\gamma_{01}e^{H(\mathbf{x}_n)}+\mathbb{1}(\tilde{y}_n=-1)\gamma_{10}e^{-H(\mathbf{x}_n)}\Big\} \qquad (12)
\end{aligned}$$

Define $w_{00}=\gamma_{00}e^{-H(\mathbf{x}_n)}$, $w_{01}=\gamma_{01}e^{H(\mathbf{x}_n)}$, $w_{11}=\gamma_{11}e^{H(\mathbf{x}_n)}$ and $w_{10}=\gamma_{10}e^{-H(\mathbf{x}_n)}$, we simplify the objective into.

$$\begin{aligned}
\arg\min_{h,\alpha}\ \ &2\sinh(\alpha)\sum_{n=1}^{N}\Big\{w_n\mathbb{1}(h(\mathbf{x}_n)\neq\tilde{y}_n)\Big\}\\
&+e^{-\alpha}\sum_{n=1}^{N}\Big\{\mathbb{1}(\tilde{y}_n=1)w_{00}+\mathbb{1}(\tilde{y}_n=-1)w_{11}\Big\}
\end{aligned}$$

$$+ e^{\alpha} \sum_{n=1}^{N} \Big\{ \mathbb{1}(\tilde{y}_n = 1) w_{01} + \mathbb{1}(\tilde{y}_n = -1) w_{10} \Big\} \tag{13}$$

where

$$w_n = \begin{cases} (w_{00} - w_{01}), & \text{if } \tilde{y}_n = +1. \\ (w_{11} - w_{10}), & \text{if } \tilde{y}_n = -1. \end{cases} \tag{14}$$

APPENDIX B

Probability background

B.1 Hoeffding's inequality

Let $Z_1, ...Z_m$ be a sequence of i.i.d random variables and assume that $\mathbb{E}[Z_1] = \mu$ and $p[a \leq Z_1 \leq b] = 1$. Then for any $\epsilon > 0$

$$p[|\frac{1}{m}\sum_{i=1}^{M} Z_i - \mu| > \epsilon] \leq 2\exp\left(\frac{-2m\epsilon^2}{(b-a)^2}\right) \tag{1}$$

B.2 Jensen's inequality

If g is a convex function then

$$\mathbb{E}g(x) \geq g(\mathbb{E}(x)) \tag{2}$$

If g is a concave function then

$$\mathbb{E}g(x) \leq g(\mathbb{E}(x)) \tag{3}$$

B.3 Boole's inequality (union bound)

For any two sets of event A, B we have

$$p[A \cup B] \leq p[A] + p[B] \tag{4}$$

B.4 Markov's inequality

Let Z be a non-negative random variable and $\forall a > 0$, we have

$$p[Z \geq a] \leq \frac{\mathbb{E}[Z]}{a} \tag{5}$$

List of References

Uri Alon, Naama Barkai, Daniel A. Notterman, Kurt Gish, Suzanne Ybarra, Douglas G. Mack, and Arnold J. Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Sciences of the United States of America*, 96(12):6745–6750, 1999. 3 citations in sections 5.5.2, 5.5.4, and 5.8.

Massih R. Amini and Patrick Gallinari. Semi-supervised learning with an imperfect supervisor. *Knowledge and Information Systems*, 8(4):385–413, 2005. One citation in section 2.3.2.

Dana Angluin and Philip Laird. Learning from noisy examples. *Machine Learning*, 2: 343–370, 1988. 2 citations in sections 2.4 and 2.4.

Sylvain Arlot and Alain Celisse. A survey of cross-validation procedures for model selection. July 2009. `http://arxiv.org/abs/0907.4728`. One citation in section 1.3.2.

Ricardo Barandela and Eduardo Gasca. Decontamination of training samples for supervised pattern recognition methods. In *Advances in Pattern Recognition*, volume 1876 of *Lecture Notes in Computer Science*, pages 621–630. Springer, 2000. 6 citations in sections 1.2, 2.3.1, 3, 3.4.2, 4.5.3, and 7.1.

