Convex Latent Modeling

by

Özlem Aslan

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Statistical Machine Learning

Department of Computing Science
University of Alberta

Abstract

Most machine learning problems can be posed as solving a mathematical program that describes the structure of the prediction problem, usually expressed in terms of carefully chosen losses and regularizers. However, many machine learning problems yield mathematical programs that are not convex in model parameters, forcing the consideration of heuristic optimization strategies that do not provide guarantees of solution quality. The main focus of this thesis is to develop convex approximations of important non-convex machine learning problems; in particular, new convex formulations for deep latent modelling and robust estimation are developed.

Training deep predictive models with latent hidden layers poses a hard computational problem: since the model parameters have to be trained jointly with inference over latent variables, the convexity of the training problem is usually destroyed. This thesis first proposes a novel reformulation of supervised training of a twolayer architecture by introducing a latent feature kernel, which allows a rich set of latent feature representations to be captured while still allowing useful convex formulations via semidefinite relaxation. To tackle the resulting computational problem, efficient training algorithms are developed to exploit the specific structure of the problem and overcome the inadequate scaling of general purpose semidefinite solvers. Promising empirical results are obtained that show useful hidden structure can still be captured even in the presence of convex relaxations.

The thesis then shows that the two–layer approach can be extended to handle an arbitrary number of latent layers. To achieve this extension, a novel layer loss

is proposed that is jointly convex in the adjacent normalized latent feature kernels. An efficient algorithmic approach is then developed for this extended formulation. Again, promising empirical results are obtained that demonstrate improved capabilities over single latent layer models. These results demonstrate the first fully convex formulation of training a deep architecture with an arbitrary number of hidden layers.

A final non-convex problem this thesis addresses is robust regression in presence of outliers. Although the field of robust regression is well established, standard estimators in the robust regression literature are not convex and pose intractable computational problems, while robust estimators proposed in the machine learning literature are only robust under unrealistic assumptions. To address these shortcomings, this thesis proposes a new formulation of robust regression that admits a convex relaxation and efficient training algorithm, while still satisfying nontrivial robustness and consistency guarantees.

To *my parents*.

" . . . in fact, the great watershed in optimization isn't between linearity and nonlinearity, but convexity and nonconvexity."

– R. Tyrrell Rockafellar

"Truth . . . is much too complicated to allow anything but approximations."

– John von Neumann

Acknowledgements

I have been very fortunate to be supervised by two masters, Dale Schuurmans and Csaba Szepesvári. I am grateful to them for giving me a chance to work with them and providing an excellent research environment with their encouragement, mentoring, wisdom, patience, kindness and humble communication. Not only have I acquired invaluable background from their diverse knowledge, intuition, and achievements in many topics, they have also shown the path to becoming a productive scientist with their elegant and brilliant scientific approaches and intellectual humility. They are my role models as excellent scientists and teachers.

I would like to thank my committee members Russ Greiner, Koby Crammer and Nilanjan Ray for their time and constructive feedback. I thank Xinhua Zhang and Yaoliang Yu for their precious collaboration.

I am proud to be member of the Reinforcement Learning Artificial Intelligence (RLAI) group. The culture and successes of the RLAI group has always been very motivating and inspiring. I would like to thank the RLAI group, the Alberta Machine Intelligence Institute (Amii) and all of the people at the Department of Computing Science for their direct or indirect support during my Ph.D.

Finally, I would like to thank my family and all of my friends for their encouragement and support along the way.

Contents

1 Introduction **1**

1.1 Convex Approximations for Global Training 5

1.2 Deep Nonlinear Models 6

1.3 Robust Models 12

1.4 Contributions 15

2 Background **18**

2.1 Tractability in Deep Learning 18

2.2 Tractability in Robust Statistics 22

3 Tractable Two-Layer Modeling **24**

3.1 Introduction 24

3.2 Two-Layer Conditional Modeling 25

3.2.1 Multi-Layer Perceptrons and Large-Margin Losses 28

3.3 Equivalent Reformulation 29

3.4 Convex Relaxation 33

3.5 Efficient Training Approach 34

3.6 Experimental Evaluation 36

3.7 Conclusion 42

4 Tractable Multi-Layer Modeling **44**

4.1 Introduction 45

4.2 Background . . . 46
4.3 Multi-layer Convex Modeling via Normalized Kernels . . . 50
4.3.1 Convex Relaxation of the Domain of Output Kernels M . . . 52
4.3.2 A Jointly Convex Multi-label Loss for Normalized Kernels . . . 53
4.4 Efficient Optimization . . . 54
4.5 Empirical Investigation . . . 60
4.6 Synthetic Experiments . . . 61
4.7 Experiments on Real Data . . . 63
4.8 Conclusion . . . 64

5 Tractable Robust Modeling **65**
5.1 Introduction . . . 65
5.2 Background . . . 66
5.2.1 Consistency . . . 69
5.2.2 Robustness . . . 72
5.2.3 Computational Dilemma . . . 73
5.3 Variational M-estimation . . . 77
5.4 Computationally Efficient Approximation . . . 78
5.5 Properties . . . 82
5.5.1 Tractability Properties . . . 82
5.5.2 Robustness Properties . . . 85
5.5.3 Consistency Properties . . . 90
5.6 Experimental Evaluation . . . 91
5.7 Conclusion . . . 95

6 Conclusion and Future Work **96**

A Supplementary Material **119**
A.1 Details for Training a Two-layer Model . . . 119

A.1.1 Faster Gradient Calculation by Low Rank Decomposition . 119
A.1.2 Experimental Details for Two-layer Model 121
A.2 Details for Training a Multi-layer Model 123
A.2.1 Proof that the Feasible Region of (4.12) is Convex 123
A.3 Details for Training a Robust Model 124
A.3.1 Variational form for bounded loss functions 124
A.3.2 Least Trimmed Loss 124
A.3.3 Variational form of other Robust regression forms 125
A.3.4 Proofs of NP-hardness 125

B Some Basic Background **139**
B.1 Convexity Background 139
B.2 Semidefinite Relaxations 143
B.2.1 Semidefinite Programming 143
B.2.2 Recovering a Solution from a SDR 146
B.2.3 SDR in Machine Learning 148
B.2.4 SDP Solvers 149
B.3 Deep Nonlinear Models 154
B.4 Robust Models 155

List of Tables

3.1 Mean test misclassification error % (± stdev) for artificial datasets . 40
3.2 Mean test misclassification error % (± stdev) for 100/100 labeled/unlabeled. 41
3.3 Mean test misclassification error % (± stdev) for 200/200 labeled/unlabeled. 42

4.1 Mean test misclassification error % ((± stdev) 100/100 labeled/unlabeled. 63
4.2 Mean test misclassification error % ((± stdev) 200/200 labeled/unlabeled. 64

5.1 RMSE on clean test data for an artificial data set with 5 features and 100 training points, with outlier probability p, and 10000 test data points. Results are averaged over 10 repetitions. Standard deviations are given in parentheses. 92
5.2 RMSE on clean test data for 108 training data points and 1000 test data points, with 10 repeats. Standard deviations shown parentheses. The mean gap values of CvxBndL2 and CvxBndL1, Gap(Cvx2) and Gap(Cvx1) respectively, are given in the last two rows. 94

List of Figures

1.1 Surrogate losses. 4
1.2 Deep latent architecture in a conditional setting. 8
1.3 The outputs of layers of CNN. 11
1.4 The edge filter visualization of first layers. 11
1.5 Hierarchical learnt representations in second and third layers. 12
1.6 The squared error on a data with outliers. 14
1.7 The squared error on data with one outlier. 14
1.8 The squared, Huber and absolute loss. 15

3.1 Latent conditional model 26
3.2 "Xor" dataset 38
3.3 "boxes" dataset 39
3.4 "interval" dataset 39
3.5 The latent kernel for "Xor" test data 41

4.1 Multi-layer conditional models 46
4.2 Parity dataset results 62
4.3 Inner Product dataset results 62

A.1 Weight matrix V for second layer 121
A.2 The latent matrix H 122
A.3 Response matrix VH for second layer 122

A.4 Depiction of the error surface for the $u_i \vee u_j$ type of clause (for $i \neq j$) 128
A.5 Depiction of the error surface for the $u_i \vee \neg u_j$ type of clause (for $i \neq j$) . . . 129
A.6 Depiction of the error surface for the $\neg u_i \vee u_j$ type of clause (for $i \neq j$) . . . 129
A.7 Depiction of the error surface for the $\neg u_i \vee \neg u_j$ type of clause (for $i \neq j$) . . . 130
A.8 Illustrating bins of width $\frac{B_1\sqrt{2}}{s}$ between $\frac{B_1(1+\sqrt{2})}{s}$ and $\frac{1}{2}$, and their mirror images between $\frac{1}{2}$ and $1 - \frac{B_1(1+\sqrt{2})}{s}$. . . . 131
A.9 Illustrating how an empty (bin, mirror bin) pair can be used to define which component weight values are "nearly discrete" versus "close to $\frac{1}{2}$". . . . 132
A.10 Illustrating the rounding scheme of Lemma 6 for a $u_i \vee u_j$ clause. (The scheme for the other three clause types depicted in Figures A.4, A.5, A.6 and A.7 are isomorphic. . . . 133
A.11 Possible weight pairs that can occur after rounding. . . . 135

B.1 Some convex and nonconvex sets and the illustration of convex hull (red dotted lines and inside) of nonconvex sets . . . 140
B.2 The shapes of some convex and nonconvex functions. . . . 141
B.3 A FW iteration illustration. . . . 152
B.4 Some well known activation functions for hidden layers. . . . 155
B.5 The formulations of the losses and the influence functions for some M-estimators. . . . 157
B.6 The figures of Huber, Cauchy, Geman-McClure, Welsch and Turkey from left to right. . . . 158

Chapter 1

Introduction

Machine Learning (ML) has been playing the lead role in achieving impressive advances in challenging prediction problems across several areas of perceptual computing; including speech recognition (Dahl et al., 2012), image analysis and object detection (Krizhevsky et al., 2012; Le et al., 2012), and natural language processing (Socher et al., 2011). Recently, ML has also facilitated revolutionary progress in more general artificial intelligence tasks (Mnih et al., 2015; Silver et al., 2016). It is interesting to note that these results have all been based on deep learning methods, suggesting that *predictive representation learning* is a critical principle for obtaining state of the art prediction performance.

The automatic acquisition of representations is motivated by the observation that appropriate features make any learning problem easy, whereas poor features hamper learning. Given the practical significance of feature engineering, automated methods for feature discovery offer a valuable tool for applied machine learning. Ideally, automatically acquired features capture simple but salient aspects of the input distribution, upon which subsequent feature detection can compose increasingly abstract and invariant aspects (Bengio, 2009); an intuition that appears to be well supported by recent empirical evidence (Bengio et al., 2013).

The fundamental principles that underlie deep learning (formerly artificial neu-

ral networks) and reinforcement learning have been well studied for decades. However, the recent availability of massive computational resources has enabled well-known but not previously dominant methods to improve the state of the art in several large scale problems (Bottou et al., 2016). While these recent achievements are promising, there remains a need for significant progress on reducing training time and improving prediction performance to yield satisfactory performance on many real world problems. Of course, there is also a need for a comprehensive theoretical understanding. These goals require the careful design of learning procedures and improvement in their algorithmic efficiency to solve large scale problems more quickly and accurately. A working hypothesis of this thesis is that such improvements can be achieved by leveraging current techniques in field of numerical optimization.

Most training problems in ML can be formulated as a mathematical program (MP) (Bennett & Hernandez, 2006; Sra et al., 2012) (or as an integration problem, but I will not consider integration formulations in this thesis). Generally, an MP is formulated as

$$\min_{x \in \Sigma} f(x) \tag{1.1}$$

$$f_i \leq 0 \tag{1.2}$$

$$f_i = 0 \tag{1.3}$$

where Σ is the subset of $\mathbb{R}^n$, $\mathbb{N}$ or $\mathbb{S}^n$ where $\mathbb{S}^n$ denotes set of positive semi-definite matrices. Below we will cover how the problems in ML can be formulated as MP's.

According to the *empirical risk minimization* (ERM) principle described by Vapnik (1998), a 'learning process' consists of an input space $\mathcal{X}$ where $\mathbf{x} \in \mathcal{X}$ has a fixed but unknown distribution $P(\mathbf{x})$, an output space $\mathcal{Y}$ where $y \in \mathcal{Y}$ has a fixed but unknown distribution $P(y|\mathbf{x})$, and a hypothesis space $\mathcal{H}$ that consists of hypotheses $h \in \mathcal{H}$ each of which is a predictor function $h : \mathcal{X} \mapsto \mathcal{Y}$. Let us define the product space as $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ where $\mathbf{z} \in \mathcal{Z}$ and a fixed but unknown joint

distribution $\mathcal{D}(\mathbf{x}, y) = P(\mathbf{x})P(y|\mathbf{x})$. We are given a training set that is composed of n instances $\mathcal{S} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_1) \dots (\mathbf{x}_n, y_n)\} = \{\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3 \dots \mathbf{z}_n\}$ identically and independently drawn according to a joint distribution $\mathcal{D}(\mathbf{z})$. The goal in a supervised learning problem is to solve the following risk minimization problem

$$\min_{\mathbf{h} \in \mathcal{H}} \mathcal{R}(\mathbf{h}) = \mathbb{E}_{Z \sim \mathcal{D}}[f(\mathbf{h}; Z)] \tag{1.4}$$

where $f(\mathbf{h}; (\mathbf{x}, y)) = \ell(h(\mathbf{x}), y)$ is a known cost function (Shalev-Shwartz et al., 2010)(Vapnik, 1998). We cannot perform this optimization since the joint distribution $\mathcal{D}$ is unknown. Hence we use the training set to calculate and minimize the empirical risk:

$$\mathcal{R}_{emp}(\mathbf{h}) = \frac{1}{n} \sum_{i=1}^{n} \ell(h(\mathbf{x}), y) \tag{1.5}$$

which defines the objective function of an MP. In the case of classification, where the y values correspond to class labels, the true error consists of counting misclassifications, referred to as 'zero-one' loss

$$\ell(h(\mathbf{x}), y) = \mathbb{1}_{(h(\mathbf{x}) \neq y)} \tag{1.6}$$

where $\mathbb{1}$ is the indicator function. However this loss is neither convex (Appendix B.1 provides some basic background on convexity), or continuous, nor differentiable, and this loss is known to be NP-hard to minimize (Ben-David et al., 2003; Feldman et al., 2009). Using *surrogate* losses that are generally a convex upper bound of the true error is a common approach to this problem. Some well-known such surrogate losses are

$$\text{Exponential} : \exp(-yf) \tag{1.7}$$

$$\text{Binomial Deviance} : \log_2(1 + 2^{-2yf}) \tag{1.8}$$

$$\text{Squared} : (y - f)^2 \tag{1.9}$$

$$\text{Hinge} : \max(0, 1 - yf). \tag{1.10}$$

These losses are illustrated in Figure 1.1.

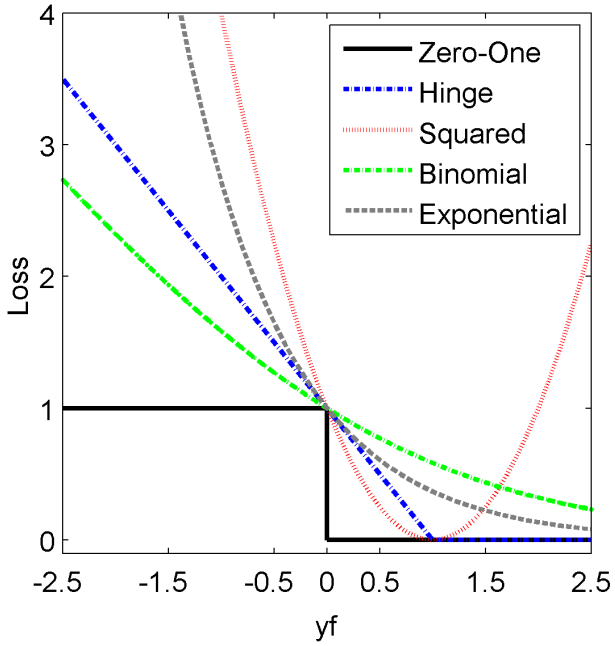


Figure 1.1: Surrogate losses.

A well-known performance measure for training procedure is the misclassification error which is simply the average difference between predictions and known targets. A standard caveat is that when one selects the best hypothesis based on only training error, there is a risk of obtaining a high *generalization* error (i.e. a large misclassification error on unseen test data). In general, we therefore also define a structure over the hypothesis space to overcome such issues. For example, in *structural risk minimization* (SRM), for a predictor f with parameter w, $f(\mathbf{x}, w)$, a structure is constructed by controlling the model complexity via bounding the norm of the weights using some $\mathcal{S}_i = \{w : ||w|| \leq C_i\}$ and $C_1 < C_2 \cdots < C_k$. Then, the MP equivalent to this constraint minimization has the form

$$\mathcal{R}_{emp}(\mathbf{h}) = \frac{1}{n}\sum_{i=1}^{n} \ell(h(\mathbf{x}), y) + \gamma_i ||w|| \tag{1.11}$$

where γ_i is the Lagrange multiplier, which can be approximately chosen such that $\gamma_1 > \gamma_2 \cdots > \gamma_k$ for γ_i matching C_i with same index respectively. The term $\gamma_i ||w||$

is also known as a *regularization* term, and different norms are designed for various problem structures. Different choices of loss and regularization, which yield different MP's, can require the development of novel algorithms that exploit the structure of the MP, since off-the-shelf optimization methods often do not give reasonable performance for large-scale learning problems (Bottou et al., 2016). Since ML problems can be formulated as a MP, optimization is almost always at the heart of ML (Bennett & Hernandez, 2006).

1.1 Convex Approximations for Global Training

Convexity has always been a central topic in the field of optimization (Rockafellar, 1993). It has been shown that checking whether a feasible solution is not a local minimum for a general smooth non-convex function is NP-complete (Murty & Kabadi, 1987) which means it is NP-complete to find a global minimum. A convex optimization problem guarantees that any local solution is globally optimal, while local solutions can be suboptimal in a non-convex optimization problem; worse, it is hard to know how close a local minima might be to global optimality in such cases (Boyd & Vandenberghe, 2004). In the context of ML, where one typically fits models to data by optimizing some parameters such as in Equation 1.11, non-convexity can negatively effect prediction performance. Sophisticated and often computationally expensive heuristics have been used in practice to handle non-convex machine learning problems, usually without any guarantees of solution quality. The heuristics that have been proved to work well in practice have been very difficult to discover, often taking a decade or more of empirical exploration. Moreover, these heuristic approaches yield methods that have limited foundation, making their use difficult for anyone who has not subjected themselves to the same experience of designing such heuristics. Experienced experts can carefully initialize, tune and adopt the solvers that try to minimize a non-convex learning problem for obtaining

some acceptable results. Note that this trial and error process can be also costly as exhaustively trying bad initializations and parameter values without experience can take a significant time (e.g. in deep learning setting). If one obtains local solutions for a non-convex problem, different results can be obtained with different initializations and heuristics. It is hard to see whether any change on the result is coming from a poor local minima or poorly designed objective function for a specific problem. By contrast, convex problems do not create such difficulties since any local minimum is guaranteed to be globally optimal. Non-experts are often able to solve convex problems by downloading packages and running them on their own tasks without worrying about the impact of suboptimality on their solutions. Despite the current popularity of non-convex formulations in ML, there have been many popular non-convex ML problems that have been improved by convex approximations (Lanckriet et al., 2004; Xu & Schuurmans, 2005; Lanckriet et al., 2007; Argyriou et al., 2008). The primary focus of this thesis is expanding the set of non-convex problems in ML that can be tackled by convex formulations, by employing convex relaxation (Vandenberghe & Boyd, 1996).

1.2 Deep Nonlinear Models

Deep learning has recently been enjoying a resurgence (Le et al., 2012; Srivastava & Salakhutdinov, 2012; Krizhevsky et al., 2012) due to the discovery that initialization heuristics can significantly improve the results of classical training methods (Bengio et al., 2013; Bengio, 2009; Hinton, 2007; Hinton et al., 2006). Deep learning is essentially an instance of *latent variable modelling*. The advantage of latent variable models is that they allow abstract *semantic* features of observed data to be represented, which can enhance the ability to capture predictive relationships between observed variables. Capturing the semantic features of words is a well-known example of latent modelling in natural language processing: for instance,

'girl' has the semantic features of 'human', 'young' and 'female', while 'boy' has has the semantic features of 'human', 'young' and ''male'. For image analysis, the presence of high level objects can define latent features in the context of scene classification: a 'street' scene can have instances of 'tree', 'road', 'building' and 'sky', for example.