Peter L. Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: risk

bounds and structural results. *Journal of Machine Learning Research*, 3:463–482, March 2003. 2 citations in sections 1.1 and 4.4.5.

Vladimir Batagelj and Andrej Mrvar. PAJEK – Program for large network analysis, 1998. http://pajek.imfm.si/. One citation in section 4.5.4.

Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is "nearest neighbor" meaningful? In *Proceeding of the International Conference on Database Theory*, pages 217–235, 1999. One citation in section 2.3.1.

Yingtao Bi and Daniel R. Jeske. The efficiency of logistic regression compared to normal discriminant analysis under class-conditional classification noise. *Journal of Multivarariate Analysis*, 101(7):1622–1637, 2010. 2 citations in sections 1.3 and 2.2.

Battista Biggio, Blaine Nelson, and Pavel Laskov. Support Vector Machines Under Adversarial Label Noise. *Journal of Machine Learning Research - Proceedings Track*, 20: 97–112, 2011. One citation in section 2.1.

Dankmar Böhning. Multinomial logistic regression algorithm. *Annals of the Institute of Statistical Mathematics*, 44:197–200, 1992. One citation in section 4.3.

Jakramate Bootkrajang and Ata Kabán. Multi-class classification in the presence of labelling errors. In *Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, (ESANN'11)*, pages 345–350, 2011. No citations.

Jakramate Bootkrajang and Ata Kabán. Label-noise robust logistic regression and its applications. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, (ECML-PKDD'12)*, pages 143–158, 2012. No citations.

Jakramate Bootkrajang and Ata Kabán. Boosting in the presence of labelling errors. In *Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence, (UAI'13)*, 2013a. No citations.

Jakramate Bootkrajang and Ata Kabán. Classification of mislabelled microarrays using robust sparse logistic regression. *Bioinformatics*, 29(7):870–877, 2013b. No citations.

Jakramate Bootkrajang and Ata Kabán. Learning kernel logistic regression in the presence of class label noise. *Pattern Recognition*, 2013c. revised and resubmitted. No citations.

Charles Bouveyron and Stephane Girard. Robust supervised classification with mixture models: Learning from data with uncertain labels. *Pattern Recognition*, 42(11):2649–2658, 2009. One citation in section 6.

Carla E. Brodley and Mark A. Friedl. Identifying and eliminating mislabeled training instances. In *Proceedings of the 13th National Conference on Artificial Intelligence, (AAAI'96)*, pages 799–805, 1996. One citation in section 2.3.1.

Carla E. Brodley and Mark A. Friedl. Identifying mislabeled training data. *Journal of Artificial Intelligence Research*, 11:131–167, 1999. One citation in section 1.2.

Gavin C. Cawley and Nicola L. Talbot. Efficient approximate leave-one-out cross-validation for kernel logistic regression. *Machine Learning*, 71(2-3):243–264, June 2008. One citation in section 6.2.3.

Gavin C. Cawley and Nicola L. C. Talbot. Gene selection in cancer classification using sparse logistic regression with bayesian regularization. *Bioinformatics*, 22:2348–2355, 2006. 5 citations in sections 5, 5.1.1, 5.1.1, 5.2.1, and 5.5.1.

Gavin C. Cawley and Nicola L. C. Talbot. Preventing over-fitting during model selec-

tion via bayesian regularisation of the hyper-parameters. *Journal of Machine Learning Research*, 8:841–861, 2007. One citation in section 6.

Gavin C. Cawley and Nicola L.C. Talbot. On over-fitting in model selection and subsequent selection bias in performance evaluation. *Journal of Machine Learning Research*, 99:2079–2107, August 2010. One citation in section 6.2.1.

Gilles Celeux, Stéphane Chrétien, Florence Forbes, and Abdallah Mkhadri. A component-wise EM algorithm for mixtures. *Journal of Computational and Graphical Statistics*, 10(4):697–712, 2001. One citation in section 4.4.2.

Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. `http://www.csie.ntu.edu.tw/~cjlin/libsvm`. One citation in section 2.

Raj S. Chhikara and Jim McKeon. Linear Discriminant Analysis with Misallocation in Training Samples. *Journal of the American Statistical Association*, 79(388):899–906, 1984. 2 citations in sections 1.3 and 2.2.

C. B. Chittineni. Maximum likelihood estimation of label imperfection probabilities and its use in the identification of mislabeled patterns. *IEEE Transactions on Geoscience and Remote Sensing*, 20:99–111, 1982. One citation in section 2.3.2.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20: 273–297, 1995. One citation in section 4.5.3.

Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, (CVPR'05)*, volume 1, pages 886–893, june 2005. One citation in section 6.2.5.

Theodoros Damoulas and Mark A. Girolami. Pattern recognition with a bayesian kernel combination machine. *Pattern Recognition Letters*, 30(1):46–54, 2009. One citation in section 6.1.1.

Sanjoy Dasgupta. Learning mixtures of gaussians. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science, (FOCS'99)*, pages 634–644, 1999. 2 citations in sections 3.4.1 and 3.4.2.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977. One citation in section 3.2.

Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995. One citation in section 2.3.1.

Robert J. Durrant and Ata Kabán. Compressed fisher linear discriminant analysis: classification of randomly projected data. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data mining, (KDD'10)*, pages 1119–1128, 2010. One citation in section 3.5.

Bradley Efron. The Efficiency of Logistic Regression Compared to Normal Discriminant Analysis. *Journal of the American Statistical Association*, 70(352):892–898, 1975. One citation in section 2.2.

Wei Fan and Salvatore J. Stolfo. AdaCost: misclassification cost-sensitive boosting. In *Proceedings of the 16th International Conference on Machine Learning, (ICML'99)*, pages 97–105, 1999. One citation in section 7.2.4.

Ronald A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(7):179–188, 1936. One citation in section 3.4.1.

Andrew J. Frank and Arthur Asuncion. UCI Machine Learning Repository, 2010. http://archive.ics.uci.edu/ml. One citation in section 3.4.1.

Benoît Frénay, Gael de Lannoy, and Michel Verleysen. Label noise-tolerant hidden markov models for segmentation: Application to ECGs. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, (ECML-PKDD'11)*, pages 455–470, 2011. One citation in section 2.3.2.

Yoav Freund. A more robust boosting algorithm. Technical Report arXiv:0905.2138, Department of Computer Science and Engineering, University of California, San Diego, 2009. One citation in section 2.3.2.

Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the 2nd European Conference on Computational Learning Theory, (EuroCOLT '95)*, pages 23–37, 1995. One citation in section 2.2.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28, 1998. 3 citations in sections 2.2, 2.3.2, and 7.2.4.

Terrence S. Furey, Nello Cristianini, Nigel Duffy, David W. Bednarski, Michèl Schummer, and David Haussler. Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, pages 906–914, 2000. One citation in section 5.8.

Claudio Gentile and David P. Helmbold. Improved lower bounds for learning from noisy examples: An information-theoretic approach. *Information and Computation*, 166(2): 133–155, 2001. One citation in section 2.4.

Todd R. Golub, Donna K. Slonim, Pablo Tamayo, Christine Huard, Michelle Gaasenbeek, Jill P. Mesirov, Hilary Coller, Mignon L. Loh, James R. Downing, Mark A. Caligiuri, Clara D. Bloomfield, and Eric S. Lander. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999. 2 citations in sections 5.5.2 and 5.5.4.

Mehmet Gönen and Ethem Alpaydin. Multiple kernel learning algorithms. *Journal of Machine Learning Research*, 12:2211–2268, July 2011. One citation in section 1.