In this way, latent variable models can greatly simplify the description of otherwise complex relationships between observed variates. For example, in unsupervised settings, latent variable models have been used to express feature discovery problems such as dimensionality reduction (Lawrence, 2005), clustering (Banerjee et al., 2005), and sparse coding (Elad & Aharon, 2006), to name a few. More recently, such latent variable models have been used to discover abstract features of visual data that are invariant to low level transformations (Le et al., 2012; Srivastava & Salakhutdinov, 2012; Hinton, 2007). These learned representations not only facilitate understanding, they can also enhance subsequent learning.

In this thesis, I restrict attention to latent variable models for conditional modeling. One simple setup for the architecture of a basic multi-layer conditional model with fully connected layers is shown in Figure 1.2. Here, the pixel values of the grayscale digits are the values of an example shown in first layer and the corresponding label that identifies the digit is the output in last layer. The middle layers, $\{\phi_i\}$, are latent (or *hidden*) layers that store the values of hierarchical representations. Hierarchical latent layers are modelled by nested nonlinearities: each successive layer, ϕ_i, is calculated by multiplying the previous layer ϕ_{i-1} by a weight matrix W_i and applying a nested nonlinear transfer (*activation*) function. A loss function with the output of last layer and target labels as two arguments is minimized to obtain the parameters. A simple two-layer model can discriminate some datasets that are not linearly separable, by performing appropriate nonlinear transformations at a hidden layer (e.g. representations that correspond to AND operations in a latent layer can separate the well-known XOR dataset).

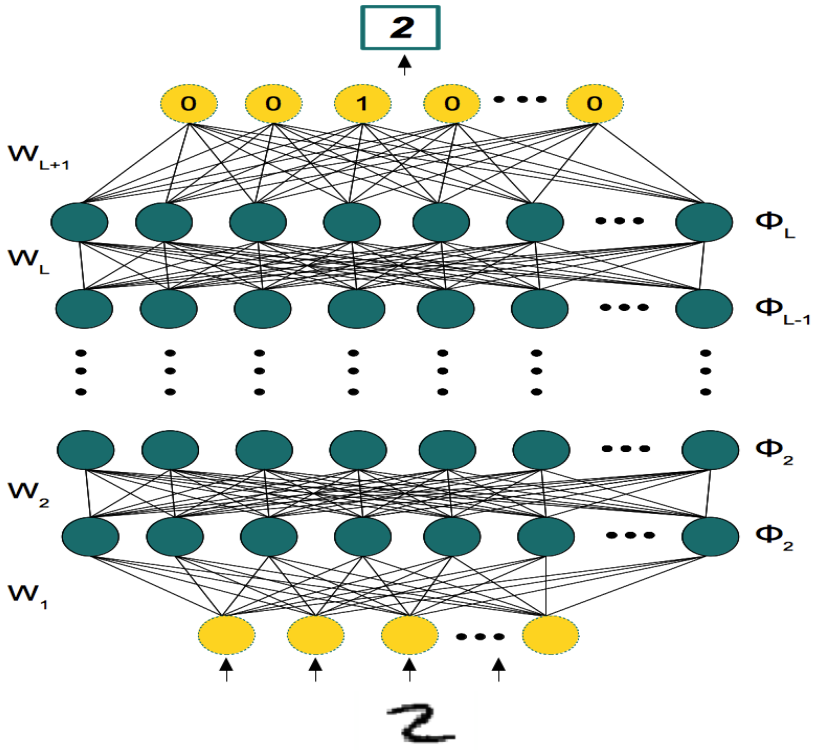


Figure 1.2: Deep latent architecture in a conditional setting.

In a supervised (i.e. *conditional*) setting, latent variable models are used to discover intervening feature representations that allow more accurate reconstruction of outputs from inputs. One advantage in the supervised case is that output information can be used to better identify relevant features to be inferred. However, latent variables also cause difficulty in this case because they impose nested nonlinearities between the input and output variables. Some important examples of conditional latent learning approaches include those that seek an intervening lower dimensional representation (Carreira-Perpiñán & Lu, 2010), latent clustering (Tishby et al., 1999), sparse feature representation (Elad & Aharon, 2006) or invariant latent representation (Le et al., 2012; Rifai et al., 2011; Bengio, 2009; Hinton, 2007) between inputs and outputs.

Depth has been an essential property for obtaining state of the art results in

benchmark datasets, such as image datasets like CIFAR 10 and ImageNet (Bengio et al., 2013). Although this fact has been hard to rigorously formalize, the common intuition is that feature re-use in a directed acyclic hierarchy leads to more accurate and invariant representations being learned than in shallower models (Bengio et al., 2013). There have been some theoretical attempts to understand the effect of depth. For networks with piecewise linear activations, Montufar et al. (2014) show that a composition of layers exponentially increases the number of linear input regions, and claim that this property increases the expressiveness of the model class while yielding better generalization performance for deeper models.

The intuition behind using hierarchical latent layers via nested nonlinearities has been grounded in the fact that natural perceptive signals in the world exhibit compositional hierarchies: higher level representations are compositions of lower-level representations (LeCun et al., 2015; Cun et al., 1990). If we consider the features of images, it can be observed that local combinations of edges form motifs, motifs are combined to form parts and parts form objects (LeCun et al., 2015). This hierarchical feature structure is also valid for speech and text data. Methods that yield state of art results are mostly based on the *convolutional neural network* (CNN) architectures which have four key properties: *local connections*, *shared weights*, *pooling* and many layers (LeCun et al., 2015). The first three properties are enforced by convolutional and pooling layers. Convolutional layers are composed of feature maps where each unit is connected to each of the local patches (*receptive fields*) in a previous feature map, where units of each feature map share the same weights. This locality idea is based on two principles: the local features in the image are highly correlated and the same motif can appear in different locations. The pooling operation merges similar features with similar semantics into one. A CNN architecture is shown in Figure 1.3 which is taken from LeCun et al. (2015). The Figure 1.4 that is taken from Lee et al. (2011) shows the edge filters that are commonly extracted in lower layers of deep architectures. The higher lever representations learnt, which

are taken from for different tasks, are shown in Figure 1.5 which is taken from Lee et al. (2011). These visualizations empirically support the intuitive claims about the inherent structures that can be captured by deep architectures.

Despite their growing success, the difficulty of training a predictive latent variable model remains clear: since the model parameters have to be trained concurrently with inference over latent variables, the convexity of the training problem is usually destroyed. A well-known result states that a neural network with just one hidden layer can approximate any smooth function (Barron, 1993), however it has been shown that finding the weights that best fit training data in a one layer network is NP-hard (Blum & Rivest, 1992). Therefore, in practice, one relies upon heuristic methods. In particular, stochastic gradient descent algorithms have become the most common approach to training deep models. These methods optimize the hypothesis parameters through a calculation of gradients on individual training examples via the chain rule (known as *back propagation*), which allows scaling to large datasets more readily than obtaining a full gradient by summing over an entire dataset. There are also second order methods that shown some promising results (Martens, 2016). These stochastic methods have theoretical guarantees for convex functions, but there are only a few weak results for non-convex functions.

Unfortunately, only highly restricted models can be trained to optimality, and current deep learning strategies provide no guarantees about solution quality (Gori & Tesi, 1992). This remains true even when restricting attention to a single stage of stage-wise pre-training: simple models such as the two-layer auto-encoder or restricted Boltzmann machine (RBM) still pose intractable training problems, even within a single stage (in fact, simply computing the gradient of the RBM objective is currently believed to be intractable (Swersky et al., 2011)). Beyond well known problems like local minima (Gori & Tesi, 1992), deep training landscapes also exhibit plateaus (Erhan et al., 2010) that arise from credit assignment problems in backpropagation. An intuitive understanding of the optimization landscape

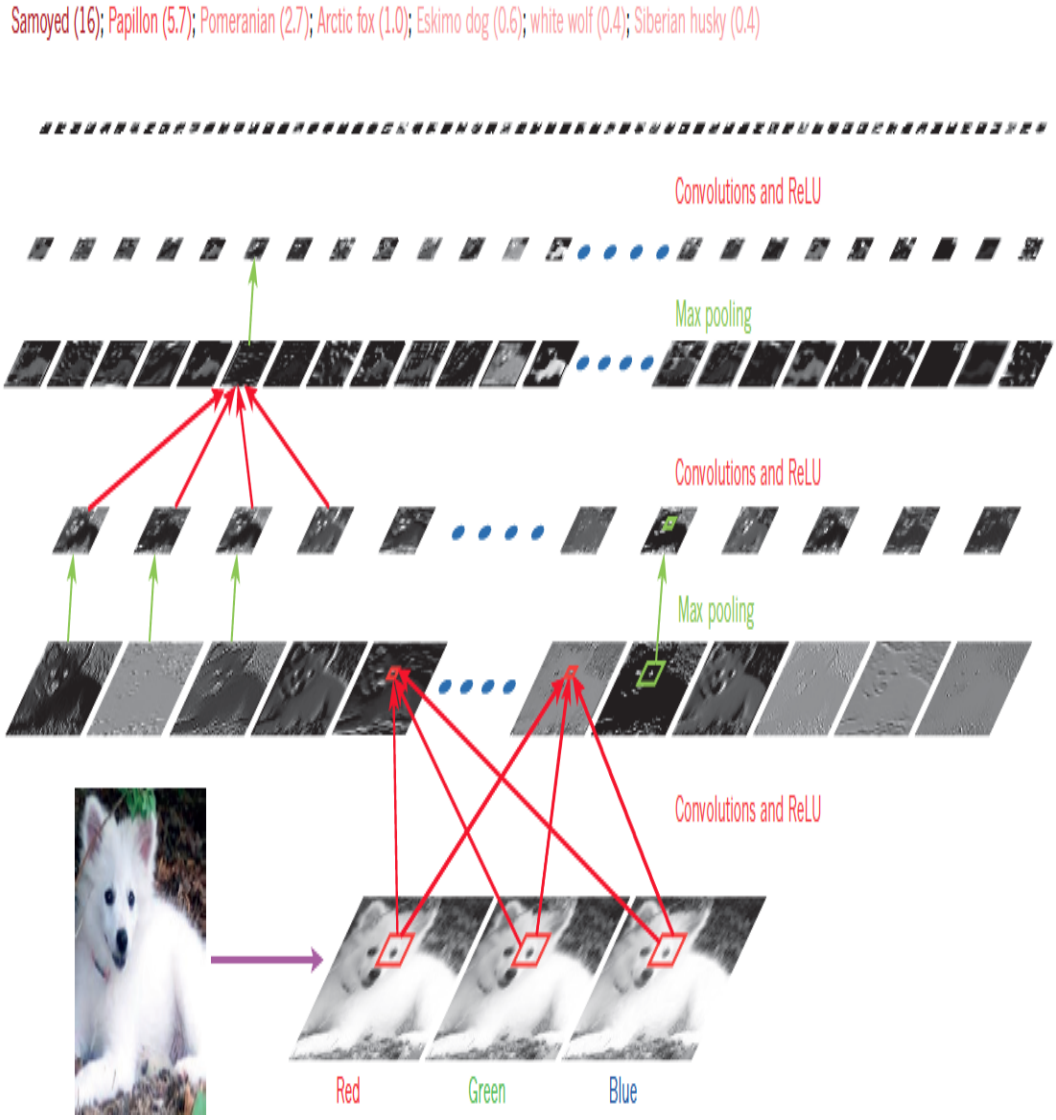


Figure 1.3: The outputs of layers of CNN.

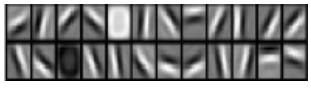


Figure 1.4: The edge filter visualization of first layers.

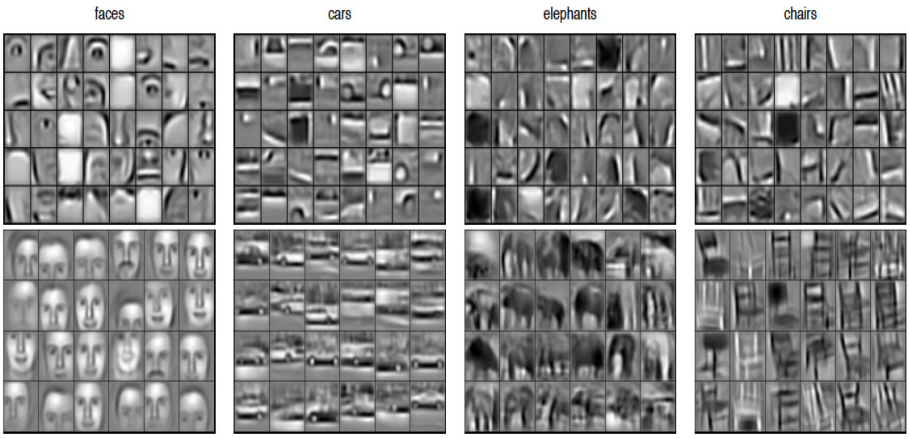


Figure 1.5: Hierarchical learnt representations in second and third layers.

and careful initialization both appear to be essential aspects of obtaining successful training (Sutskever et al., 2013). Nevertheless, the development of recent training heuristics has improved the quality of feature discovery at lower levels in deep architectures. These advances began with the idea of bottom-up, stage-wise unsupervised training of latent layers (Hinton et al., 2006; Vincent et al., 2010) (*pre-training*), and progressed to more recent ideas like dropout (Hinton et al., 2012; Krizhevsky et al., 2012). Despite the resulting empirical success, however, such advances occur in the context of a problem that is known to be NP-hard in the worst case (even to approximate (Hoeffgen et al., 1995)), hence there is no guarantee that worst case versus "typical" behavior will not show up in any particular problem.

1.3 Robust Models

Beyond learning deep representations, another important concern when learning conditional predictors is resistance to outliers in data. Outliers are data points that are abnormal relative to the rest of the data (Rousseeuw & Leroy, 1987a). Such data points usually arise from some form of contamination, whether by mistakes in measurements, or faulty experiments, etc. Such data points can adversely affect the

quality of model learned when they are not noticed.

In this thesis, I will restrict attention to regression problems. For regression, the standard convex losses are sensitive to outliers. For example, the squared loss is a well-known non-robust regression loss. The problem with squared loss is that it puts more emphasis on outliers since the loss grows quadratically for large residuals, as can be seen in Figure 1.8. The least square estimation for the number of international phone calls from Belgium in years 1950-1973 is shown in Figure 1.6 that is taken from (Rousseeuw & Leroy, 1987b). The data has many outliers since the total number of minutes instead of the number of these calls was reported in 1964 -1969 and partly in 1963 and 1970 because of a change in measurement system. It can be seen that the least squares solution is affected by the outliers in the measurement errors. In fact, even a single erroneous observation can arbitrarily change the estimates produced by methods such as least squares, which can be seen in Figure 1.7.

Although there are preprocessing techniques that attempt to eliminate outliers before parameter estimation, such methods are generally not reliable since it is not straightforward to detect the outliers without a model, which is what we are trying to learn in the first place. In general, preprocessing techniques can harm performance by eliminating original data points rather than outliers (Rousseeuw & Leroy, 1987a). Therefore, the field of robust statistics has investigated losses that are designed to be resistant to outliers, which allows estimation and outlier detection to be performed concurrently. (I present some basic background related to robust estimation in Section B.4 of the appendix.) One example is the Huber loss shown in Figure 1.8, which grows only linearly for large residuals and hence is less sensitive to outliers than squared loss. Unfortunately, even Huber loss is not robust, even in the weak sense that I will investigate in this thesis.

Although the field of robust regression is well established, it has not considered computational complexity analysis to be one of its central concerns: there is

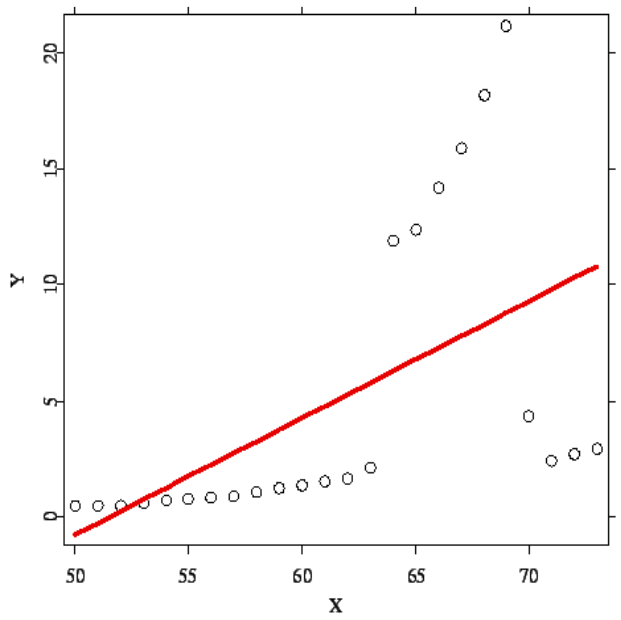


Figure 1.6: The squared error on a data with outliers.

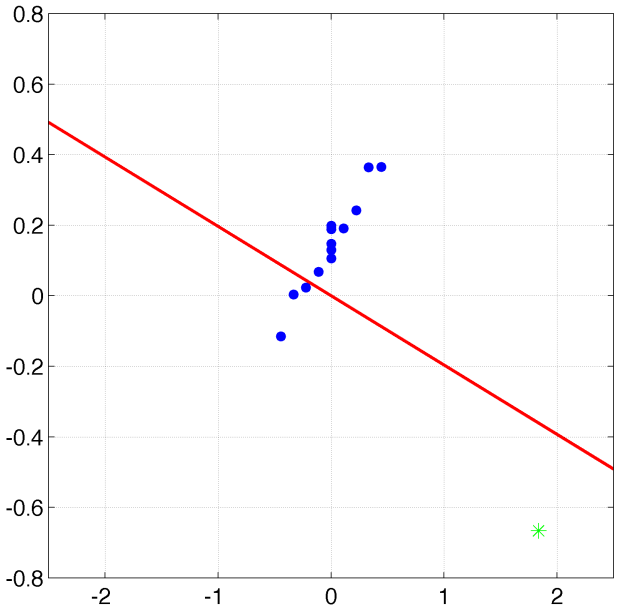


Figure 1.7: The squared error on data with one outlier.

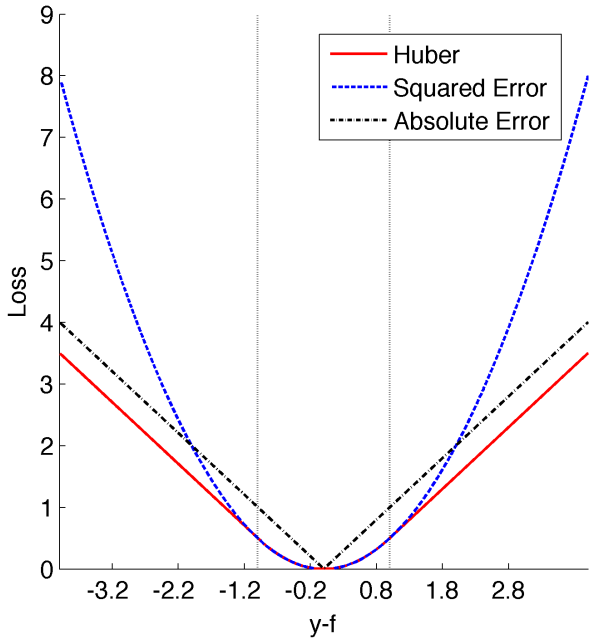


Figure 1.8: The squared, Huber and absolute loss.

no current loss based regression procedure that is both robust and tractable (Bernholt, 2005; Nunkesser & Morell, 2010). Therefore another focus of this thesis is to develop loss based estimators that are tractable and have robustness properties. The main strategy I will investigate is a convex relaxation approach to obtaining a tractable robust estimation procedure.

1.4 Contributions

In this thesis, I propose the following three main contributions that address the intractability of learning deep latent models and robust estimation.

Convex Two-layer Modeling. The first contribution of this thesis, presented in Chapter 3, is to demonstrate that a latent variable structure can be accommodated within a tractable convex framework. In particular, we show how two-layer latent

conditional models with a single latent layer can be expressed equivalently in terms of a *latent feature kernel*. This reformulation allows a rich set of latent feature representations to be captured, while allowing useful convex relaxations in terms of a semidefinite optimization. Unlike (Cho & Saul, 2010), the latent kernel in this model is explicitly learned (non-parametrically). To cope with scaling issues, we further develop an efficient algorithmic approach for the proposed relaxation. Importantly, the resulting method preserves sufficient problem structure to recover prediction models that cannot be represented by any one-layer architecture over the same input features, while improving the quality of local training. This work was published in Advances in Neural Information Processing Systems 2013 (Aslan et al., 2013).