Jerry A. Hausman, Jason Abrevaya, and Fiona M. Scott-Morton. Misclassification of the dependent variable in a discrete-response setting. *Journal of Econometrics*, 87(2): 239–269, 1998. One citation in section 2.3.2.

Daniel Hernández-Lobato, José Miguel Hernández-Lobato, and Pierre Dupont. Robust multi-class gaussian process classification. In *Proceedings of The Conference on Neural Information Processing Systems, (NIPS'11)*, pages 280–288, 2011. One citation in section 7.1.

Magnus R. Hestenes and Eduard Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(6):409–436, 1952. One citation in section 4.4.1.

Thomas Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 42(1-2):177–196, 2001. One citation in section 2.3.2.

Yuan Jiang and Zhi-Hua Zhou. Editing training data for k-nn classifiers with neural network ensemble. In *Advances in Neural Networks*, volume 3173 of *Lecture Notes in Computer Science*, pages 356–361. 2004. 2 citations in sections 1.2 and 2.3.1.

George H. John. Robust decision trees: Removing outliers from databases. In *Proceedings*

of the Conference on Knowledge Discovery and Data Mining, (KDD'95), pages 174–179, 1995. One citation in section 2.3.1.

Ata Kabán, Peter Tino, and Mark Girolami. A general framework for a principled hierarchical visualization of multivariate data. In *Proceedings of the 3rd International Conference on Intelligent Data Engineering and Automated Learning, (IDEAL'02)*, pages 518–523, 2002. One citation in section 1.

Koji Kadota, Daisuke Tominaga, Yutaka Akiyama, and Katsutoshi Takahashi. Detecting outlying samples in microarray data: A critical assessment of the effect of outliers on sample classification. *Chem-Bio Informatics Journal*, 3(1):30–45, 2003. One citation in section 5.8.

Amitava Karmaker and Stephen Kwek. A boosting approach to remove class label noise. *International Journal of Hybrid Intelligent Systems*, 3(3):169–177, August 2006. One citation in section 2.3.2.

Michael Kearns and Ming Li. Learning in the presence of malicious errors. In *Proceedings of the 20th annual ACM Symposium on Theory of Computing, (STOC '88)*, pages 267–280, 1988. 2 citations in sections 2.4 and 2.4.

Michael Kearns and Leslie Valiant. Cryptographic limitations on learning Boolean formulae and finite automata. *Journal of the ACM*, 41(1):67–95, January 1994. One citation in section 2.2.

George S. Kimeldorf and Grace Wahba. Some results on Tchebycheffian spline functions. *Journal of Mathematical Analysis and Applications*, 33(1):82–95, 1971. One citation in section 6.1.

Abba Krieger, Chuan Long, and Abraham Wyner. Boosting noisy data. In *Proceedings*

of the 18th International Conference on Machine Learning, (ICML'01), pages 274–281, 2001. One citation in section 2.3.2.

Thriyambakam Krishnan and Subhas C. Nandy. Efficiency of discriminant analysis when initial samples are classified stochastically. *Pattern Recognition*, 23(5):529–537, 1990. One citation in section 1.2.

Anastasia Krithara, Massih R. Amini, Jean-Michel Renders, and Cyril Goutte. Semi-supervised document classification with a mislabeling error model. In *Proceedings of the IR research, 30th European conference on Advances in Information Retrieval, (ECIR'08)*, pages 370–381, 2008. One citation in section 2.3.2.

Peter A. Lachenbruch. Discriminant analysis when the initial samples are misclassified. *Technometrics*, 8(4):657–662, 1966. 2 citations in sections 2.1 and 2.2.

Peter A. Lachenbruch. Discriminant analysis when the initial samples are misclassified ii: Non-random misclassification models. *Technometrics*, 16(3):pp. 419–424, 1974. 2 citations in sections 1.3 and 2.2.

Gert R.G. Lanckriet, Tijl De Bie, Nello Cristianini, Michael I. Jordan, and William S. Noble. A statistical framework for genomic data fusion. *Bioinformatics*, 20(16):2626–2635, 2004. One citation in section 6.1.1.