Convex Multi-layer Modeling. Although the first contribution develops a convex formulation for latent kernel learning, the approach is restricted to a single latent layer. The next contribution of this thesis, presented in Chapter 4, is to develop a convex formulation of multi-layer learning that allows multiple latent kernels to be connected through nonlinear conditional losses. In particular, each pair of successive layers is optimized with a prediction loss that is jointly convex in the adjacent kernels, while expressing a non-trivial, non-linear mapping between layers that supports multi-factor latent representations. The resulting formulation significantly extends the previous convex model, which is only able to train a single adaptive kernel while maintaining a convex training objective (Aslan et al., 2014). Additional algorithmic developments yield an approach with improved scaling properties over previous methods, although not yet at the level of current deep learning methods. We believe the result is the first fully convex training formulation of a deep learning architecture with adaptive hidden layers, which demonstrates some useful potential in empirical investigations. This work has been published in Advances in Neural Information Processing Systems 2014 (Aslan et al., 2014).

Convex Robust Modeling. Another contribution of this thesis, presented in Chapter 5, is to develop a convex approximation of general robust regression in the presence of outliers. First, instead of focusing on one particular robust regression form (e.g. a bounded loss from robust statistics), we adopt a general formulation of adaptive M-estimation, *Variational M-estimation*. We show that many bounded losses can be expressed with this adaptive form. Hence the variational form allows us to obtain a general flexible formulation. We then add Tikhonov regularization that allows us to extend the setting to reproducing kernel Hilbert spaces (RKHSs). We derive an equivalent formulation of the original adaptive estimation that is convenient to develop a novel convex relaxation. We provide results that show the convex relaxation gives estimator that is robust, consistent and tractable. We also empirically demonstrate the robustness and tightness of the approximation on an artificial dataset and some real datasets.This work has been published in Advances in Neural Information Processing Systems 2012 (Yu et al., 2012).

Chapter 2

Background

I first briefly review related work on global learning approaches to deep learning and robust estimation.

2.1 Tractability in Deep Learning

A growing body of research has investigated reformulations of latent variable learning that are able to yield tractable global training methods in special cases. Even though global training formulations are not a universally accepted goal of deep learning research (LeCun, 2007), there are several useful methodologies that have been been applied successfully to other latent variable models, including: boosting strategies (Bengio et al., 2005; Nowozin & Bakir, 2008; Bradley & Bagnell, 2009), semidefinite relaxations (Joulin & Bach, 2012; Joulin et al., 2010; Guo & Schuurmans, 2007) and matrix factorization (Goldberg et al., 2010; Candes et al., 2009; Zhang et al., 2012). Unfortunately, none of these approaches has yet been able to accommodate a non-trivial hidden layer between an input and output layer while retaining the representational capacity of an auto-encoder or RBM (e.g. boosting strategies embed an intractable subproblem in these cases (Bengio et al., 2005; Nowozin & Bakir, 2008; Bradley & Bagnell, 2009)). Some recent work has been

able to capture restricted forms of latent structure in a conditional model—namely, a single latent cluster variable (Joulin & Bach, 2012; Joulin et al., 2010; Guo & Schuurmans, 2007)—but this remains a rather limited approach.

One key motivation for recent theoretical work was to ground deep learning on a well understood computational foundation. For example, Arora et al. (2014) demonstrates that polynomial time (high probability) identification of an optimal deep architecture can be achieved by restricting weights to bounded random variates and considering hard-threshold generative gates. Other recent work (Livni et al., 2014a) considers a sum-product formulation (Gens & Domingos, 2012), where guarantees can be made about the efficient recovery of an approximately optimal polynomial basis. Although these treatments do not cover the specific models that have been responsible for state of the art results, they do provide insight into the computational structure of deep learning.

Another recent line of research focuses on exploring the structure of the non-convex cost landscape for training a deep model. This type of work has recently been unified under the label of non-convex optimization in ML. Initially, Baldi & Hornik (1989) showed that the optimization surface for a linear neural network with one hidden layer has no local minima (however, it has saddle points). Although they claim that this result should intuitively generalize to nonlinear models, no such result has been proved. More recently, Saxe et al. (2014) have shown that optimizing deep linear models share similarities to optimizing deep nonlinear models, such as experiencing plateaus followed by rapid transitions to lower error; they achieved these results by deriving exact solutions to the nonlinear learning dynamics for deep linear networks.

It has also been shown that the critical points of random Gaussian error functions on high dimensional continuous spaces are exponentially likely to be saddle points with both negative and approximate plateau directions, while all local minima are likely to have error close to the global minimum (Bray & Dean, 2007; Fyo-

dorov & Williams, 2007). Although this result seem promising, it is not straightforward to generalize these conclusions to the optimization surfaces created by deep neural networks. Dauphin et al. (2014) attempt to show empirically that there is a connection between nonlinear neural networks and the Gaussian error functions, arguing that neural networks also tend to produce many saddle points rather than local minima in high dimensional spaces. However this work is restricted to empirical analysis without any theoretical justification. They also propose a saddle-free Newton method to tackle saddle points, but the proposed method does not have reasonable scaling properties for training deep models. Choromanska et al. (2015a) extend this work by theoretically showing that neural network landscapes exhibit structures like the Hamiltonian of a spin-glass model, and these in turn have structure that suggests they are likely to have saddle points but not local minima in high dimensions. However Choromanska et al. (2015a) state that this connection relies on many unrealistic assumptions, and have posed the question of eliminating such constraints as an open problem at COLT 2015 (Choromanska et al., 2015b). Goodfellow et al. (2015) have also empirically investigated optimization trajectories taken by standard training methods on neural network landscapes. Kawaguchi (2016) recently extended the work of Choromanska et al. (2015a), solving one of the open problems in Choromanska et al. (2015a), by eliminating some of the constraints. However there is still a significant gap between theory and practice.

Another approach to the problem of non-convexity of neural networks is to impose assumptions on the cost function, regularizers and/or weights to be able to develop algorithms that can find a global minimum in polynomial time. For example, Sedghi & Anandkumar (2014) investigate a method-of-moments approach by using the cross-moments between the labels and score function of the input. In particular, they show that they can recover the first layer weights in a network with one hidden layer, given the assumptions that: the input comes from a known score function, the network is sparse, the weights come from a specific distribution, and

the problem has large dimension and a large number of output labels. Janzamin et al. (2015) eliminate many of these assumptions and provide an algorithm that is based on a tensor decomposition, under some non-degeneracy assumptions. These works are restricted to networks with a single hidden layer: extending the analysis to multi-layer networks remains a challenging open problem.

Livni et al. (2014b) investigate an alternative way to address intractability by considering an *improper* learning framework, where instead of fixing the network architecture, a different architecture that is almost as good as the target architecture is considered. They show that a neural network with quadratic activation functions and a sufficient number of layers and hidden units can approximate the sigmoid transfer function, from which they obtain a guaranteed approximation to the best network by greedily adding single hidden neurons using an eigen-decomposition. Unfortunately, the approach is not entirely practical since training time and sample complexities both grow quickly in the number of layers. A key bottleneck is that the sigmoidal approximation requires a high degree polynomial network. Zhang et al. (2015a) extend this approach to obtain a polynomial time guarantee with a kernel based classifier, under the assumption that the depth of the network is constant and the l_1 norm of the weight in each layer is also bounded by a constant. Unfortunately, these assumption do not normally hold in practice. (Zhang et al., 2015b) extend the result to non-kernel based learning, with a time complexity that is polynomial in the input dimension and sample size. Unfortunately, this result is exponential in the excess risk, and they could only provide results for artificial data, which suggests that the method does not easily scale to large datasets.

Training deep models with many layers is a hard task in practice since it is a highly non-convex optimization problem (Romero et al., 2015). Many heuristics have been tried to overcome the optimization difficulties. To eliminate vanishing gradients, common activation functions have been recently changed to ReLu type functions (Nair & Hinton, 2010). To further speedup training, batch normaliza-

tion has been introduced to reduce covariate shift (Ioffe & Szegedy, 2015). Deeper architectures such as AlexNet (Krizhevsky et al., 2012), VGG (Simonyan & Zisserman, 2015) and GoogLeNet (Szegedy et al., 2015) have exploited such heuristics to achieve state of the art performance on the ImageNet dataset. However accommodating dozens of layers still remains an empirically hard problem. Very recently, residual networks (ResNet), which explicitly add an identity map and parametrizes the residual, have made the training of hundreds of layers possible and significantly improved the performance on some image classification benchmarks (He et al., 2016).

Although many heuristics and theoretical analyses have been attempted, current deep learning strategies still do not offer guarantees of solution quality under realistic assumptions. The dramatic changes in the claims based on empirical evidence obtained from a few benchmark datasets (or from analysis based on unrealistic assumptions) indicates that there is still a great need for careful rigorous work, rather than a sole focus on trial and error based investigation.

2.2 Tractability in Robust Statistics

None of the standard regression estimators in the literature are both robust and tractable, even in a weak sense: it has been shown that standard robust regression formulations with non-zero breakdown are NP-hard (Bernholt, 2005; Nunkesser & Morell, 2010), while any estimator based on minimizing a convex loss cannot guarantee bounded response to even a single leverage point (Maronna et al., 2006) Surprisingly, there remain no standard regression formulations that guarantee both polynomial run-time with bounded response to even single outliers.

It is important to note that robustness and tractability can be achieved under restricted conditions. For example, if the domain is *bounded*, then any estimator based on minimizing a convex and Lipschitz-continuous loss achieves high breakdown

(Huber & Ronchetti, 2009). Such results have been extended to kernel-based regression under the analogous assumption of a bounded kernel (Christmann & Steinwart, 2007; Christmann et al., 2009). Unfortunately, these results can no longer hold when the domain or kernel is *unbounded*: in such a case arbitrary leverage can occur (Huber & Ronchetti, 2009; Rousseeuw & Leroy, 1987a) and no (non-constant) convex loss, even Lipschitz-continuous, can ensure robustness against even a single outlier (Maronna et al., 2006).

The closest previous work is by Black & Rangarajan (1996), who formulated variational representations of certain robust losses, and by Yu et al. (2010), who formulated a convex relaxation of bounded loss minimization. Unfortunately, Black & Rangarajan (1996) did not offer a general characterization, while Yu et al. (2010) did not prove that their final estimator was robust, nor was any form of consistency established. The formulation we present in Chapter 5 generalizes the work of Black & Rangarajan (1996) while the convex relaxation scheme we propose is simpler and tighter than that given by Yu et al. (2010); we are thus able to establish non-trivial forms of both robustness and consistency while maintaining tractability.

There are many other notions of robust estimation in the ML literature that do not correspond to the specific notion being addressed in this thesis. Work on *robust optimization* (El Ghaoui & Lebret, 1997; Xu et al., 2008, 2009) for example, considers minimizing the worst case loss achieved given bounds on the maximum data deviation that will be considered. Such results are not relevant to the present investigation because we explicitly do not bound the magnitude of the outliers. Another notion of robustness is algorithmic stability under leave-one-out perturbation (Mukherjee et al., 2006), which analyzes specific learning procedures rather than describing how a stable algorithm might be generally achieved.

Chapter 3

Tractable Two-Layer Modeling

Latent variable prediction models, such as multi-layer networks, impose auxiliary latent variables between inputs and outputs to allow automatic inference of implicit features useful for prediction. Unfortunately, such models are difficult to train because inference over latent variables must be performed concurrently with parameter optimization, creating a highly non-convex problem. Instead of proposing another local training method, in this chapter we instead develop a convex relaxation of hidden-layer conditional models that admits global training. The work presented in this chapter was published in Neural Information Processing Systems(NIPS) conference (Aslan et al., 2013).

3.1 Introduction

Many non-convex ML problems have been tackled by convex relaxations, as discussed in the introduction. The main bottleneck that makes the convex approximation much harder for this particular learning problem is nested nonlinearities. There are approaches that tackle the non-convexity of conditional latent models, however none of these can handle latent models that consist of one or more hidden layers with global optimum guarantee under realistic assumptions. In this chapter, rather

than tackling a model with many hidden layers, we first focus on a conditional model with one hidden layer, since this already creates a challenging non-convex problem. We present a convex relaxation strategy that yields a fully convex model for this problem. This appealing result allows a non-trivial extension to many hidden layers, which is covered in Chapter 4 in detail.

Our first contribution in this chapter is designing a predictive two-layer latent modeling as a joint optimization problem. This novel approach gives a convenient form by decoupling the nested nonlinearities. The next significant contribution is using output kernels to obtain a convex approximation for losses that have the output as unknown variable. Although we obtain the first convex relaxation result for a conditional model with one hidden layer, the nonlinear constrained semidefinite problem is computationally expensive to solve using off-the-shelf solvers. Therefore, we also develop an efficient algorithm to solve the final relaxation by carefully exploiting the problem structure.

The rest of the chapter is organized as follows: we first introduce our framework for two-layer conditional modelling in Section 3.2 where we eliminate nested nonlinearities by decoupling layers in a joint formulation. Here we apply a relaxation that accommodates losses for discrepancy between output of a layer and output of previous layer. We then develop a specific discriminative margin loss for hidden layers in Section 3.2.1. In Section 3.3, we derive the equivalent non-convex formulation which enables obtaining jointly convex formulation in Section 3.4. We then present the efficient algorithm that solves the final convex form in Section 3.5. Finally, we empirically demonstrate that proposed convex relaxation preserves the structure of the hidden layers in Section 3.6.

3.2 Two-Layer Conditional Modeling

We address the problem of training a two-layer latent conditional model in the form of Figure 3.1; i.e., where there is a single layer of h-dimensional latent variables,

$\boldsymbol{\phi}$, between a layer of n-dimensional input variables, $\mathbf{x}$, and m-dimensional output variables, $\mathbf{y}$. The goal is to predict an output vector $\mathbf{y}$ given an input vector $\mathbf{x}$. Here, a prediction model consists of the composition of two nonlinear conditional models, $\mathbf{f}_1(W\mathbf{x}) \rightsquigarrow \boldsymbol{\phi}$ and $\mathbf{f}_2(V\boldsymbol{\phi}) \rightsquigarrow \hat{y}$, parameterized by the matrices $W \in \mathbb{R}^{h \times n}$ and $V \in \mathbb{R}^{m \times h}$. Once the parameters W and V have been specified, this architecture defines a point predictor that can determine $\hat{\mathbf{y}}$ from $\mathbf{x}$ by first computing an intermediate representation $\boldsymbol{\phi}$.

To learn the model parameters, we assume we are given t training pairs $\{(\mathbf{x}_j, \mathbf{y}_j)\}_{j=1}^{t}$ stacked in two matrices $X = (\mathbf{x}_1, ..., \mathbf{x}_t) \in \mathbb{R}^{n \times t}$ and $Y = (\mathbf{y}_1, ..., \mathbf{y}_t) \in \mathbb{R}^{m \times t}$, but the corresponding set of latent variable values $\Phi = (\boldsymbol{\phi}_1, ..., \boldsymbol{\phi}_t) \in \mathbb{R}^{h \times t}$ remains *unobserved*. Also, let $\mathbf{1}$ denote the vector of all 1s with length determined by context.

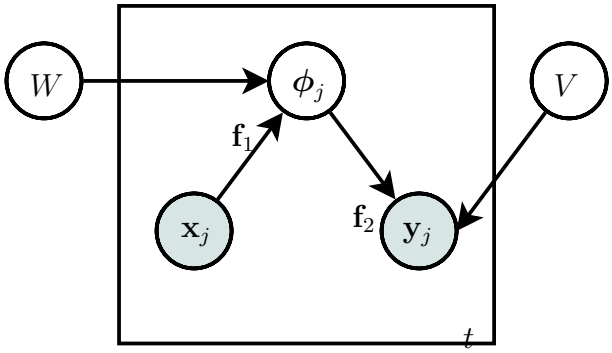


Figure 3.1: Latent conditional model

To formulate the training problem, we will consider two losses, L_1 and L_2, that relate the input to the latent layer, and the latent to the output layer respectively. For example, one can think of losses as negative log-likelihoods in a conditional model that generates each successive layer given its predecessor; i.e., $L_1(W\mathbf{x}, \boldsymbol{\phi}) = -\log p_W(\boldsymbol{\phi}|\mathbf{x})$ and $L_2(V\boldsymbol{\phi}, \mathbf{y}) = -\log p_V(\mathbf{y}|\boldsymbol{\phi})$. (However, a loss based formulation is more flexible, since every negative log-likelihood is a loss but not vice versa.)

Given such a set-up many training principles become possible. For simplicity, we consider a Viterbi based training principle where the parameters W and V are optimized with respect to an optimal imputation of the latent values Φ. To do so, define the first and second layer training objectives as

$$F_1(W, \Phi) = L_1(WX, \Phi) + \tfrac{\alpha}{2}\|W\|_F^2 \text{ and } F_2(\Phi, V) = L_2(V\Phi, Y) + \tfrac{\beta}{2}\|V\|_F^2 \quad (3.1)$$

where we assume the losses L_1 and L_2 are convex in their first arguments. Here it is typical to assume that the losses decompose columnwise; that is, $L_1(\hat{\Psi}, \Phi) = \sum_{j=1}^{t} L_1(\hat{\boldsymbol{\psi}}_j, \boldsymbol{\phi}_j)$ and $L_2(Z, Y) = \sum_{j=1}^{t} L_2(\hat{\mathbf{z}}_j, \mathbf{y}_j)$, where $\hat{\boldsymbol{\psi}}_j$ is the jth column of $\hat{\Psi}$ and $\hat{\mathbf{z}}_j$ is the jth column of $\hat{Z}$ respectively. This follows for example if the training pairs $(\mathbf{x}_j, \mathbf{y}_j)$ are assumed I.I.D., but such a restriction is not necessary. Note that we have also introduced Euclidean regularization over the parameters, that will provide a useful representer theorem (Kimeldorf & Wahba, 1971) we exploit later. These two objectives can be combined to obtain the following joint training problem:

$$\min_{W,V} \min_{\Phi} F_1(W, \Phi) + \gamma F_2(\Phi, V), \quad (3.2)$$

where $\gamma > 0$ is a trade off parameter that balances the first versus second layer discrepancy. Unfortunately (3.2) is not jointly convex in the unknowns W, V and Φ.

A key modeling question concerns the structure of the latent representation $\boldsymbol{\phi}$. As noted, the extensive literature on latent variable modeling has proposed a variety of forms for latent structure. Here, we follow work on deep learning and sparse coding and assume that the latent variables are boolean, $\phi \in \{0, 1\}^{h \times 1}$; an assumption that is also often made in auto-encoders (Swersky et al., 2011), PFNs (Neal, 1992), and RBMs (Hinton et al., 2006). A boolean representation can capture structures that range from a single latent clustering (Tishby et al., 1999; Joulin et al., 2010; Guo & Schuurmans, 2007), by imposing the assumption that $\boldsymbol{\phi}'\mathbf{1} = 1$, to a general sparse code, by imposing the assumption that $\boldsymbol{\phi}'\mathbf{1} = k$ for some small k (Le et al.,

2012; Swersky et al., 2011; Hinton, 2007). Observe that, in the latter case, one can control the complexity of the latent representation by imposing a constraint on the number of "active" variables k rather than directly controlling the latent dimensionality h.

3.2.1 Multi-Layer Perceptrons and Large-Margin Losses

To complete a specification of the two-layer model in Figure 3.1 (where $\mathbf{f}_1(W\mathbf{x}) \rightsquigarrow \boldsymbol{\phi}$, $\mathbf{f}_2(V\boldsymbol{\phi}) \rightsquigarrow \hat{y}$, $\boldsymbol{\phi}_j$ is a latent variable, $\mathbf{x}_j$ is an observed input vector, $\mathbf{y}_j$ is an observed output vector, W are first layer parameters,and V are second layer parameters.) and the associated training problem (3.2), we need to commit to specific forms for the transfer functions $\mathbf{f}_1$ and $\mathbf{f}_2$ and the losses in (3.1). For simplicity, we will adopt a *large-margin* approach over two-layer *perceptrons*. Although it has been traditional in deep learning research to focus on exponential family conditional models (e.g. as in auto-encoders, PFNs and RBMs), these are not the only possibility; a large-margin approach offers additional sparsity and algorithmic simplifications that will clarify the development below.

First, consider the second layer model. We will conduct our primary evaluations on multi-class classification problems, where output vectors $\mathbf{y}$ encode target classes by indicator vectors $\mathbf{y} \in \{0,1\}^{m\times 1}$ such that $\mathbf{y}'\mathbf{1} = 1$. Although it is common to adopt a softmax transfer for $\mathbf{f}_2$ in such a case, it is also useful to consider a perceptron model defined by $\mathbf{f}_2(\hat{\mathbf{z}}) = \mathrm{indmax}(\hat{\mathbf{z}})$ such that $\mathrm{indmax}(\hat{\mathbf{z}}) = \mathbf{1}_i$ (vector of all 0s except a 1 in the ith position) where $\hat{z}_i \geq \hat{z}_l$ for all l. Therefore, for multi-class classification, we will simply adopt the standard large-margin multi-class loss (Crammer & Singer, 2001):

$$L_2(\hat{\mathbf{z}}, \mathbf{y}) \quad = \quad \max(\mathbf{1} - \mathbf{y} + \hat{\mathbf{z}} - \mathbf{1}\mathbf{y}'\hat{\mathbf{z}}). \tag{3.3}$$

Intuitively, if $y_c = 1$ is the correct label, this loss encourages the response $\hat{z}_c = \mathbf{y}'\hat{\mathbf{z}}$ on the correct label to be a margin greater than the response $\hat{z}_i$ on any other label $i \neq$

c. Second, consider the first layer model. Although the loss (3.3) has proved to be highly successful for multi-class classification problems, it is not suitable for the first layer because it assumes there is only a *single* target component active in any latent vector $\boldsymbol{\phi}$; i.e. $\boldsymbol{\phi}'\mathbf{1} = 1$. Although some work has considered learning a latent *clustering* in a two-layer architecture (Joulin & Bach, 2012; Joulin et al., 2010; Guo & Schuurmans, 2007; Tishby et al., 1999), such an approach is not able to capture the latent sparse code of a classical PFN or RBM in a reasonable way: using clustering to simulate a multi-dimensional sparse code causes exponential blow-up in the number of latent classes required. Therefore, we instead adopt a *multi-label* perceptron model for the first layer, defined by the transfer function $\mathbf{f}_1(\hat{\boldsymbol{\psi}}) = \text{step}(\hat{\boldsymbol{\psi}})$ applied componentwise to the response vector $\hat{\boldsymbol{\psi}}$; i.e. $\text{step}(\hat{\psi}_i) = 1$ if $\hat{\psi}_i > 0$, 0 otherwise. Although several loss formulations exist for multi-label classification (Fuernkranz et al., 2008; Guo & Schuurmans, 2011), we adopt the following:

$$\begin{aligned} L_1(\hat{\boldsymbol{\psi}}, \boldsymbol{\phi}) = & \quad \max(\mathbf{1} - \boldsymbol{\phi} + \hat{\boldsymbol{\psi}}\boldsymbol{\phi}'\mathbf{1} - \mathbf{1}\boldsymbol{\phi}'\hat{\boldsymbol{\psi}}) \\ \equiv & \quad \max\left((\mathbf{1} - \boldsymbol{\phi})/(\boldsymbol{\phi}'\mathbf{1}) + \hat{\boldsymbol{\psi}} - \mathbf{1}\boldsymbol{\phi}'\hat{\boldsymbol{\psi}}/(\boldsymbol{\phi}'\mathbf{1})\right). \end{aligned} \tag{3.4}$$

Intuitively, this loss encourages the average response on the active labels, $\boldsymbol{\phi}'\hat{\boldsymbol{\psi}}/(\boldsymbol{\phi}'\mathbf{1})$, to exceed the response $\hat{\psi}_i$ on any inactive label i, $\phi_i = 0$, by some margin, while also encouraging the response on any active label to match the average of the active responses. Despite their simplicity, large-margin multi-label losses have proved to be highly successful in practice (Fuernkranz et al., 2008; Guo & Schuurmans, 2011). Therefore, the overall architecture we investigate embeds two nonlinear conditionals around a non-trivial latent layer.

3.3 Equivalent Reformulation

The main contribution of this section is to show that the training problem (3.2) has an exact equivalent reformulation. To demonstrate this reformulation, we first

need to establish the key observation that problem (3.2) can be re-expressed in terms of a *kernel matrix* between latent representation vectors. Importantly, this reformulation allows the problem to be re-expressed in terms of an optimization objective that is jointly convex in all participating variables. We establish this key intermediate result in this section in three steps: first, by re-expressing the latent representation in terms of a latent kernel; second, by reformulating the second layer objective; and third, by reformulating the first layer objective by exploiting large-margin formulation outlined in Section 3.2.1. Below let $K = X'X$ denote the kernel matrix over the input data, let $\mathrm{Im}(N)$ denote the row space of N, and let and † denote Moore-Penrose pseudo-inverse.

First, simply define $N = \Phi'\Phi$. Next, re-express the second layer objective F_2 in (3.1):

Lemma 1. For any fixed Φ, letting $N = \Phi'\Phi$, it follows that

$$\min_V F_2(\Phi, V) \quad = \quad \min_{B \in \mathrm{Im}(N)} L_2(B, Y) + \tfrac{\beta}{2} \operatorname{tr}(BN^\dagger B'). \tag{3.5}$$

Proof. The result follows from the following sequence of equivalence preserving transformations:

$$\begin{aligned} \min_V L_2(V\Phi, Y) + \tfrac{\beta}{2}\|V\|_F^2 \quad &= \quad \min_A L_2(AN, Y) + \tfrac{\beta}{2} \operatorname{tr}(ANA') && (3.6) \\ &= \quad \min_{B \in \mathrm{Im}(N)} L_2(B, Y) + \tfrac{\beta}{2} \operatorname{tr}(BN^\dagger B'), && (3.7) \end{aligned}$$

where, starting with the definition of F_2 in (3.1), the first equality in (3.6) follows from the representer theorem applied to $\|V\|_F^2$, which implies that the optimal V must be in the form of $V = A\Phi'$ for some $A \in \mathbb{R}^{m \times t}$ (Kimeldorf & Wahba, 1971); and finally, (3.7) follows by the change of variable $B = AN$. ∎

Note that Lemma 1 holds for any loss L_2. In fact, the result follows solely from the structure of the regularizer. However, we require L_2 to be convex in its first

argument to ensure a convex problem below. Convexity is indeed satisfied by the choice (3.3). Moreover, the term $\mathrm{tr}(BN^\dagger B')$ is jointly convex in N and B since it is a perspective function (Argyriou et al., 2008), hence the objective in (3.5) is jointly convex.

Next, we reformulate the first layer objective F_1 in (3.1). Since this transformation exploits specific structure in the first layer loss, we present the result in two parts: first, by showing how the desired outcome follows from a general assumption on L_1, then demonstrating that this assumption is satisfied by the specific large-margin multi-label loss defined in (3.4). To establish this result we will exploit the following augmented forms for the data and variables: let $\tilde{\Phi} = [\Phi, kI]$, $\tilde{N} = \tilde{\Phi}'\tilde{\Phi}$, $\tilde{\Psi} = [\hat{\Psi}, 0]$, $\tilde{X} = [X, 0]$, $\tilde{K} = \tilde{X}'\tilde{X}$, and $\tilde{t} = t + h$.

Lemma 2. For any L_1 if there exists a function $\tilde{L}_1$ such that $L_1(\hat{\Psi}, \Phi) = \tilde{L}_1(\tilde{\Phi}'\tilde{\Psi}, \tilde{\Phi}'\tilde{\Phi})$ for all $\hat{\Psi} \in \mathbb{R}^{h\times t}$ and $\Phi \in \{0,1\}^{h\times t}$, such that $\Phi'\mathbf{1} = \mathbf{1}k$, it then follows that

$$\min_W F_1(W, \Phi) \quad = \quad \min_{D\in \mathrm{Im}(\tilde{N})} \tilde{L}_1(D\tilde{K}, \tilde{N}) + \tfrac{\alpha}{2}\,\mathrm{tr}(D'\tilde{N}^\dagger D\tilde{K}). \tag{3.8}$$

Proof. Similar to above, consider the sequence of equivalence preserving transformations:

$$\begin{aligned}
\min_W L_1(WX, \Phi) + \tfrac{\alpha}{2}\|W\|_F^2 &= \min_W \tilde{L}_1(\tilde{\Phi}'W\tilde{X}, \tilde{\Phi}'\tilde{\Phi}) + \tfrac{\alpha}{2}\|W\|_F^2 && (3.9)\\
&= \min_C \tilde{L}_1(\tilde{\Phi}'\tilde{\Phi}C\tilde{X}'\tilde{X}, \tilde{\Phi}'\tilde{\Phi}) \\
&\quad + \tfrac{\beta}{2}\,\mathrm{tr}(\tilde{X}C'\tilde{\Phi}'\tilde{\Phi}C\tilde{X}') && (3.10)\\
&= \min_{D\in \mathrm{Im}(\tilde{N})} \tilde{L}_1(D\tilde{K}, \tilde{N}) + \tfrac{\alpha}{2}\,\mathrm{tr}(D'\tilde{N}^\dagger D\tilde{K}), && (3.11)
\end{aligned}$$

where, starting with the definition of F_1 in (3.1), the first equality (3.9) simply follows from the assumption. The second equality (3.10) follows from the representer theorem applied to $\|W\|_F^2$, which implies that the optimal W must be in

the form of $W = \tilde{\Phi} C \tilde{X}'$ for some $C \in \mathbb{R}^{\tilde{t} \times \tilde{t}}$ (using the fact that $\tilde{\Phi}$ has full rank h) (Kimeldorf & Wahba, 1971). Finally, (3.11) follows by the change of variable $D = \tilde{N} C$. ∎

Observe that the term $\mathrm{tr}(D' \tilde{N}^{\dagger} D \tilde{K})$ is again jointly convex in $\tilde{N}$ and D (also a perspective function), while it is easy to verify that $\tilde{L}_1(D\tilde{K}, \tilde{N})$ as defined in Lemma 3 below is also jointly convex in $\tilde{N}$ and D (Argyriou et al., 2008); therefore the objective in (3.8) is jointly convex.

Next, we show in Lemma 3 that the assumption of Lemma 2 is satisfied by the specific large-margin multi-label formulation in Section 3.2.1; that is, assume L_1 is given by the large-margin multi-label loss (3.4):

$$\begin{aligned} L_1(\hat{\Psi}, \Phi) &= \textstyle\sum_j \max\left(\mathbf{1} - \boldsymbol{\phi}_j + \hat{\boldsymbol{\psi}}_j \boldsymbol{\phi}_j' \mathbf{1} - \mathbf{1} \boldsymbol{\phi}_j' \hat{\boldsymbol{\psi}}_j\right) \\ &= \tau\big(\mathbf{1}\mathbf{1}' - \Phi + \hat{\Psi}\,\mathrm{diag}(\Phi'\mathbf{1}) - \mathbf{1}\,\mathrm{diag}(\Phi'\hat{\Psi})'\big), \\ &\qquad \text{such that } \tau(\Theta) := \textstyle\sum_j \max(\boldsymbol{\theta}_j), \end{aligned} \tag{3.12}$$

where we use $\hat{\boldsymbol{\psi}}_j$, $\boldsymbol{\phi}_j$ and $\boldsymbol{\theta}_j$ to denote the jth columns of $\hat{\Psi}$, Φ and Θ respectively.

Lemma 3. For the multi-label loss L_1 defined in (3.4), and for any fixed $\Phi \in \{0,1\}^{h \times t}$ where $\Phi'\mathbf{1} = \mathbf{1}k$, the definition $\tilde{L}_1(\tilde{\Phi}'\tilde{\Psi}, \tilde{\Phi}'\tilde{\Phi}) := \tau(\tilde{\Phi}'\tilde{\Psi} - \tilde{\Phi}'\tilde{\Phi}/k) + t - \mathrm{tr}(\tilde{\Phi}'\tilde{\Psi})$ using the augmentation above satisfies the property that $L_1(\hat{\Psi}, \Phi) = \tilde{L}_1(\tilde{\Phi}'\tilde{\Psi}, \tilde{\Phi}'\tilde{\Phi})$ for any $\hat{\Psi} \in \mathbb{R}^{h \times t}$.

Proof. Since $\Phi'\mathbf{1} = \mathbf{1}k$ we obtain a simplification of L_1:

$$\begin{aligned} L_1(\hat{\Psi}, \Phi) &= \tau\big(\mathbf{1}\mathbf{1}' - \Phi + k\hat{\Psi} - \mathbf{1}\,\mathrm{diag}(\Phi'\hat{\Psi})'\big) \\ &= \tau(k\hat{\Psi} - \Phi) + t - \mathrm{tr}(\tilde{\Phi}'\tilde{\Psi}). \end{aligned} \tag{3.13}$$

It only remains is to establish that $\tau(k\hat{\Psi} - \Phi) = \tau(\tilde{\Phi}'\tilde{\Psi} - \tilde{\Phi}'\tilde{\Phi}/k)$. To do so, consider the sequence of equivalence preserving transformations:

$$\tau(k\hat{\Psi} - \Phi) = \max_{\Lambda \in \mathbb{R}_+^{h \times \tilde{t}} : \Lambda' \mathbf{1} = \mathbf{1}} \operatorname{tr}\left(\Lambda'(k\tilde{\Psi} - \tilde{\Phi})\right) \tag{3.14}$$

$$= \max_{\Omega \in \mathbb{R}_+^{\tilde{t} \times \tilde{t}} : \Omega' \mathbf{1} = \mathbf{1}} \frac{1}{k} \operatorname{tr}\left(\Omega'\tilde{\Phi}'(k\tilde{\Psi} - \tilde{\Phi})\right) = \tau(\tilde{\Phi}'\tilde{\Psi} - \tilde{\Phi}'\tilde{\Phi}/k), \tag{3.15}$$

where the equalities in (3.14) and (3.15) follow from the definition of τ and the fact that linear maximizations over the simplex obtain their solutions at the vertices. To establish the equality between (3.14) and (3.15), since $\tilde{\Phi}$ embeds the submatrix kI, for any $\Lambda \in \mathbb{R}_+^{h \times \tilde{t}}$ there must exist an $\Omega \in \mathbb{R}_+^{\tilde{t} \times \tilde{t}}$ satisfying $\Lambda = \tilde{\Phi}\Omega/k$. Furthermore, these matrices satisfy $\Lambda'\mathbf{1} = 1$ iff $\Omega'\tilde{\Phi}'\mathbf{1}/k = \mathbf{1}$ iff $\Omega'\mathbf{1} = \mathbf{1}$. ∎

Combining Lemmas 1–3 yields the desired result of this section.

Theorem 1. For any second layer loss and any first layer loss that satisfies the assumption of Lemma 2 (for example the large-margin multi-label loss (3.4)), the following holds:

$$(3.2) = \min_{\{\tilde{N} : \exists \Phi \in \{0,1\}^{t \times h} \text{ s.t. } \Phi\mathbf{1} = \mathbf{1}k, \tilde{N} = \tilde{\Phi}'\tilde{\Phi}\}} \min_{B \in \operatorname{Im}(\tilde{N})} \min_{D \in \operatorname{Im}(\tilde{N})}$$
$$\tilde{L}_1(D\tilde{K}, \tilde{N}) + \tfrac{\alpha}{2} \operatorname{tr}(D'\tilde{N}^\dagger D\tilde{K}) + \gamma L_2(B, Y) + \tfrac{\gamma\beta}{2} \operatorname{tr}(B\tilde{N}^\dagger B'). \tag{3.16}$$

Theorem 1 follows immediately from Lemmas 1–3. Note that no relaxation has occurred thus far: the objective value of (3.16) matches that of (3.2). Not only has this reformulation resulted in (3.2) being entirely expressed in terms of the latent kernel matrix $\tilde{N}$, the objective in (3.16) is jointly convex in all participating unknowns, $\tilde{N}$, B and D. Unfortunately, the *constraints* in (3.16) are not convex.

3.4 Convex Relaxation

We first relax the problem by dropping the augmentation $\Phi \mapsto \tilde{\Phi}$ and working with the $t \times t$ variable $N = \Phi'\Phi$. Without the augmentation, Lemma 3 becomes a lower

bound (i.e. (3.14)≥(3.15)), hence a relaxation. To then achieve a convex form we further relax the constraints in (3.16). To do so, consider

$$\mathcal{N}_0 = \left\{N : \exists \Phi \in \{0,1\}^{t \times h} \text{ such that } \Phi \mathbf{1} = \mathbf{1}k \text{ and } N = \Phi'\Phi\right\} \quad (3.17)$$

$$\mathcal{N}_1 = \left\{N : N \in \{0,...,k\}^{t \times t}, N \succeq 0, \text{diag}(N) = \mathbf{1}k, \text{rank}(N) \leq h\right\} \quad (3.18)$$

$$\mathcal{N}_2 = \left\{N : N \geq 0, N \succeq 0, \text{diag}(N) = \mathbf{1}k\right\}, \quad (3.19)$$

where it is clear from the definitions that $\mathcal{N}_0 \subseteq \mathcal{N}_1 \subseteq \mathcal{N}_2$. (Here we use $N \succeq 0$ to also encode $N' = N$.) Note that the set $\mathcal{N}_0$ corresponds to the original set of constraints from (3.16). The set $\mathcal{N}_1$ simplifies the characterization of this constraint set on the resulting kernel matrices $N = \Phi'\Phi$. However, neither $\mathcal{N}_0$ nor $\mathcal{N}_1$ are convex. Therefore, we need to adopt the further relaxed set $\mathcal{N}_2$, which is convex. (Note that $N_{ij} \leq k$ has been implied by $N \succeq 0$ and $N_{ii} = k$ in $\mathcal{N}_2$.) By also dropping constraints $B \in \text{Im}(N)$ and $D \in \text{Im}(N)$ in (3.16), we obtain the following relaxed problem, which is jointly convex in N, B and D:

$$\min_{N \in \mathcal{N}_2} \min_{B \in \mathbb{R}^{t \times t}} \min_{D \in \mathbb{R}^{t \times t}} \tilde{L}_1(DK, N) + \tfrac{\alpha}{2} \text{tr}(D'N^\dagger DK) + \gamma L_2(B, Y) \tfrac{\gamma\beta}{2} \text{tr}(BN^\dagger B'). \quad (3.20)$$

3.5 Efficient Training Approach

Unfortunately, nonlinear semidefinite optimization problems in the form (3.20) are generally thought to be too expensive in practice despite their polynomial theoretical complexity (Nesterov & Nimirovskii, 1994; Boyd & Vandenberghe, 2004). Therefore, we develop an effective training algorithm that exploits problem structure to bypass the main computational bottlenecks. The key challenge is that $\mathcal{N}_2$ contains both semidefinite and affine constraints, and the pseudo-inverse $N^\dagger$ makes optimization over N difficult even for fixed B and D.

To mitigate these difficulties we first treat (3.20) as the reduced problem, $\min_{N \in \mathcal{N}_2} \mathcal{F}(N)$, where $\mathcal{F}$ is an implicit objective achieved by minimizing out B and D. Note

that $\mathcal{F}$ is still convex in N by the joint convexity of (3.20). To cope with the constraints on N we adopt the alternating direction method of multipliers (ADMM) (Boyd et al., 2010) as the main outer optimization procedure; see Algorithm 1. This approach allows one to divide $\mathcal{N}_2$ into two groups, $N \succeq 0$ and $\{N_{ij} \geq 0, N_{ii} = k\}$, yielding the augmented Lagrangian

$$\begin{aligned}\mathcal{L}(N, M, \Gamma) = \mathcal{F}(N) + \delta(N \succeq 0) + \delta(M_{ij} \geq 0, M_{ii} = k) - \langle \Gamma, N - M \rangle \\ + \tfrac{1}{2\mu} \|N - M\|_F^2 ,\end{aligned} \tag{3.21}$$

where $\mu > 0$ is a constant and δ is an indicator such that $\delta(\cdot) = 0$ if $\cdot$ is true, ∞ otherwise. In this procedure, Steps 4 and 5 cost $O(t^2)$ time; whereas the main bottleneck is Step 3, which involves minimizing $\mathcal{G}_T(N) := \mathcal{L}(N, M_{T-1}, \Gamma_{T-1})$ over $N \succeq 0$ for fixed M_{T-1} and Γ_{T-1}.

Boosting for Optimizing over the Positive Semidefinite Cone. To solve the problem in Step 3 we develop an efficient boosting procedure based on (Laue, 2012) that retains low rank iterates N_T while avoiding the need to determine $N^\dagger$ when computing $\mathcal{G}(N)$ and $\nabla\mathcal{G}(N)$; see Algorithm 2.

The key idea is to use a simple change of variable. For example, consider the first layer objective and let $\mathcal{G}_1(N) = \min_D \tilde{L}_1(DK, N) + \frac{\alpha}{2} \operatorname{tr}(D' N^\dagger DK)$. By defining $D = NC$, we obtain $\mathcal{G}_1(N) = \min_C \tilde{L}_1(NCK, N) + \frac{\alpha}{2} \operatorname{tr}(C' NCK)$, which no longer involves $N^\dagger$ but remains convex in C; this problem can be solved efficiently after a slight smoothing of the objective (Chapelle, 2007) (e.g. by LBFGS).