Neil D. Lawrence and Bernhard Schölkopf. Estimating a kernel Fisher discriminant in the presence of label noise. In *Proceedings of the 18th International Conference on Machine Learning, (ICML'01)*, pages 306–313, 2001. 9 citations in sections 1.2, 1.3.1, 1.3.2, 1.5, 2.3.2, 3.1, 4.5.3, 6, and 6.2.

Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In *Proceedings of The Conference on Neural Information Processing Systems, (NIPS'01)*, pages 556–562, 2001. 2 citations in sections 4.4.1 and 4.4.1.

Leping. Li, Thomas A. Darden, Clarice R. Weingberg, Arnold J. Levine, and Lee G. Pedersen. Gene assessment and sample classification for gene expression data using a genetic algorithm / k-nearest neighbor method. *Combinatorial Chemistry and High Throughput Screening*, pages 727–739, 2001. One citation in section 5.8.

Yunlei Li, Lodewyk F.A. Wessels, Dick de Ridder, and Marcel J.T. Reinders. Classification in the presence of class noise using a probabilistic kernel Fisher method. *Pattern Recognition*, 40(12):3349–3357, 2007. 4 citations in sections 1.2, 1.3.2, 2.3.2, and 6.2.

Philip M. Long and Rocco A. Servedio. Random classification noise defeats all convex potential boosters. *Machine Learning*, 78(3):287–304, March 2010. 2 citations in sections 2.2 and 2.3.2.

David G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision, (ICCV'99)*, pages 1150–1157, 1999. 2 citations in sections 4.5.4 and 6.2.5.

Gábor Lugosi. Learning with an unreliable teacher. *Pattern Recognition*, 25:79–87, 1992. 3 citations in sections 2.1, 2.4, and 5.5.1.

David J.C. MacKay. Probable networks and plausible predictions - a review of practical Bayesian methods for supervised neural networks. *Network: Computation in Neural Systems*, 6:469–505, 1995. One citation in section 5.1.1.

Laurence S. Magder and James P. Hughes. Logistic regression when the outcome is measured with uncertainty. *American Journal of Epidemiology*, 146(2):195–203, 1997. One citation in section 2.3.2.

Jonathan I. Maletic and Andrian Marcus. Data cleansing: Beyond integrity analysis. In *Proceedings of the International Conference on Information Quality, (ICIQ'00)*, pages 200–209, 2000. One citation in section 1.2.

Andrea Malossini, Enrico Blanzieri, and Raymond T. Ng. Detecting potential labeling errors in microarrays by data perturbation. *Bioinformatics*, 22(17):2114–2121, 2006. 4 citations in sections 1.2, 1.3, 2.3.1, and 5.8.

Hamed Masnadi-Shirazi and Nuno Vasconcelos. Cost-sensitive boosting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(2):294–309, 2011. One citation in section 7.2.4.

Fabrice Muhlenbach, Stéphane Lallich, and Djamel A. Zighed. Identifying and handling mislabelled instances. *Journal of Intelligent Information Systems*, 22(1):89–109, 2004. 2 citations in sections 1.2 and 2.3.1.

Andrew Y. Ng. Feature selection, L1 vs. L2 regularization, and rotational invariance. In *Proceedings of the 21st International Conference on Machine Learning, (ICML'04)*, pages 78–84, 2004. One citation in section 5.5.2.

Andrew Y. Ng and Michael I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Proceedings of The Conference on Neural Information Processing Systems, (NIPS'01)*, pages 841–848. MIT Press, 2001. One citation in section 1.3.1.

Alexandru Niculescu-Mizil and Rich Caruana. Obtaining calibrated probabilities from boosting. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence , (UAI'05)*, 2005. One citation in section 7.2.4.

Steven W. Norton and Haym Hirsh. Classifier learning from noisy data as probabilistic evidence combination. In *Proceeding of the 10th National Conference on Artificial Intelligence, (AAAI'92)*, pages 141–146, 1992. 2 citations in sections 1.2 and 2.3.2.