Moreover, the gradient $\nabla\mathcal{G}_1(N)$ can be readily computed given C^*. Applying the same technique to the second layer yields an efficient procedure for evaluating $\mathcal{G}(N)$ and $\nabla\mathcal{G}(N)$. Finally note that many of the matrix-vector multiplications in this procedure can be further accelerated by exploiting the low rank factorization of N maintained by the boosting algorithm.

Algorithm 1: ADMM to optimize $\mathcal{F}(N)$ for $N \in \mathcal{N}_2$.

Initialize: $M_0 = I$, $\Gamma_0 = \mathbf{0}$.

while $T = 1, 2, \ldots$ **do**

  $N_T \leftarrow \arg\min_{N \succeq 0} \mathcal{L}(N, M_{T-1}, \Gamma_{T-1})$, by using the boosting Algorithm 2.

  $M_T \leftarrow \arg\min_{M \succeq 0, M_{ii}=k} \mathcal{L}(N_T, M, \Gamma_{T-1})$, which has an efficient closed form solution.

  $\Gamma_T \leftarrow \Gamma_{T-1} + \frac{1}{\mu}(M_T - N_T)$; i.e. update the multipliers.

return N_T.

Additional Relaxation. One can further reduce computation cost by adopting additional relaxations to (3.20). For example, by dropping $N \succeq 0$ and relaxing $\operatorname{diag}(N) = \mathbf{1}k$ to $\operatorname{diag}(N) \leq \mathbf{1}k$, the objective can be written as $\min_{\{N \succeq 0, \max_i N_{ii} \leq k\}} \mathcal{F}(N)$. Since $\max_i N_{ii}$ is convex in N, it is well known that there must exist a constant $c_1 > 0$ such that the optimal N is also an optimal solution to $\min_{N \succeq 0} \mathcal{F}(N) + c_1 \left(\max_i N_{ii}\right)^2$. While $\max_i N_{ii}$ is not smooth, one can further smooth it with a softmax, to instead solve $\min_{N \succeq 0} \mathcal{F}(N) + c_1 \left(\log \sum_i \exp(c_2 N_{ii})\right)^2$ for some large c_2. This formulation avoids the need for ADMM entirely and can be directly solved by Algorithm 2.

3.6 Experimental Evaluation

To investigate the effectiveness of the proposed relaxation scheme for training a two-layer conditional model, we conducted a number of experiments to compare learning quality against baseline methods. Note that, given an optimal solution N, B and D to (3.20), an approximate solution to the original problem (3.2) can be recovered heuristically by first rounding N to obtain Φ, then recovering W and V, as shown in Lemmas 1 and 2. However, since our primary objective is to determine

Algorithm 2: Boosting algorithm to optimize $\mathcal{G}(N)$ for $N \succeq 0$.

Initialize: $N_0 \leftarrow \mathbf{0}$, $H_0 \leftarrow [\,]$ (empty set).
while $T = 1, 2, \ldots$ **do**
  Find the smallest arithmetic eigenvalue of $\nabla\mathcal{G}(N_{T-1})$, and its eigenvector $\mathbf{h}_T$.
  Conic search by LBFGS: $(a_T, b_T) \leftarrow \min_{a\geq 0, b\geq 0} \mathcal{G}(aN_{T-1} + b\mathbf{h}_T\mathbf{h}_T')$.
  Local search by LBFGS: $H_T \leftarrow \text{local\_min}_H \mathcal{G}(HH')$ initialized by $H = (\sqrt{a}H_{T-1}, \sqrt{b}\mathbf{h}_T)$.
  Set $N_T \leftarrow H_T H_T'$; **break** if stopping criterion met.
return N_T.

whether *any* convex relaxation of a two-layer model can even compete with one-layer or locally trained two-layer models (rather than evaluate heuristic rounding schemes), we consider a transductive evaluation that does not require any further modification of N, B and D. In such a set-up, training data is divided into a labeled and unlabeled portion, where the method receives $X = [X_\ell, X_u]$ and Y_ℓ, and at test time the resulting predictions $\hat{Y}_u$ are evaluated against the held-out labels Y_u.

Methods. We compared the proposed convex relaxation scheme (CVX2) against the following methods: simple alternating minimization of the same two-layer model (3.2) (LOC2), a one-layer linear SVM trained on the labeled data (SVM1), the transductive one-layer SVM methods of (Joachims, 1999) (TSJ1) and (Sindhwani & Keerthi, 2006) (TSS1), and the transductive latent clustering method of (Joulin & Bach, 2012; Joulin et al., 2010) (TJB2), which is also a two-layer model. Linear input kernels were used for all methods (standard in most deep learning models) to control the comparison between one and two-layer models. Our experiments were conducted with the following common protocol: First, the data was split into a separate training and test set. Then the parameters of each procedure were

optimized by a three-fold cross validation on the training set. Once the optimal parameters were selected, they were fixed and used on the test set. For transductive procedures, the same three training sets from the first phase were used, but then combined with ten new test sets drawn from the disjoint test data (hence 30 overall) for the final evaluation. At no point were test examples used to select any parameters for any of the methods. We considered different proportions between labeled/unlabeled data; namely, 100/100 and 200/200.

Synthetic Experiments. We initially ran a proof of concept experiment on three binary labeled artificial data sets depicted in Figure 3.2, Figure 3.3 and Figure 3.4 (showing data set sizes $n \times t$) with 100/100 labeled/unlabeled training points. Here the goal was simply to determine whether the relaxed two-layer training method could preserve sufficient structure to overcome the limits of a one-layer architecture. Clearly, none of the data sets in Figure 3.2, Figure 3.3 and Figure 3.4 are adequately modeled by a one-layer architecture (that does not cheat and use a nonlinear kernel).

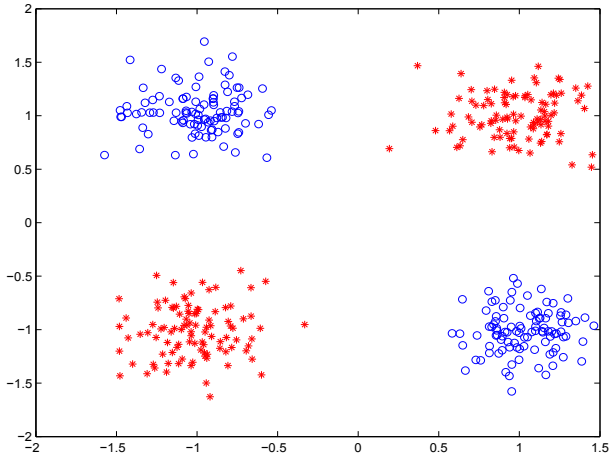


Figure 3.2: "Xor" dataset

The results are shown in the Table 3.1.

As expected, the one-layer models SVM1 and TSS1 were unable to capture any useful classification structure in these problems. (TSJ1 behaves similarly to

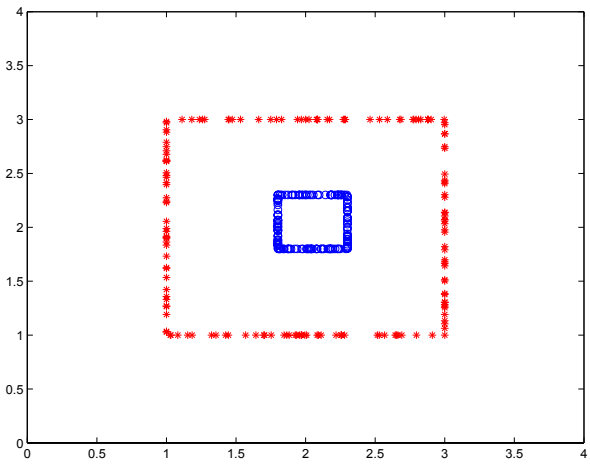


Figure 3.3: “boxes” dataset

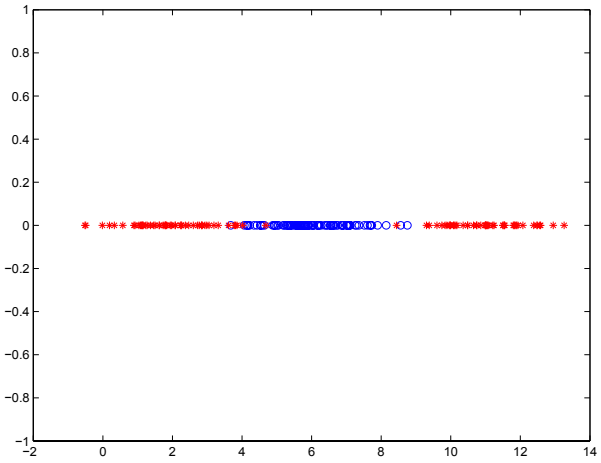


Figure 3.4: “interval” dataset

TSS1.) The results obtained by CVX2, on the other hand, are encouraging. In these data sets, CVX2 is easily able to capture latent nonlinearities while outperforming the locally trained LOC2. Although LOC2 is effective in the first two cases, it exhibits weaker test accuracy while failing on the third data set. The two-layer method TJB2 exhibited convergence difficulties on these problems that prevented

	XOR	BOXES	INTER
TJB2	49.8 $\pm$0.7	45.7 $\pm$0.6	49.3 $\pm$1.3
TSS1	50.2 $\pm$1.2	35.7 $\pm$1.3	42.6 $\pm$3.9
SVM1	50.3 $\pm$1.1	31.4 $\pm$0.5	50.0 $\pm$0.0
LOC2	4.2 $\pm$0.9	11.4 $\pm$0.6	50.0 $\pm$0.0
CVX2	0.2 $\pm$0.1	10.1 $\pm$0.4	20.0 $\pm$2.4

Table 3.1: Mean test misclassification error % ($\pm$ stdev) for artificial datasets

reasonable results.

Figure 3.5 shows the output kernel of a test sample for the Xor dataset. If we denote each of classes in the Xor dataset by P and N, one can observe that the classes P and N are each composed of two subclasses, denoted $\{P_1, P_2\}$ and $\{N_1, N_2\}$ respectively. To allow convenient visualization, the instances with indexes from top to down and left to right in the matrix belong to subclasses P_1, N_1, P_2 and N_2 respectively. Therefore, the kernel demonstrates that the latent representation acquired by CVX2 captures the structure very well. The corresponding weight, latent matrix and response matrix are given in Appendix A.1.2 which show clearly how the discrimination works.

Experiments on "Real" Data Sets. Next, we conducted experiments on real data sets to determine whether the advantages in controlled synthetic settings could translate into useful results in a more realistic scenario. For these experiments we used a collection of binary labeled data sets: USPS, COIL and G241N from (Chapelle et al., 2006), Letter from (Lichman, 2013), MNIST from (LeCun & Cortes, 2010), and CIFAR-100 from (Krizhevsky, 2009). (See Appendix A.1.2 for further details.)

The results are shown in Tables 3.2 and 3.3 for the labeled/unlabeled proportions 100/100 and 200/200 respectively.

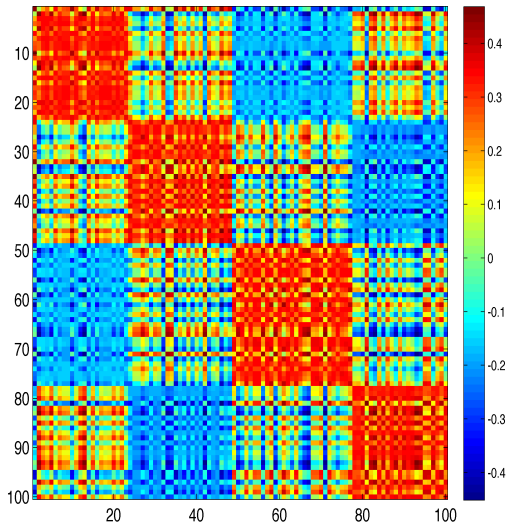


Figure 3.5: The latent kernel for "Xor" test data

	MNIST	USPS	Letter	COIL	CIFAR	G241N
TJB2	19.3 $\pm$1.2	53.2 $\pm$2.9	20.4 $\pm$2.1	30.6 $\pm$0.8	29.2 $\pm$2.1	26.3 $\pm$0.8
LOC2	19.3 $\pm$1.0	13.9 $\pm$1.1	10.4 $\pm$0.6	18.0 $\pm$0.5	31.8 $\pm$0.9	41.6 $\pm$0.9
SVM1	16.2 $\pm$0.7	11.6 $\pm$0.5	6.2 $\pm$0.4	16.9 $\pm$0.6	27.6 $\pm$0.9	27.1 $\pm$0.9
TSS1	13.7 $\pm$0.8	11.1 $\pm$0.5	5.9 $\pm$0.5	17.5 $\pm$0.6	26.7 $\pm$0.7	25.1 $\pm$0.8
TSJ1	14.6 $\pm$0.7	12.1 $\pm$0.4	5.6 $\pm$0.5	17.2 $\pm$0.6	26.6 $\pm$0.8	24.4 $\pm$0.7
CVX2	9.2 $\pm$0.6	9.2 $\pm$0.5	5.1 $\pm$0.5	13.8 $\pm$0.6	26.5 $\pm$0.8	25.2 $\pm$1.0

Table 3.2: Mean test misclassification error % ($\pm$ stdev) for 100/100 labeled/unlabeled.

The relaxed two-layer method CVX2 again demonstrates effective results, although some data sets caused difficulty for all methods. The data sets can be divided into two groups, (MNIST, USPS, COIL) versus (Letter, CIFAR, G241N). In the first group, two-layer modeling demonstrates a clear advantage: CVX2 outperforms SVM1 by a significant margin. Note that this advantage must be due to two-

	MNIST	USPS	Letter	COIL	CIFAR	G241N
TJB2	13.7 ± 0.6	46.6 ± 1.0	14.0 ± 2.6	45.0 ± 0.8	30.4 ± 1.9	22.4 ± 0.5
LOC2	16.3 ± 0.6	9.7 ± 0.5	8.5 ± 0.6	12.8 ± 0.6	28.2 ± 0.9	40.4 ± 0.7
SVM1	11.2 ± 0.4	10.7 ± 0.4	5.0 ± 0.3	15.6 ± 0.5	25.5 ± 0.6	22.9 ± 0.5
TSS1	11.4 ± 0.5	11.3 ± 0.4	4.4 ± 0.3	14.9 ± 0.4	24.0 ± 0.6	23.7 ± 0.5
TSJ1	12.3 ± 0.5	11.8 ± 0.4	4.8 ± 0.3	13.5 ± 0.4	23.9 ± 0.5	22.2 ± 0.6
CVX2	8.8 ± 0.4	6.6 ± 0.4	3.8 ± 0.3	8.2 ± 0.4	22.8 ± 0.6	20.3 ± 0.5

Table 3.3: Mean test misclassification error % ($\pm$ stdev) for 200/200 labeled/unlabeled.

layer versus one-layer modeling, since the transductive SVM methods TSS1 and TSJ1 demonstrate no advantage over SVM1. For the second group, the effectiveness of SVM1 demonstrates that only minor gains can be possible via transductive or two-layer extensions, although some gains are realized. The locally trained two-layer model LOC2 performed quite poorly in all cases. Unfortunately, the convex latent clustering method TJB2 was also not competitive on any of these data sets. Overall, CVX2 appears to demonstrate useful promise as a two-layer modeling approach.

3.7 Conclusion

We have introduced a new convex approach to two-layer conditional modeling by reformulating the original non-convex problem in terms of a latent output kernel over intermediate feature representations. The proposed model can accommodate latent feature representations that go well beyond a latent clustering, extending current convex approaches. A semidefinite relaxation of the latent kernel allows a reasonable implementation that is able to demonstrate advantages over single-layer models and local training methods. From a deep learning perspective, this work

demonstrates that trainable latent layers can be expressed in terms of reproducing kernel Hilbert spaces, while large margin methods can be usefully applied to multi-layer prediction architectures. Although the nonlinear semidefinite programs are difficult to solve, we also designed an efficient algorithm adopting a low-rank approach. We then empirically demonstrated that the structure of latent modeling is preserved by showing promising results.

Chapter 4

Tractable Multi-Layer Modeling

We demonstrated in Chapter 3 that an adaptive hidden layer could be expressed as a formulation that is jointly convex in weight parameters and the output latent variable kernel. Although this shows clearly how latent kernel learning can be formulated, the two-layer convex model remained restricted to a single adaptive layer, with no clear paths suggested for a multi-layer extension.

In this chapter, we develop a new architecture for nested nonlinearities that allows arbitrarily deep compositions to be trained to global optimality. Similarly to the model in Chapter 3, we represent the nonlinearities in each layer through nonlinear losses that are extended to the multi-layer architecture; these new losses are combined to obtain a joint form, after which we apply a convex relaxation to obtain the final convex formulation. The approach admits both parametric and nonparametric forms through the use of a new type of kernel to represent each latent layer. The outcome is a fully convex formulation that is able to capture compositions of trainable nonlinear layers to arbitrary depth. We also carefully design a novel efficient algorithm for this new model. The work in this chapter was published in Neural Information Processing Systems conference(Aslan et al., 2014).

4.1 Introduction

One of the key contributions of Chapter 3 was to develop a convex formulation of one hidden layer training utilizing output kernels. One focus of this thesis therefore is to ground deep learning in kernel-based approaches, which offer a potentially easier path to achieving a simple computational understanding. Kernels (Kimeldorf & Wahba, 1971) have had a significant impact in machine learning, partly because they offer flexible modeling capability without sacrificing convexity in common training scenarios (Schoelkopf & Smola, 2002). Given the convexity of the resulting training formulations, suboptimal local minima and plateaus are eliminated while reliable computational procedures are widely available. A common misconception about kernel methods is that they are inherently "shallow" (Bengio, 2009), but depth is an aspect of how such methods are used and not an intrinsic property. For example, (Cho & Saul, 2010) demonstrates how nested compositions of kernels can be incorporated in a convex training formulation, which can be interpreted as learning over a (fixed) composition of hidden layers with infinite features. Other work has formulated adaptive learning of nested kernels, albeit by sacrificing convexity (Zhuang et al., 2011). More recently, (Joulin & Bach, 2012; Joulin et al., 2012) has considered learning kernel representations of latent clusters, achieving convex formulations under some relaxations.

After providing the background material about the multi-layer and showing the significant difficulty of extension of two-layer convex model introduced in Chapter 3 to multi-layer convex model in Section 4.2, we introduce a key novel regularization trick that results in a new output kernel type that yields a convex reformulation for arbitrary number of layers in Section 4.3. We exploit the structure of the new formulation presented in Section 4.3 and develop a novel efficient algorithm with further advantages compared to the algorithm in Chapter 3 in Section 4.4. We also present appealing empirical results in Section 4.5 which shows that the multi-layer structure is preserved after the relaxations.

4.2 Background

We consider a multi-layer conditional model where the input $\mathbf{x}_i$ is n dimensional and the output $\mathbf{y}_i \in \{0,1\}^m$ is a multi-label target vector over m labels. For concreteness, consider a three-layer model (Figure 4.1). Here, the output of the first hidden layer is determined by multiplying the input, $\mathbf{x}_i$, with a weight matrix $W \in \mathbb{R}^{h \times n}$ and passing the result through a nonlinear transfer σ_1, yielding $\boldsymbol{\phi}_i = \sigma_1(W\mathbf{x}_i)$.

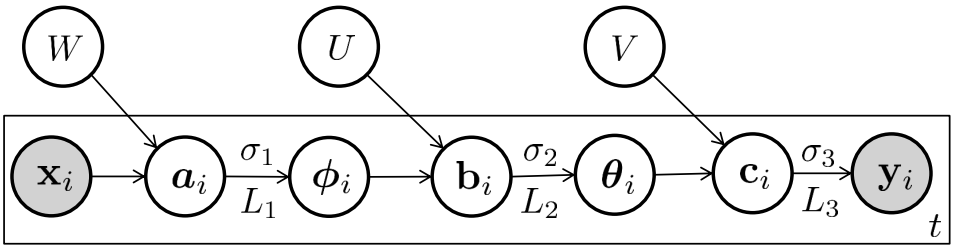


Figure 4.1: Multi-layer conditional models

The output of the second layer is determined by multiplying the first layer output, $\boldsymbol{\phi}_i$, with a second weight matrix $U \in \mathbb{R}^{h' \times h}$ and passing the result through a nonlinear transfer σ_2, yielding $\boldsymbol{\theta}_i = \sigma_2(U\boldsymbol{\phi}_i)$, etc. The final output is then determined via $\hat{\mathbf{y}}_i = \sigma_3(V\boldsymbol{\theta}_i)$, for $V \in \mathbb{R}^{m \times h'}$. For simplicity, we will set $h' = h$.

The goal of training is to find the weight matrices, W, U, and V, that minimize a training objective defined over the training data (with regularization). In particular, we assume the availability of t training examples $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^t$, and denote the feature matrix $X := (\mathbf{x}_1, \ldots, \mathbf{x}_t) \in \mathbb{R}^{n \times t}$ and the label matrix $Y := (\mathbf{y}_1, \ldots, \mathbf{y}_t) \in \mathbb{R}^{m \times t}$ respectively. One of the key challenges for training arises from the fact that the latent variables $\Phi := (\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_t)$ and $\Theta := (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_t)$ are unobserved.

To introduce our main development, we begin with a reconsideration of the formulation from Chapter 3, which proposed a convex formulation of a simpler two-layer model. Although the techniques proposed in Chapter 3 are intrinsically restricted to two layers, we will eventually show how this barrier can be surpassed

through the introduction of a new tool—normalized output kernels. However we first need to provide a more general treatment of the three main obstacles to obtaining a convex training formulation for multi-layer architectures like Figure 4.1.

First Obstacle: Nonlinear Transfers. The first key obstacle arises from the presence of the transfer functions, σ_i, which provide the nonlinearity of the model. If, as in Chapter 3, one optimizes rather than marginalizes over hidden layer values, Φ and Θ, a generalized training objective for a multi-layer architecture (Figure 4.1) can be expressed similarly as:

$$\min_{W,U,V,\Phi,\Theta} L_1(WX,\Phi)+\tfrac{1}{2}\|W\|^2+L_2(U\Phi,\Theta)+\tfrac{1}{2}\|U\|^2+L_3(V\Theta,Y)+\tfrac{1}{2}\|V\|^2 \quad (4.1)$$

where $\|W\|^2$, $\|U\|^2$ and $\|V\|^2$ are regularizers such that the norm is the Frobenius norm and the nonlinear loss L_1 bridges the nonlinearity introduced by σ_1 and L_2 bridges the nonlinearity introduced by σ_2. These losses are assumed to be convex in their first argument. For clarity we have omitted the regularization parameters, relative weightings between layers, and offset weights from the model. These are obviously important in practice, but they play no key role in the technical development and removing them simplifies the expressions.

Unfortunately, even though the overall objective (4.1) is convex in the weight matrices (W,U,V) *given* (Φ,Θ), it is not *jointly* convex in all participating variables due to the interaction between the latent variables (Φ,Θ) and the weight matrices (W,U,V).

Second Obstacle: Bilinear Interaction. The second key obstacle arises from the bilinear interaction between the latent variables and weight matrices in (4.1). To overcome this obstacle, consider a single connecting layer, which consists of an input matrix (Φ) and output matrix (Θ) and associated weight matrix (U):

$$\min_U L(U\Phi,\Theta)+\tfrac{1}{2}\|U\|^2. \quad (4.2)$$

By the representer theorem, it follows that the optimal U can be expressed as $U = A\Phi'$ for some $A \in \mathbb{R}^{m \times t}$. Denote the linear response $Z = U\Phi = A\Phi'\Phi = AK$ where $K = \Phi'\Phi$ is the input kernel matrix. Then $\operatorname{tr}(UU') = \operatorname{tr}(AKA') = \operatorname{tr}(AKK^\dagger KA') = \operatorname{tr}(ZK^\dagger Z')$, where $K^\dagger$ is the Moore-Penrose pseudo-inverse (recall $KK^\dagger K = K$ and $K^\dagger KK^\dagger = K^\dagger$), therefore

$$(4.2) = \min_Z L(Z, \Theta) + \tfrac{1}{2} \operatorname{tr}(ZK^\dagger Z'), \quad \text{where } Z \in \mathbb{R}K. \tag{4.3}$$

This is essentially the value regularization framework (Rifkin & Lippert, 2007). Importantly, the objective in (4.3) is *jointly convex* in Z and K, since $\operatorname{tr}(ZK^\dagger Z)$ is a perspective function (Argyriou et al., 2008). Therefore, although the single layer model is not jointly convex in the input features Φ and model parameters U, it is convex in the equivalent reparameterization (K, Z) given Θ. This is the technique used in Chapter 3 for the output layer. Finally note that Z satisfies the constraint $Z \in \mathbb{R}^{m \times n}\Phi := \{U\Phi : U \in \mathbb{R}^{m \times n}\}$, which we will write as $Z \in \mathbb{R}\Phi$ for convenience. Clearly it is equivalent to $Z \in \mathbb{R}K$.

Third Obstacle: Joint Input-Output Optimization. The third key obstacle is that each of the latent variables, Φ and Θ, simultaneously serve as the inputs and output targets for successive layers. Therefore, it is necessary to reformulate the connecting problem (4.2) so that it is jointly convex in all three components, U, Φ and Θ; and unfortunately (4.3) is not convex in Θ. Although this appears to be an insurmountable obstacle in general, Chapter 3 proposed an exact reformulation in the case when Θ is boolean valued (consistent with the probabilistic assumptions underlying a PFM or RBM) by assuming the loss function satisfies an additional postulate.

Postulate 1. $L(Z, \Theta)$ can be rewritten as $L^u(\Theta'Z, \Theta'\Theta)$ for L^u *jointly* convex in both arguments.

Intuitively, this assumption allows the loss to be parameterized in terms of the

propensity matrix $\Theta' Z$ and the **u**nnormalized *output kernel* $\Theta'\Theta$ (hence the superscript of L^u). That is, the (i, j)-th component of $\Theta' Z$ stands for the linear response value of example j with respect to the label of the example i. The j-th column therefore encodes the propensity of example j to all other examples. This reparameterization is critical because it bypasses the linear response value, and relies solely on the relationship between *pairs* of examples. Chapter 3 proposed a particular multi-label prediction loss that satisfies Postulate 1 for boolean target vectors $\boldsymbol{\theta}_i$; we propose an alternative below.

Recall that letting the $Z = U\Theta$, the objective function in (4.2) can be rewritten as

$$L^u(\Theta' U\Phi, \Theta'\Theta) + \frac{1}{2}\|U\|^2, \tag{4.4}$$

using Postulate 1. Then by the representer theorem, the optimal U can be expressed by $U = \Theta A\Phi'$ for some matrix A. Denoting $S := \Theta' Z = \Theta' U\Phi$ and $N := \Theta'\Theta$, the objective becomes

$$\min_U L^u(S, N) + \frac{1}{2}\operatorname{tr}(K^\dagger S' N^\dagger S), \tag{4.5}$$

since

$$\begin{aligned}
\operatorname{tr}(K^\dagger S' N^\dagger S) &= \operatorname{tr}(K^\dagger \Phi' U'\Theta N^\dagger \Theta' U\Phi) = \operatorname{tr}(K^\dagger \Phi'\Phi A'\Theta'\Theta N^\dagger \Theta'\Theta A\Phi'\Phi) && (4.6)\\
&= \operatorname{tr}(K^\dagger K A' N N^\dagger N A K) = \operatorname{tr}(K A' N A) \text{ (using } KK^\dagger K = K) && (4.7)\\
&= \operatorname{tr}(\Phi'\Phi A'\Theta'\Theta A) = \|\Theta A\Phi'\|^2 = \|U\|^2. && (4.8)
\end{aligned}$$

Therefore, Postulate 1 allows (4.2) to be re-expressed in a form that is jointly convex in the propensity matrix S and output kernel N. Given that N is a discrete but positive semidefinite matrix, a final relaxation is required to achieve a convex training problem.

Postulate 2. The domain of $N = \Theta'\Theta$ can be relaxed to a convex set preserving structure.

Below we will introduce an improved scheme for such relaxation. Although these developments supported the formulation of Chapter 3, they appear insufficient for deeper models. For example, by applying (4.3) and (4.5) to the three-layer model of Figure 4.1, one obtains

$$L_1^u(S_1, N_1) + \tfrac{1}{2}\operatorname{tr}(K^\dagger S_1' N_1^\dagger S_1) + L_2^u(S_2, N_2) + \tfrac{1}{2}\operatorname{tr}(N_1^\dagger S_2' N_2^\dagger S_2) \\ + L_3(Z_3, Y) + \tfrac{1}{2}\operatorname{tr}(Z_3 N_2^\dagger Z_3'),$$

where $N_1 = \Phi'\Phi$ and $N_2 = \Theta'\Theta$ are two latent kernels imposed between the input and output. Unfortunately, this objective is *not* jointly convex, since $\operatorname{tr}(N_1^\dagger S_2' N_2^\dagger S_2)$ is not jointly convex in (N_1, S_2, N_2), hence the approach of Chapter 3 cannot extend beyond a single hidden layer.

4.3 Multi-layer Convex Modeling via Normalized Kernels

Although obtaining a convex formulation for general multi-layer models appears to be a significant challenge, progress can be made by considering an alternative approach. The failure of the previous development in Chapter 3 can be traced back to (4.2), which eventually causes the coupled, non-convex regularization to occur between connected latent kernels. A natural response therefore is to reconsider the original regularization scheme, keeping in mind that the representer theorem must still be supported. One such regularization scheme appears has been investigated in the clustering literature (Peng & Wei, 2007; Cheng et al., 2013), which suggests a reformulation of the connecting model (4.2) using value regularization (Rifkin & Lippert, 2007):

$$\min_U L(U\Phi, \Theta) + \tfrac{1}{2}\|\Theta' U\|^2. \tag{4.9}$$

Here $\|\Theta' U\|^2$ replaces $\|U\|^2$ from (4.2). The significance of this reformulation is that it still admits the representer theorem, implying that the optimal U has the form

$U = (\Theta\Theta')^\dagger A\Phi'$ for some $A \in \mathbb{R}^{m\times n}$. Since Θ generally has full row rank (*i.e.* more examples than labels), one may execute a change of variables $A = \Theta B$. Such a substitution leads to the regularizer $\left\|\Theta'(\Theta\Theta')^\dagger\Theta B\Phi'\right\|^2$, which can be expressed in terms of the *normalized output kernel* (Peng & Wei, 2007):

$$M := \Theta'(\Theta\Theta')^\dagger\Theta. \tag{4.10}$$

The term $(\Theta\Theta')^\dagger$ normalizes the spectrum of the kernel $\Theta'\Theta$, and it is obvious that all eigen-values of M are either 0 or 1, *i.e.* $M^2 = M$ (Peng & Wei, 2007). The regularizer can be finally written:

$$\|MB\Phi'\|^2 = \operatorname{tr}(MBKB'M) = \operatorname{tr}(MBKK^\dagger KB'M) = \operatorname{tr}(SK^\dagger S'), \tag{4.11}$$

where $S := MBK$. It is easy to show $S = \Theta'Z = \Theta'U\Phi$, which is exactly the propensity matrix.

As before, to achieve a convex training formulation, additional structure must be postulated on the loss function, but now allowing convenient expression in terms of normalized latent kernels.

Postulate 3. The loss $L(Z, \Theta)$ can be written as $L^n(\Theta'Z, \Theta'(\Theta\Theta')^\dagger\Theta)$ where L^n is *jointly* convex in both arguments. Here we write L^n to emphasize the use of normalized kernels.

Under Postulate 3, an alternative convex objective can be achieved

$$L^n(S, M) + \tfrac{1}{2}\operatorname{tr}(SK^\dagger S'), \quad \text{where} \quad S \in M\mathbb{R}K. \tag{4.12}$$

Crucially, this objective is now jointly convex in S, M and K; in comparison to (4.5), the normalization has removed the output kernel from the regularizer. The feasible region $\{(S, M, K) : M \succeq \mathbf{0}, K \succeq \mathbf{0}, S \in M\mathbb{R}K\}$ is also convex (see Appendix A.2.1). Applying (4.12) to the first two layers and (4.3) to the output layer, a fully convex objective for a multi-layer model (*e.g.*, as in Figure 4.1) is

obtained:

$$L_1^n(S_1, M_1) + \tfrac{1}{2}\operatorname{tr}(S_1 K^\dagger S_1') + L_2^n(S_2, M_2) + \tfrac{1}{2}\operatorname{tr}(S_2 M_1^\dagger S_2') \\ + L_3(Z_3, Y) + \tfrac{1}{2}\operatorname{tr}(Z_3 M_2^\dagger Z_3'), \tag{4.13}$$

where $S_1 \in M_1\mathbb{R}K$, $S_2 \in M_2\mathbb{R}M_1$, and $Z_3 \in \mathbb{R}M_2$. (Clearly the first layer can still use (4.5) with an unnormalized output kernel N_1 since its input X is observed.) All that remains is to design a convex relaxation of the domain of M (for Postulate 2) and to design the loss L^n (for Postulate 3).

4.3.1 Convex Relaxation of the Domain of Output Kernels M

Based on its definition (4.10), M has a non-convex domain. Ideally one should design convex relaxations for each domain of Θ. However, M exhibits some nice properties for *any* Θ:

$$M \succeq \mathbf{0}, \quad M \preceq I, \quad \operatorname{tr}(M) = \operatorname{tr}((\Theta\Theta')^\dagger(\Theta\Theta')) = \operatorname{rank}(\Theta\Theta') = \operatorname{rank}(\Theta). \tag{4.14}$$

Here I is the identity matrix, and we also use $M \succeq \mathbf{0}$ to encode $M' = M$. Therefore, $\operatorname{tr}(M)$ provides a convenient proxy for controlling the rank of the latent representation, *i.e.* the number of hidden nodes in a layer. Given a specified number of hidden nodes h, we may enforce $\operatorname{tr}(M) = h$. The main relaxation introduced here is replacing the eigenvalue constraint $\lambda_i(M) \in \{0, 1\}$ (implied by $M^2 = M$) with $0 \leq \lambda_i(M) \leq 1$. Such a relaxation retains sufficient structure to allow, *e.g.*, a 2-approximation of optimal clustering to be preserved even by only imposing spectral constraints (Peng & Wei, 2007). Experimental results below further demonstrate that nesting preserves sufficient structure, even with relaxation, to capture relationships that cannot be recovered by shallower architectures.

More refined constraints can be included to better account for the domain of Θ. For example, if Θ expresses target values for a multiclass classification (*i.e.* $\Theta_{ij} \in \{0, 1\}$, $\Theta'\mathbf{1} = \mathbf{1}$ where $\mathbf{1}$ is a vector of all one's), we further have $M_{ij} \geq 0$

and $M\mathbf{1} = \mathbf{1}$. If Θ corresponds to multilabel classification where each example belongs to exactly k (out of the h) labels (*i.e.* $\Theta \in \{0,1\}^{h\times t}$, $\Theta'\mathbf{1} = k\mathbf{1}$), then M can have negative elements, but the spectral constraint $M\mathbf{1} = \mathbf{1}$ still holds: consider the compact SVD: $\Theta = U\Sigma V'$ where $U'U = I$ and $V'V = I$. Then $\Theta'\mathbf{1} = k\mathbf{1}$ implies $\Sigma U'\mathbf{1} = kV'\mathbf{1}$. Since $M = VV'$: $M\mathbf{1} = VV'\mathbf{1} = \frac{1}{k}V\Sigma U'\mathbf{1} = \frac{1}{k}\Theta'\mathbf{1} = \mathbf{1}$ (Note $\Theta_{ij} \in \{0,1\}$ is not used). So we will choose the domains for M_1 and M_2 in (4.13) to consist of the spectral constraints:

$$\mathcal{M} := \{\mathbf{0} \preceq M \preceq I : M\mathbf{1} = \mathbf{1},\ \mathrm{tr}(M) = h\}. \tag{4.15}$$

4.3.2 A Jointly Convex Multi-label Loss for Normalized Kernels

An important challenge is to design an appropriate nonlinear loss to connect each layer of the model. Rather than conditional log-likelihood in a generative model, Chapter 3 introduced the idea of a using large margin, multi-label loss between a linear response, $\mathbf{z}$, and a boolean target vector, $\mathbf{y} \in \{0,1\}^h$:

$$\tilde{L}(\mathbf{z},\mathbf{y}) = \max(\mathbf{1} - \mathbf{y} + k\,\mathbf{z} - \mathbf{1}(\mathbf{y}'\mathbf{z})) \tag{4.16}$$

where $\mathbf{1}$ denotes the vector of all 1s. Intuitively this encourages the responses on the active labels, $\mathbf{y}'\mathbf{z}$, to exceed k times the response of any inactive label, kz_i, by a margin, where the implicit nonlinear transfer is a step function. Remarkably, this loss can be shown to satisfy Postulate 1 which follows from Chapter 3.

This loss can be easily adapted to the normalized case as follows. We first generalize the notion of margin to consider a a "normalized label" $(YY')^\dagger\mathbf{y}$:

$$L(\mathbf{z},\mathbf{y}) = \max(\mathbf{1} - (YY')^\dagger\mathbf{y} + k\,\mathbf{z} - \mathbf{1}(\mathbf{y}'\mathbf{z}))$$

To obtain some intuition, consider the multiclass case where $k = 1$. Then YY' is a diagonal matrix whose (i,i)-th element is the number of examples in each class i. Dividing by this number allows the margin requirement to be weakened for popular labels, while more focus is shifted to less represented labels. For a

given set of input/output pairs (Z, Y) the sum of the losses can then be compactly expressed as $L(Z, Y) = \sum_j L(\mathbf{z}_j, \mathbf{y}_j) = \tau(kZ - (YY')^\dagger Y) + t - \operatorname{tr}(Y'Z)$, where $\tau(\Gamma) := \sum_j \max_i \Gamma_{ij}$. This loss can be shown to satisfy Postulate 3.

Proposition 1.

$$L^n(S, M) = \tau(S - \tfrac{1}{k}M) + t - \operatorname{tr}(S), \tag{4.17}$$

where $S = Y'Z$ and $M = Y'(YY')^\dagger Y$.

Proof. A simple derivation extends the one given in Chapter 3. Observe that

$$\begin{aligned}
\tau(kZ - (YY')^\dagger Y) &= \max_{\Lambda:\mathbb{R}_+^{m\times t}:\Lambda'\mathbf{1}=\mathbf{1}} \operatorname{tr}(\Lambda'(kZ - (YY')^\dagger Y)) \\
&= \max_{\Omega:\mathbb{R}_+^{t\times t}:\Omega'\mathbf{1}=\mathbf{1}} \tfrac{1}{k}\operatorname{tr}(\Omega'Y'(kZ - (YY')^\dagger Y)) \\
&= \tau(Y'Z - \tfrac{1}{k}M).
\end{aligned}$$

Here the second equality follows because for any $\Lambda \in \mathbb{R}_+^{m\times t}$ satisfying $\Lambda'\mathbf{1} = \mathbf{1}$, there must be an $\Omega \in \mathbb{R}_+^{t\times t}$ satisfying $\Omega'\mathbf{1} = \mathbf{1}$ and $\Lambda = Y\Omega/k$. ∎

This loss can be naturally interpreted using the remark following Postulate 1. It encourages that the propensity of example j with respect to itself, S_{jj}, should be higher than its propensity with respect to other examples, S_{ij}, by a margin that is defined through the normalized kernel M. However note this loss does not correspond to a linear transfer between layers, even in terms of the propensity matrix S or normalized output kernel M. As in all large margin methods, the initial loss (4.16) is a convex upper bound for an underlying discrete loss defined with respect to a step transfer.

4.4 Efficient Optimization

Efficient optimization for the multi-layer model (4.13) is challenging, largely due to the matrix pseudo-inverse. Fortunately, the constraints on M are all spectral, which

Algorithm 3: Conditional gradient algorithm to optimize $f(M_1, M_2)$ for $M_1, M_2 \in \mathcal{M}$.

1 Initialize $\tilde{M}_1$ and $\tilde{M}_2$ with some random matrices.
2 **while** $s = 1, 2, \ldots$ **do**
3 Compute the gradients $G_1 = \frac{\partial}{\partial M_1} f(\tilde{M}_1, \tilde{M}_2)$ and $G_2 = \frac{\partial}{\partial M_2} f(\tilde{M}_1, \tilde{M}_2)$.
4 Compute the new bases M_1^s and M_2^s by invoking oracle (4.19) with G_1 and G_2 respectively.
5 Totally corrective update: $\min_{\boldsymbol{\alpha} \in \Delta_s, \boldsymbol{\beta} \in \Delta_s} f\left(\sum_{i=1}^s \alpha_i M_1^i, \sum_{i=1}^s \beta_i M_2^i\right)$.
6 Set $\tilde{M}_1 = \sum_{i=1}^s \alpha_i M_1^i$ and $\tilde{M}_2 = \sum_{i=1}^s \beta_i M_2^i$; **break** if stopping criterion is met.
7 **return** $(\tilde{M}_1, \tilde{M}_2)$.

makes it easier to apply conditional gradient (CG) methods (Jaggi, 2013). This is much more convenient than the models based on unnormalized kernels in Chapter 3, where the presence of both spectral and non-spectral constraints necessitated expensive algorithms such as alternating direction method of multipliers (Boyd et al., 2010).