Timo Ojala, Matti Pietikäinen, and Topi Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on*

Pattern Analysis and Machine Intelligence, 24(7):971–987, 2002. 2 citations in sections 4.5.4 and 6.2.5.

Paul Pavlidis, Jason Weston, Jinsong Cai, and William Noble Grundy. Gene functional classification from heterogeneous data. In *Proceedings of the 5th Annual International Conference on Computational Biology, (RECOMB'01)*, pages 249–255, 2001. One citation in section 6.1.1.

Mykola Pechenizkiy, Alexey Tsymbal, Seppo Puuronen, and Oleksandr Pechenizkiy. Class noise and supervised learning in medical domains: The effect of feature extraction. In *Proceeding of the 19th IEEE International Symposium on Computer-Based Medical Systems, (CBMS'06)*, pages 708–713, 2006. One citation in section 2.3.1.

John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in large margin classifiers*, pages 61–74. MIT Press, 1999. One citation in section 7.2.4.

Martin F. Porter. Readings in information retrieval. chapter An algorithm for suffix stripping, pages 313–316. Morgan Kaufmann, 1997. One citation in section 4.5.2.

John R. Quinlan. Bagging, boosting, and c4.5. In *Proceedings of the 13th National Conference on Artificial Intelligence, (AAAI'96)*, pages 725–730, 1996. One citation in section 2.3.1.

Gunnar Rätsch, Takashi Onoda, and Klaus-Robert Müller. Soft margins for AdaBoost. *Machine Learning*, pages 287–320, 2001. 2 citations in sections 6.2.1 and 7.3.2.

Vikas C. Raykar, Shipeng Yu, Linda H. Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. Learning from crowds. *Journal of Machine Learning Research*, 11:1297–1322, 2010. 5 citations in sections 1.2, 2.3.2, 4.4.2, 6.2.5, and 6.2.5.

Tim Robertson, Wright Farroll T., and Dykstra Richard L. *Order Restricted Statistical Inference*. John Wiley and Sons, New York, 1988. One citation in section 7.2.4.

Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24:513–523, 1988. One citation in section 4.5.2.

José S. Sánchez, Ricardo Barandela, A. I. Marqués, Roberto Alejo, and Jorge Badenas. Analysis of new techniques to obtain quality training sets. *Pattern Recognition Letters*, 24(7):1015–1022, 2003. 3 citations in sections 1.2, 2.3.1, and 4.5.3.

Robert E. Schapire. A brief introduction to boosting. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence, (IJCAI'99)*, pages 1401–1406. Morgan Kaufmann, 1999. One citation in section 2.2.

Shai Shalev-Shwartz. Introduction to machine learning. The Hebrew University of Jerusalem , `http://www.cs.huji.ac.il/~shais/Handouts.pdf`, 2009. One citation in section 4.4.6.

Shirish K. Shevade and Sathiya S. Keerthi. A simple and efficient algorithm for gene selection using sparse logistic regression. *Bioinformatics*, 19(17):2246–2253, 2003. 4 citations in sections 5.1.1, 5.2, 5.2.1, and 5.5.1.

Robert H. Sloan. Four types of noise in data for PAC learning. *Information Processing Letters*, 54:157–162, May 1995. 2 citations in sections 1.2 and 2.4.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. Cheap and fast - but is it good? evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, (EMNLP'08)*, pages 254–263, 2008. 2 citations in sections 1.2 and 6.2.5.

Takashi Takenouchi, Shinto Eguchi, Noboru Murata, and Takafumi Kanamori. Robust boosting algorithm against mislabeling in multiclass problems. *Neural Computation*, 20(6):1596–1630, June 2008. 2 citations in sections 2.1 and 2.3.2.

Leslie G. Valiant. A theory of the learnable. *Communications of the ACM*, 27:1134–1142, November 1984. One citation in section 2.4.

Leslie G. Valiant. Learning disjunction of conjunctions. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence, (IJCAI'85)*, pages 560–566, 1985. One citation in section 2.4.

Hamed Valizadegan and Pang-Ning Tan. Kernel based detection of mislabeled training examples. In *Proceedings of the 7th SIAM International Conference on Data Mining, (SIAM'07)*, 2007. One citation in section 2.3.1.

Vladimir N. Vapnik. *Statistical learning theory*. Wiley, 1 edition, September 1998. 3 citations in sections 1.1, 1.1, and 1.1.

Sundara Venkataraman, Dimitris Metaxas, Dmitriy Fradkin, Casimir Kulikowski, and Ilya Muchnik. Distinguishing mislabeled data from correctly labeled data in classifier design. In *Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence, (ICTAI'04)*, pages 668–672, 2004. One citation in section 2.3.1.

Roman Vershynin. Introduction to the non-asymptotic analysis of random matrices. Technical report, Department of Mathematics, University of Michigan, November 2010. 4 citations in sections 3.5, 3.5, 3.5, and 3.5.5.

Alexander Vezhnevets and Vladimir Vezhnevets. Modest AdaBoost – teaching AdaBoost to generalize better. In *Proceeding of the International Conference on Computer Graphics and Vision, (GraphiCon'05)*, Novosibirsk Akademgorodok, Russia, 2005. One citation in section 2.3.2.

Mike West, Carrie Blanchette, Holly Dressman, Erich Huang, Seiichi Ishida, Rainer Spang, Harry Zuzan, John A. Olson Jr., Jeffrey R. Marks, and Joseph R. Nevins. Predicting the clinical status of human breast cancer by using gene expression profiles. *Proceedings of the National Academy of Sciences of the United States of America*, 98(20):11462–11467, 2001. 2 citations in sections 5.5.2 and 5.5.4.

Linli Xu, Koby Crammer, and Dale Schuurmans. Robust support vector machine training via convex outlier ablation. In *Proceedings of the 21st National Conference on Artificial Intelligence , (AAAI'06)*, pages 536–542, 2006. One citation in section 6.2.

Zenglin Xu, Rong Jin, Haiqin Yang, Irwin King, and Michael R. Lyu. Simple and efficient multiple kernel learning by group lasso. In *Proceedings of the 27th International Conference on Machine Learning, (ICML'10)*, pages 1175–1182, 2010. 2 citations in sections 1 and 6.2.4.

Tianbao Yang, Mehrdad Mahdavi, Rong Jin, Lijun Zhang, and Yang Zhou. Multiple kernel learning from noisy labels by stochastic programming. In *Proceedings of the 29th International Conference on Machine Learning, (ICML'12)*, pages 233–240, 2012. 3 citations in sections 6.2, 6.2.4, and 6.2.4.

Yutaka Yasui, Margaret Pepe, Li Hsu, Bao-Ling Adam, and Ziding Feng. Partially supervised learning using an EM-boosting algorithm. *Biometrics*, 60(1):199–206, 2004. 2 citations in sections 1.2 and 2.3.2.

Chen Zhang, Chunguo Wu, Enrico Blanzieri, You Zhou, Yan Wang, Wei Du, and Yanchun Liang. Methods for labeling error detection in microarrays based on the effect of data perturbation on the regression model. *Bioinformatics*, 25:2708–2714, 2009. 4 citations in sections 1.3, 2.3.1, 4.5.3, and 5.5.1.

Ji Zhu and Trevor Hastie. Kernel logistic regression and the import vector machine. In

Journal of Computational and Graphical Statistics, pages 1081–1088. MIT Press, 2001. One citation in section 1.3.2.

Xingquan Zhu, Xindong Wu, and Qijun Chen. Eliminating class noise in large datasets. In *Proceedings of the 20th International Conference on Machine Learning, (ICML'03)*, pages 920–927, 2003. One citation in section 2.3.1.