Denote the objective in (4.13) as $g(M_1, M_2, S_1, S_2, Z_3)$. The idea is to optimize

$$f(M_1, M_2) := \min_{S_1 \in M_1 \mathbb{R} K, S_2 \in M_2 \mathbb{R} M_1, Z_3 \in \mathbb{R} M_2} g(M_1, M_2, S_1, S_2, Z_3) \tag{4.18}$$

by CG. The algorithm is described in Algorithm 3.

We next demonstrate how each step can be executed efficiently.

Oracle problem in Step 4. This requires solving, given a real symmetric gradient G:

$$\max_{M \in \mathcal{M}} \operatorname{tr}(-GM) \Leftrightarrow \max_{\mathbf{0} \preceq M_1 \preceq I,\ \operatorname{tr}(M_1) = h-1} \operatorname{tr}(-G(HM_1H + \tfrac{1}{t}\mathbf{1}\mathbf{1}')), \tag{4.19}$$

where $H = I - \frac{1}{t}\mathbf{1}\mathbf{1}'$. Here we used Lemma 1 of (Cheng et al., 2013). By (Overton & Womersley, 1993, Theorem 3.4), $\max_{\mathbf{0} \preceq M_1 \preceq I,\ \operatorname{tr}(M_1) = h-1} \operatorname{tr}(-HGHM_1) =$

$\sum_{i=1}^{h-1} \lambda_i$ where $\lambda_1 \geq \lambda_2 \geq \ldots$ are the leading eigenvalues of $-HGH$. The maximum is attained at $M_1 = \sum_{i=1}^{h-1} \mathbf{v}_i \mathbf{v}_i'$, where $\mathbf{v}_i$ is the eigenvector corresponding to λ_i. The optimal solution to $\operatorname{argmax}_{M \in \mathcal{M}} \operatorname{tr}(-GM)$ can be recovered by $\sum_{i=1}^{h-1} \mathbf{v}_i \mathbf{v}_i' + \frac{1}{t} \mathbf{1}\mathbf{1}'$, which has low rank for small h.

Totally corrective update in Step 5. This is the most computationally intensive step:

$$\min_{\boldsymbol{\alpha} \in \Delta_s,\, \boldsymbol{\beta} \in \Delta_s} f\left(\sum\nolimits_{i=1}^{s} \alpha_i M_1^i, \sum\nolimits_{i=1}^{s} \beta_i M_2^i\right), \tag{4.20}$$

where Δ_s stands for the s dimensional probability simplex (sum up to 1). If one can solve (4.20) efficiently (which also provides the optimal S_1, S_2, Z_3 in (4.18) for the optimal $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$), then the gradient of f can also be obtained easily by Danskin's theorem (for Step 3 of Algorithm 3). However, the totally corrective update is expensive because given $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, each evaluation of the objective f itself requires an optimization over S_1, S_2, and Z_3. Such a nested optimization can be prohibitive.

A key idea is to show that this totally corrective update can be accomplished with considerably improved efficiency through the use of block coordinate descent (Dinuzzo et al., 2011). Taking into account the structure of the solution to the oracle, we denote

$$M_1(\boldsymbol{\alpha}) := \sum_i \alpha_i M_1^i = V_1 D(\boldsymbol{\alpha}) V_1', \text{ and } M_2(\boldsymbol{\beta}) := \sum_i \beta_i M_2^i = V_2 D(\boldsymbol{\beta}) V_2', \tag{4.21}$$

where $D(\boldsymbol{\alpha}) = \operatorname{diag}([\alpha_1 \mathbf{1}_h', \alpha_2 \mathbf{1}_h', \ldots]')$ and $D(\boldsymbol{\beta}) = \operatorname{diag}([\beta_1 \mathbf{1}_h', \beta_2 \mathbf{1}_h', \ldots]')$. Denote

$$P(\boldsymbol{\alpha}, \boldsymbol{\beta}, S_1, S_2, Z_3) := g\left(M_1(\boldsymbol{\alpha}), M_2(\boldsymbol{\beta}), S_1, S_2, Z_3\right). \tag{4.22}$$

Clearly $S_1 \in M_1(\boldsymbol{\alpha})\mathbb{R}K$ iff $S_1 = V_1 A_1 K$ for some A_1, $S_2 \in M_2(\boldsymbol{\beta})\mathbb{R}M_1(\boldsymbol{\alpha})$ iff $S_2 = V_2 A_2 M_1(\boldsymbol{\alpha})$ for some A_2, and $Z_3 \in \mathbb{R}M_2(\boldsymbol{\beta})$ iff $Z_3 = A_3 M_2(\boldsymbol{\beta})$ for some

A_3. So (4.20) is equivalent to

$$\min_{\boldsymbol{\alpha}\in\Delta_s,\ \boldsymbol{\beta}\in\Delta_s, A_1, A_2, A_3} P\left(\boldsymbol{\alpha}, \boldsymbol{\beta}, V_1 A_1 K, V_2 A_2 M_1(\boldsymbol{\alpha}), A_3 M_2(\boldsymbol{\beta})\right) \tag{4.23}$$

$$= L_1^n(V_1 A_1 K, M_1(\boldsymbol{\alpha})) + \tfrac{1}{2}\operatorname{tr}(V_1 A_1 K A_1' V_1') \tag{4.24}$$

$$+ L_2^n(V_2 A_2 M_1(\boldsymbol{\alpha}), M_2(\boldsymbol{\beta})) + \tfrac{1}{2}\operatorname{tr}(V_2 A_2 M_1(\boldsymbol{\alpha}) A_2' V_2') \tag{4.25}$$

$$+ L_3(A_3 M_2(\boldsymbol{\beta}), Y) + \tfrac{1}{2}\operatorname{tr}(A_3 M_2(\boldsymbol{\beta}) A_3'). \tag{4.26}$$

Thus we have eliminated all matrix pseudo-inverses. However, it is still expensive because the size of A_i depends on t. To simplify further, assume X', V_1 and V_2 all have full column rank. (This assumption is valid provided the features in X are linearly independent, since the bases (eigen-vectors) accumulated through all iterations so far are also independent. The only exception is the eigen-vector $\frac{1}{\sqrt{t}}\mathbf{1}$. But since $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ lie on a simplex, it always contributes a *constant* $\frac{1}{t}\mathbf{1}\mathbf{1}'$ to $M_1(\boldsymbol{\alpha})$ and $M_2(\boldsymbol{\beta})$.) Denote $B_1 = A_1 X'$ (note $K = X'X$), $B_2 = A_2 V_1$, $B_3 = A_3 V_2$. Noting (4.21), the objective becomes

$$R(\boldsymbol{\alpha}, \boldsymbol{\beta}, B_1, B_2, B_3) := L_1^n(V_1 B_1 X, V_1 D(\boldsymbol{\alpha}) V_1') + \tfrac{1}{2}\operatorname{tr}(V_1 B_1 B_1' V_1') \tag{4.27}$$

$$+ L_2^n(V_2 B_2 D(\boldsymbol{\alpha}) V_1', V_2 D(\boldsymbol{\beta}) V_2') + \tfrac{1}{2}\operatorname{tr}(V_2 B_2 D(\boldsymbol{\alpha}) B_2' V_2') \tag{4.28}$$

$$+ L_3(B_3 D(\boldsymbol{\beta}) V_2', Y) + \tfrac{1}{2}\operatorname{tr}(B_3 D(\boldsymbol{\beta}) B_3'). \tag{4.29}$$

This problem is much easier to solve, since the size of B_i depends on the number of input features, the number of nodes in two latent layers, and the number of output labels. Due to the greedy nature of CG, the number of latent nodes is generally low. So we can optimize R by block coordinate descent (BCD), *i.e.* alternating between:

1. Fix $(\boldsymbol{\alpha}, \boldsymbol{\beta})$, and solve (B_1, B_2, B_3) (unconstrained smooth optimization, *e.g.* by LBFGS).

2. Fix (B_1, B_2, B_3), and solve $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ (*e.g.* by LBFGS with projection to simplex).

BCD is guaranteed to converge to a critical point when L_1^n, L_2^n and L_3 are smooth. (Technically, for BCD to converge to a critical point, each block optimization needs to have a unique optimal solution. To ensure uniqueness, we used a method equivalent to the proximal method in Proposition 7 of (Grippoa & Sciandrone, 2000).) In practice, these losses can be made smooth by, *e.g.* approximating the max in (4.17) by a softmax. It is crucial to note that although both of the two steps are convex, R is *not* jointly convex in its variables. So in general, this alternating scheme can only produce a *stationary point* of R. Interestingly, we further show that *any* stationary point must provide a *global* optimal solution to P in (4.22).

Theorem 2. Suppose $(\boldsymbol{\alpha}, \boldsymbol{\beta}, B_1, B_2, B_3)$ is a stationary point of R with $\alpha_i > 0$ and $\beta_i > 0$. Assume X', V_1 and V_2 all have full column rank. Then it must be a *globally* optimal solution to R, and this $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ must be an optimal solution to the totally corrective update (4.20).

To enforce Theorem 2, we will use constraints

$$\alpha_i \geq 1/s, \quad \beta_i \geq 1/s, \quad \mathbf{1}'\boldsymbol{\alpha} = 1, \quad \mathbf{1}'\boldsymbol{\beta} = 1. \tag{4.30}$$

Proof. Let

$$C_2 = B_2 D(\boldsymbol{\alpha}), \qquad C_3 = B_3 D(\boldsymbol{\beta}) \tag{4.31}$$

Then R is equivalent to

$$\begin{aligned} R(\boldsymbol{\alpha}, \boldsymbol{\beta}, B_1, B_2, B_3) &= S(\boldsymbol{\alpha}, \boldsymbol{\beta}, B_1, C_2, C_3) && (4.32)\\ &:= L_1^n(V_1 B_1 X, V_1 D(\boldsymbol{\alpha}) V_1') + \frac{1}{2}\operatorname{tr}(V_1 B_1 B_1' V_1') && (4.33)\\ &+ L_2^n(V_2 C_2 V_1', V_2 D(\boldsymbol{\beta}) V_2') + \frac{1}{2}\operatorname{tr}(V_2 C_2 D(\boldsymbol{\alpha})^\dagger C_2' V_2) && (4.34)\\ &+ L_3(C_3 V_2', Y) + \frac{1}{2}\operatorname{tr}(C_3 D(\boldsymbol{\beta})^\dagger C_3'). && (4.35) \end{aligned}$$

Clearly S is jointly convex in $(\boldsymbol{\alpha}, \boldsymbol{\beta}, B_1, C_2, C_3)$. Thanks to the invertability of $D(\boldsymbol{\alpha})$ and $D(\boldsymbol{\beta})$, we apply chain rule to the stationarity condition of B_i:

$$\begin{aligned}
\mathbf{0} = \frac{\partial}{\partial B_1} R(\boldsymbol{\alpha}, \boldsymbol{\beta}, B_1, B_2, B_3) = \frac{\partial}{\partial B_1} S(\boldsymbol{\alpha}, \boldsymbol{\beta}, B_1, C_2, C_3) \\
\Rightarrow \frac{\partial}{\partial B_1} S(\boldsymbol{\alpha}, \boldsymbol{\beta}, B_1, C_2, C_3) = \mathbf{0} \qquad (4.36) \\
\mathbf{0} = \frac{\partial}{\partial B_2} R(\boldsymbol{\alpha}, \boldsymbol{\beta}, B_1, B_2, B_3) = \frac{\partial}{\partial C_2} S(\boldsymbol{\alpha}, \boldsymbol{\beta}, B_1, C_2, C_3) D(\boldsymbol{\alpha} \\
\Rightarrow \frac{\partial}{\partial C_2} S(\boldsymbol{\alpha}, \boldsymbol{\beta}, B_1, C_2, C_3) = \mathbf{0} \qquad (4.37) \\
\mathbf{0} = \frac{\partial}{\partial B_3} R(\boldsymbol{\alpha}, \boldsymbol{\beta}, B_1, B_2, B_3) = \frac{\partial}{\partial C_3} S(\boldsymbol{\alpha}, \boldsymbol{\beta}, B_1, C_2, C_3) D(\boldsymbol{\beta}) \\
\Rightarrow \frac{\partial}{\partial C_3} S(\boldsymbol{\alpha}, \boldsymbol{\beta}, B_1, C_2, C_3) = \mathbf{0} \qquad (4.38)
\end{aligned}$$

Note R is convex in $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ given B_i, and α_i, β_i satisfy the constraint (4.30). So KKT conditions (regarding the optimality of $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ given B_1, B_2, B_3) ensure that there exist Lagrange multipliers (μ, λ) (corresponding to the sum up to one constraints) and (a_i, b_i) (corresponding to the greater than or equal to $1/s$ constraint) such that

$$\alpha_i \geq \frac{1}{s}, \quad \beta_i \geq \frac{1}{s}, \quad \mathbf{1}'\boldsymbol{\alpha} = 1, \quad \mathbf{1}'\boldsymbol{\beta} = 1, \qquad (4.39)$$

$$a_i(\alpha_i - \tfrac{1}{s}) = 0, \quad b_i(\beta_i - \tfrac{1}{s}) = 0, \quad a_i \geq 0, \quad b_i \geq 0, \qquad (4.40)$$

and

$$\begin{aligned}
0 &= \frac{\partial}{\partial \alpha_i} R(\boldsymbol{\alpha}, \boldsymbol{\beta}, B_1, B_2, B_3) - \mu - a_i && (4.41) \\
\text{(chain rule)} \ &= \frac{\partial}{\partial \alpha_i} S(\boldsymbol{\alpha}, \boldsymbol{\beta}, B_1, C_2, C_3) + \left\langle \frac{\partial}{\partial \alpha_i} B_2 D(\boldsymbol{\alpha}), \frac{\partial}{\partial C_2} S(\boldsymbol{\alpha}, \boldsymbol{\beta}, B_1, C_2, C_3) \right\rangle \\
&\quad - \mu - a_i && (4.42) \\
\text{by (4.37)} \ &= \frac{\partial}{\partial \alpha_i} S(\boldsymbol{\alpha}, \boldsymbol{\beta}, B_1, C_2, C_3) - \mu - a_i, && (4.43)
\end{aligned}$$

and similarly,

$$0 = \frac{\partial}{\partial \beta_i} S(\boldsymbol{\alpha}, \boldsymbol{\beta}, B_1, C_2, C_3) - \lambda - b_i. \tag{4.44}$$

Since S is jointly convex in all its variables, (4.36)-(4.40), (4.43), and (4.44) provide all the KKT conditions required to establish the global optimality of $(\boldsymbol{\alpha}, \boldsymbol{\beta}, B_1, C_2, C_3)$ for S. The claims in the theorem then follow. ∎

It is noteworthy that the conditions $\alpha_i > 0$ and $\beta_i > 0$ are trivial to meet because CG is guaranteed to converge to optimal if $\alpha_i \geq 1/s$ and $\beta_i \geq 1/s$ at each step s.

4.5 Empirical Investigation

To investigate the potential of deep versus shallow convex training methods, and global versus local training methods, we implemented the approach outlined above for a three-layer model along with comparison methods. Below we use CVX3 and CVX2 to refer respectively to three and two-layer versions of theTSJ1 proposed model. For comparison, SVM1 refers to a one-layer SVM; and TSS1 (Sindhwani & Keerthi, 2006) and TSJ1 (Joachims, 1999) refer to one-layer transductive SVMs; NET2 refers to a standard two-layer sigmoid neural network with hidden layer size chosen by cross-validation; and LOC3 refers to the proposed three-layer model with exact (unrelaxed) with local optimization. In these evaluations, we followed a similar transductive set up to that of Chapter 3: a given set of data (X, Y) is divided into separate training and test sets, (X_L, Y_L) and X_U, where labels are only included for the training set. The training loss is then only computed on the training data, but the learned kernel matrices span the union of data. For testing, the kernel responses on test data are used to predict output labels.

4.6 Synthetic Experiments

Our first goal was to compare the effective modeling capacity of a three versus two-layer architecture given the convex formulations developed above. In particular, since the training formulation involves a convex relaxation of the normalized kernel domain, $\mathcal{M}$ in (4.15), it is important to determine whether the representational advantages of a three versus two-layer architecture are maintained. We conducted two sets of experiments designed to separate one-layer from two-layer or deeper models, and two-layer from three-layer or deeper models. Although separating two from one-layer models is straightforward, separating three from two-layer models is a subtler question. Here we considered two synthetic settings defined by basic functions over boolean features:

$$\text{Parity:} \quad y = x_1 \oplus x_2 \oplus \ldots \oplus x_n, \tag{4.45}$$

$$\text{Inner Product:} \quad y = (x_1 \wedge x_{m+1}) \oplus (x_2 \wedge x_{m+2}) \oplus \ldots \oplus (x_m \wedge x_n) \tag{4.46}$$

where $m = \frac{n}{2}$. It is well known that Parity is easily computable by a two-layer linear-gate architecture but cannot be approximated by any one-layer linear-gate architecture on the same feature space (Hajnal, 1993). The Inner Product (IP) problem is motivated by a fundamental result in the circuit complexity literature: any small weights threshold circuit of depth 2 requires size $\exp(\Omega(n))$ to compute (4.46) (Hajnal, 1993; Razborov, 1992). To generate data from these models, we set the number of input features to $n = 8$ (instead of $n = 2$ as in Chapter 3), then generate 200 examples for training and 100 examples for testing; for each example, the features x_i were drawn from $\{0, 1\}$ with equal probability. Then each x_i was corrupted independently by a Gaussian noise with zero mean and variance 0.3. The experiments were repeated 100 times, and the resulting test errors of the two models are plotted in Figure 4.2 and Figure 4.3 (larger dots mean repetitions fall on the same point). Figure 4.3 clearly shows that CVX3 is able to capture the structure of the Inner Product problem much more effectively than CVX2, as the theory sug-

gests for such architectures. In almost every repetition, CVX3 yields a lower (often much lower) test error than CVX2. Even on the Parity problem (Figure 4.2), CVX3 generally produces lower error, although its advantage is not as significant. This is also consistent with theoretical analysis (Hajnal, 1993; Razborov, 1992), which shows that IP is harder to model than parity.

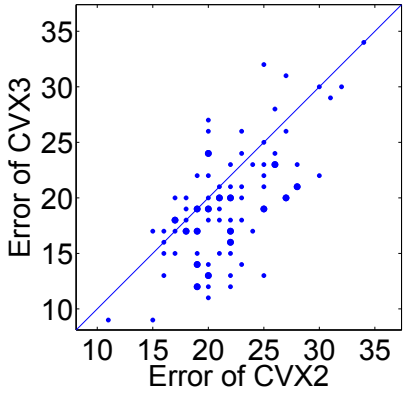


Figure 4.2: Parity dataset results

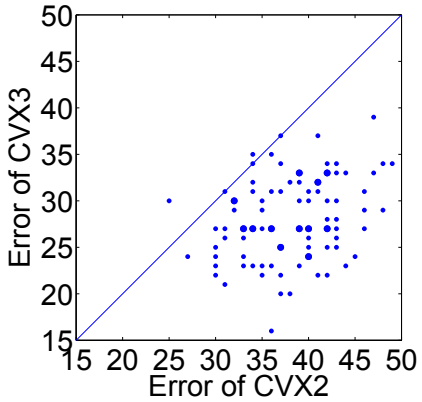


Figure 4.3: Inner Product dataset results

4.7 Experiments on Real Data

We also conducted an empirical investigation on some real data sets. Here we tried to replicate the results of Chapter 3 on similar data sets, USPS and COIL from (Chapelle et al., 2006), Letter from (Lichman, 2013), MNIST from (LeCun & Cortes, 2010), and CIFAR-100 from (Krizhevsky, 2009). Similar to (Joulin & Bach, 2012), we performed an optimistic model selection for each method on an initial sample of t training and t test examples; then with the parameters fixed the experiments were repeated 5 times on independently drawn sets of t training and t test examples from the remaining data.

	CIFAR	MNIST	USPS	COIL	Letter
TSS1	30.7 $\pm$4.2	16.3 $\pm$1.5	12.7 $\pm$1.2	16.0 $\pm$2.0	5.7 $\pm$2.0
TSJ1	26.0 $\pm$6.5	16.0 $\pm$2.0	11.0 $\pm$1.7	20.0 $\pm$3.6	5.0 $\pm$1.0
SVM1	33.3 $\pm$1.9	18.3 $\pm$0.5	12.7 $\pm$0.2	16.3 $\pm$0.7	7.0 $\pm$0.3
NET2	30.7 $\pm$1.7	15.3 $\pm$1.7	12.7 $\pm$0.4	15.3 $\pm$1.4	5.3 $\pm$0.5
CVX2	27.7 $\pm$5.5	12.7 $\pm$3.2	9.7 $\pm$3.1	14.0 $\pm$3.6	5.7 $\pm$2.9
LOC3	36.0 $\pm$1.7	22.0 $\pm$1.7	12.3 $\pm$1.1	17.7 $\pm$2.2	11.3 $\pm$0.2
CVX3	23.3 $\pm$0.5	13.0 $\pm$0.3	9.0 $\pm$0.9	9.0 $\pm$0.3	5.7 $\pm$0.2

Table 4.1: Mean test misclassification error % ((± stdev) 100/100 labeled/unlabeled.

The results shown in tables in Table 4.1 and Table 4.1 show that CVX3 is able to systematically reduce the test error of CVX2. This suggests that the advantage of deeper modeling does indeed arise from enhanced representation ability, and not merely from an enhanced ability to escape local minima or walk plateaus, since neither exist in these cases.

	CIFAR	MNIST	USPS	COIL	Letter
TSS1	32.0 $\pm$2.6	10.7 $\pm$3.1	10.3 $\pm$0.6	13.7 $\pm$4.0	3.8 $\pm$0.3
TSJ1	26.0 $\pm$3.3	10.0 $\pm$3.5	11.0 $\pm$1.3	18.9 $\pm$2.6	4.0 $\pm$0.5
SVM1	32.3 $\pm$1.6	12.3 $\pm$1.4	10.3 $\pm$0.1	14.7 $\pm$1.3	4.8 $\pm$0.5
NET2	30.7 $\pm$0.5	11.3 $\pm$1.3	11.2 $\pm$0.5	14.5 $\pm$0.6	4.3 $\pm$0.1
CVX2	23.3 $\pm$3.5	8.2 $\pm$0.6	7.0 $\pm$1.3	8.7 $\pm$3.3	4.5 $\pm$0.9
LOC3	28.2 $\pm$2.3	12.7 $\pm$0.6	8.0 $\pm$0.1	12.3 $\pm$0.9	7.3 $\pm$1.1
CVX3	19.2 $\pm$0.9	6.8 $\pm$0.4	6.2 $\pm$0.7	7.7 $\pm$1.1	3.0 $\pm$0.2

Table 4.2: Mean test misclassification error % (($\pm$ stdev) 200/200 labeled/unlabeled.

4.8 Conclusion

We have presented a new formulation of multi-layer training that can accommodate an arbitrary number of nonlinear layers while maintaining a jointly convex training objective. We proposed a new output kernel to be able to overcome the bottleneck that prevented the extension of the two-layer model of Chapter 3 to an arbitrary number of layers. We also developed a novel algorithm by exploiting the structure of the new formulation. We had to optimize the parameters for calculating the the function values in the previous two-layer formulation in Chapter 3, which proved to be an expensive step that required accurate parameter optimization to prevent instability. Instead, the current formulation enabled us to use alternation with global optimum guarantees, which eliminated the nested optimization of parameters. Accurate learning of additional layers, when required, appears to demonstrate a marked advantage over shallower architectures, even when models can be trained to optimality.

Chapter 5

Tractable Robust Modeling

We presented convex approximations of predictive latent models in Chapters 3 and 4. However real world datasets are often contaminated with many types of outliers, hence another critical aspect of ML in practice is robustness. Despite the variety of robust regression methods that have been developed, current regression formulations are either NP-hard, or allow unbounded response to even a single leverage point. In this chapter, we present a general formulation for robust regression—*Variational M-estimation*—that unifies a number of robust regression methods while allowing a tractable approximation strategy. We develop an estimator that requires only polynomial-time, while achieving certain robustness and consistency guarantees. The work in this chapter was published in Neural Information Processing Systems conference (Yu et al., 2012).

5.1 Introduction

As stated in background chapter, tractability and robustness have both only been achieved under restricted conditions, such as a bounded domain (Huber & Ronchetti, 2009; Christmann & Steinwart, 2007; Christmann et al., 2009). Our main motivation is to extend these existing results to the case of an unbounded domain. Unfor-

tunately, the inapplicability of convex losses in this situation means that computational tractability becomes a major challenge, and new computational strategies are required to achieve tractable robust estimators.

The main contribution of this chapter is to develop a new robust regression strategy that can guarantee both polynomial run-time and bounded response to individual outliers, including leverage points. Although such an achievement is modest, it is based on two developments of interest. The first is a general formulation of adaptive M-estimation, Variational M-estimation that unifies a number of robust regression formulations, including convex and bounded M-estimators with certain subset-selection estimators such as Least Trimmed Loss (Rousseeuw & Leroy, 1987a). By incorporating Tikhonov regularization, these estimators can be extended to reproducing kernel Hilbert spaces (RKHSs). The second development is a convex relaxation scheme that ensures bounded outlier influence on the final estimator. The overall estimation procedure is guaranteed to be tractable, robust to single outliers with unbounded leverage, and consistent under non-trivial conditions. An experimental evaluation of the proposed estimator demonstrates effective performance compared to standard robust estimators.

After presenting the related background material of robust regression in Section 5.2, we introduce the general form of adaptive M-estimation in Section 5.3. We then add Tikhonov regularizer and develop a convex relaxation that is lower bound on the original objective in Section 5.4. We show that the relaxation is tight and therefore maintains the robustness properties of the original formulation in Section 5.5. We empirically demonstrate the effectiveness of final relaxation in Section 5.6.

5.2 Background

We start by considering the standard linear regression model

$$y \;=\; \mathbf{x}'\boldsymbol{\theta}^* + u \tag{5.1}$$

where $\mathbf{x}$ is an $\mathbb{R}^p$-valued random variable, u is a real-valued random noise term, and $\boldsymbol{\theta}^* \in \Theta \subseteq \mathbb{R}^p$ is an unknown deterministic parameter vector. Assume we are given a sample of n independent identically distributed (i.i.d.) observations represented by a matrix $X \in \mathbb{R}^{n \times p}$ and a vector $\mathbf{y} \in \mathbb{R}^{n \times 1}$, where each row $X_{i:}$ is drawn from some unknown marginal probability measure $P_{\mathbf{x}}$, and y_i are generated according to (5.1). Our task is to estimate the unknown deterministic parameter $\boldsymbol{\theta}^* \in \Theta$. This is a well-studied problem in statistics and ML. If the noise distribution has a known density $p(\cdot)$, then a standard estimator is given by maximum likelihood, which is a well-known setting in statistics .

$$\hat{\boldsymbol{\theta}}_{ML} \in \arg\min_{\boldsymbol{\theta} \in \Theta} \tfrac{1}{n} \textstyle\sum_{i=1}^{n} -\log p(y_i - X_{i:}\boldsymbol{\theta}) = \arg\min_{\boldsymbol{\theta} \in \Theta} \tfrac{1}{n} \textstyle\sum_{i=1}^{n} -\log p(r_i), \tag{5.2}$$

where $r_i = y_i - X_{i:}\boldsymbol{\theta}$ is the ith residual. When the noise distribution is unknown, one can replace the negative log-likelihood with a *loss function* $\rho(\cdot)$ and use the estimator

$$\hat{\boldsymbol{\theta}}_{M} \in \arg\min_{\boldsymbol{\theta} \in \Theta} \tfrac{1}{n} \mathbf{1}'\boldsymbol{\rho}(\mathbf{y} - X\boldsymbol{\theta}), \tag{5.3}$$

where $\boldsymbol{\rho}(\mathbf{r})$ denotes the vector of losses obtained by applying the loss component-wise to each residual, hence $\mathbf{1}'\boldsymbol{\rho}(\mathbf{r}) = \sum_{i=1}^{n} \rho(r_i)$. Such a procedure is known as M-estimation in the robust statistics literature, and empirical risk minimization in the ML literature. (Generally one has to introduce an additional scale parameter σ and allow rescaling of the residuals, r_i/σ, to preserve parameter equivariance (Huber & Ronchetti, 2009; Maronna et al., 2006). However, we will initially assume a known scale σ.)

Although uncommon in robust regression, it is conventional in machine learning to include a regularizer. In particular we will use Tikhonov regularization by adding a squared penalty

$$\hat{\boldsymbol{\theta}}_{MR} \in \arg\min_{\boldsymbol{\theta} \in \Theta} \tfrac{1}{n} \mathbf{1}'\boldsymbol{\rho}(\mathbf{y} - X\boldsymbol{\theta}) + \tfrac{\lambda}{2}\|\boldsymbol{\theta}\|_2^2 \qquad \text{for } \lambda \geq 0. \tag{5.4}$$

The significance of Tikhonov regularization is that it ensures $\hat{\boldsymbol{\theta}}_{MR} = X'\boldsymbol{\alpha}$ for some $\boldsymbol{\alpha} \in \mathbb{R}^n$ (Kimeldorf & Wahba, 1970). More generally, under Tikhonov regularization, the regression problem can be conveniently expressed in a reproducing kernel Hilbert space (RKHS). If we let $\mathcal{H}$ denote the RKHS corresponding to positive semidefinite kernel $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, then $f(x) = \langle \kappa(x, \cdot), f \rangle_{\mathcal{H}}$ for any $f \in \mathcal{H}$ by the reproducing property (Kimeldorf & Wahba, 1970; Steinwart & Christmann, 2008). We consider the generalized regression model

$$y \quad = \quad f^*(x) + u \tag{5.5}$$

where x is an $\mathcal{X}$-valued random variable, u is a real-valued random noise term as above, and $f^* \in \mathcal{H}$ is an unknown deterministic function. Given a sample of n i.i.d. observations $(x_1, y_1), ..., (x_n, y_n)$, where each x_i is drawn from some unknown marginal probability measure P_x, and y_i are generated according to (5.5), the task is then to estimate the unknown deterministic function $f^* \in \mathcal{H}$. We are obviously assuming $\mathcal{X}$ is equipped with an appropriate σ-algebra, and $\mathbb{R}$ with the standard Borel σ-algebra, such that the joint distribution P over $\mathcal{X} \times \mathbb{R}$ is well defined and $\kappa(\cdot, \cdot)$ is measurable. To do so we can express the estimator (5.4) more generally as

$$\hat{f}_{MR} \quad \in \quad \arg\min_{f \in \mathcal{H}} \tfrac{1}{n} \textstyle\sum_{i=1}^n \rho(y_i - f(x_i)) + \tfrac{\lambda}{2} \|f\|_{\mathcal{H}}^2. \tag{5.6}$$

By the representer theorem (Kimeldorf & Wahba, 1970), the solution to (5.6) can be expressed by $\hat{f}_{MR}(x) = \sum_{i=1}^n \alpha_i \kappa(x_i, x)$ for some $\boldsymbol{\alpha} \in \mathbb{R}^n$. and therefore (5.6) can be recovered by solving the finite dimensional problem

$$\hat{\boldsymbol{\alpha}}_{MR} \quad \in \quad \arg\min_{\boldsymbol{\alpha}} \tfrac{1}{n} \mathbf{1}^T \boldsymbol{\rho}(\mathbf{y} - K\boldsymbol{\alpha}) + \tfrac{\lambda}{2} \boldsymbol{\alpha}^T K \boldsymbol{\alpha} \quad \text{ such that } K_{ij} = \kappa(x_i, x_j). \tag{5.7}$$

Our interest is understanding the tractability, robustness and consistency aspects of such estimators.

5.2.1 Consistency

Much is known about the consistency properties of estimators expressed as regularized empirical risk minimizers. For example, the ML-estimator and the M-estimator are both known to be parameter consistent under general conditions (van der Vaart & Wellner, 1996).

In particular, let $M_n(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}\rho(y_i - \mathbf{x}_i^T\boldsymbol{\theta})$, let $M(\boldsymbol{\theta}) = \mathbb{E}(\rho(y_1 - \mathbf{x}_1^T\boldsymbol{\theta}))$, and equip the parameter space Θ with the uniform metric $\|\cdot\|_\Theta$. Then $\hat{\boldsymbol{\theta}}_M^{(n)} \to \boldsymbol{\theta}^*$, provided $\|M_n - M\|_\Theta \to 0$ in outer probability (adopted to avoid measurability issues) and $M(\boldsymbol{\theta}^*) > \sup_{\boldsymbol{\theta}\in G} M(\boldsymbol{\theta})$ for every open set G that contains $\boldsymbol{\theta}^*$. The latter assumption is satisfied in particular when $M : \Theta \mapsto \mathbb{R}$ is upper semicontinuous with a unique maximum at $\boldsymbol{\theta}^*$. It is also possible to derive asymptotic convergence rates for general M-estimators (van der Vaart & Wellner, 1996).

The regularized M-estimator in RKHSs (5.6), is loss consistent under some general assumptions on the kernel, loss and training distribution.

Specifically, let $\rho^* = \inf_{f\in\mathcal{H}} E[\rho(y_1 - f(x_1))]$. Then Christmann et al. (2009) showed that $\frac{1}{n}\sum_{i=1}^{n}\rho(y_i - \hat{f}_{MR}(x_i)) \to \rho^*$ provided the regularization constant $\lambda_n \to 0$ and $\lambda_n^2 n \to \infty$, the loss ρ is convex and Lipschitz-continuous, and the RKHS $\mathcal{H}$ (induced by some bounded measurable kernel κ) is separable and dense in $L_1(\mathbb{P})$ (the space of $\mathbb{P}$-integrable functions) for all distributions $\mathbb{P}$ on $\mathcal{X}$. Also, $\mathcal{Y} \subset \mathbb{R}$ is required to be *closed* where $y \in \mathcal{Y}$.

Furthermore, a weak form of f-consistency has also established in (Christmann et al., 2009). For bounded kernel and bounded Lipschitz losses, one can similarly prove the loss consistency of the regularized M-estimator (5.6) (in RKHS).

Let us start by establishing some notation. Recall that $\mathcal{H}$ is the RKHS induced by some kernel κ, $\{(x_i, y_i)\}_{i=1}^n$ are *i.i.d.*samples from the underlying training distribution $\mathbb{P}(x, y)$. For any function f (possibly random), define

$$\mathcal{R}(f) := \int_{\mathcal{X}\times\mathbb{R}} \rho(y - f(x))\mathrm{d}\mathbb{P}(x, y).$$

Then the following functions all aim at minimizing $\mathcal{R}(f)$ in one way or another:

$$f^* \in \operatorname*{argmin}_f \mathcal{R}(f) \tag{5.8}$$

$$f_{\mathcal{H}} \in \operatorname*{argmin}_{f \in \mathcal{H}} \mathcal{R}(f) \tag{5.9}$$

$$f_{\mathcal{H},\lambda} \in \operatorname*{argmin}_{f \in \mathcal{H}} \mathcal{R}(f) + \lambda \|f\|_{\mathcal{H}}^2 \tag{5.10}$$

$$\hat{f}_{\mathcal{H},\lambda} \in \operatorname*{argmin}_{f \in \mathcal{H}} \hat{\mathcal{R}}(f) + \lambda \|f\|_{\mathcal{H}}^2, \tag{5.11}$$

where $\hat{\mathcal{R}}(f) := \frac{1}{n} \sum_{i=1}^n \rho(y_i - f(x_i))$. For simplicity, we have tacitly assumed the existence of all minimizers in the above. We will also ignore measurability issues for the time being. Note that only the last function $\hat{f}_{\mathcal{H},\lambda}$ depends on the data.

The regularized M-estimator $\hat{f}_{\mathcal{H},\lambda}$ is said to be (loss) consistent if

$$\mathcal{R}(\hat{f}_{\mathcal{H},\lambda}) - \mathcal{R}(f^*) \to 0 \tag{5.12}$$

as the sample size n increases to infinity and the regularization constant λ decreases to zero. To investigate when consistency can be assured, the following decomposition is standard and helpful (Steinwart & Christmann, 2008):

$$\begin{aligned} \mathcal{R}(\hat{f}_{\mathcal{H},\lambda}) - \mathcal{R}(f^*) = \quad & \mathcal{R}(\hat{f}_{\mathcal{H},\lambda}) - \mathcal{R}(f_{\mathcal{H},\lambda}) - \lambda \|f_{\mathcal{H},\lambda}\|_{\mathcal{H}}^2 && (5.13) \\ & + \mathcal{R}(f_{\mathcal{H},\lambda}) + \lambda \|f_{\mathcal{H},\lambda}\|_{\mathcal{H}}^2 - \mathcal{R}(f_{\mathcal{H}}) && (5.14) \\ & + \mathcal{R}(f_{\mathcal{H}}) - \mathcal{R}(f^*), && (5.15) \end{aligned}$$

where the last term is usually called the approximation error, which measures how well can functions in $\mathcal{H}$ approximate f^* under the ρ loss; the second term can be thought of as some stability error (where the regularization constant λ plays the role of perturbation); and the first term is related to the sampling error. Note that the last two terms depend only on the interaction between the RKHS $\mathcal{H}$ (hence the kernel κ), the training distribution $\mathbb{P}(x, y)$, and the loss ρ. It is however independent of any estimation procedure (except perhaps providing some insights on how to practically tune the regularization constant). Very general bounds for the last two terms exist

in the learning theory literature, see, for instance (Steinwart & Christmann, 2008) and (Cucker & Zhou, 2007). To make our presentation less complicated we will simply assume the sum of the last two terms, as a function of λ, goes to 0 when λ itself decreases to 0. The interested reader can consult the books (Steinwart & Christmann, 2008) and (Cucker & Zhou, 2007) for precise technical conditions under which this is indeed so. The right-hand side in (5.13) is apparently upper bounded by

$$\begin{aligned}\mathcal{R}(\hat{f}_{\mathcal{H},\lambda}) + \lambda\|\hat{f}_{\mathcal{H},\lambda}\|_{\mathcal{H}}^2 - \mathcal{R}(f_{\mathcal{H},\lambda}) - \lambda\|f_{\mathcal{H},\lambda}\|_{\mathcal{H}}^2 &= \mathcal{R}(\hat{f}_{\mathcal{H},\lambda}) - \hat{\mathcal{R}}(\hat{f}_{\mathcal{H},\lambda}) + \hat{\mathcal{R}}(\hat{f}_{\mathcal{H},\lambda}) \\ &\quad + \lambda\|\hat{f}_{\mathcal{H},\lambda}\|_{\mathcal{H}}^2 - \hat{\mathcal{R}}(f_{\mathcal{H},\lambda}) - \lambda\|f_{\mathcal{H},\lambda}\|_{\mathcal{H}}^2 \\ &\quad + \mathcal{R}(f_{\mathcal{H},\lambda}) - \hat{\mathcal{R}}(f_{\mathcal{H},\lambda}) \\ &\leq \mathcal{R}(\hat{f}_{\mathcal{H},\lambda}) - \hat{\mathcal{R}}(\hat{f}_{\mathcal{H},\lambda}) + \mathcal{R}(f_{\mathcal{H},\lambda}) \\ &\quad - \hat{\mathcal{R}}(f_{\mathcal{H},\lambda}) \\ &\leq \sup_{f:\|f\|_{\mathcal{H}} \leq \frac{1}{\sqrt{\lambda}}} |\mathcal{R}(f) - \hat{\mathcal{R}}(f)|,\end{aligned} \tag{5.16}$$

where the first inequality is due to the optimality of $\hat{f}_{\mathcal{H},\lambda}$, and the second inequality follows under the assumption $0 \leq \rho \leq 1$. Therefore we have related the right-hand side in (5.13) to the (uniform) sampling error. Applying standard uniform convergence bounds, for instance, the Rademacher complexity bound in (Bartlett & Mendelson, 2003, Theorem 8; Theorem 12; Lemma 22), leads to the following consistency result:

Proposition 2. Assuming (5.14) and (5.15) approach 0 when $\lambda \to 0$, the loss ρ is Lipschitz and bounded between 0 and 1, $\sup_{x\in\mathcal{X}} \kappa(x,x) \leq 1$, then the regularized M-estimator defined in (5.11) is (ρ-loss) consistent.

It is also possible to derive consistency results that depends on the capacity of the RKHS (Steinwart & Christmann, 2008). Generally speaking, any estimator that can be expressed as a regularized empirical loss minimization is consistent under

"reasonable" conditions. That is, one can consider regularized loss minimization to be a (generally) sound principle for formulating regression estimators, at least from the perspective of consistency. However, this is no longer the case when we consider robustness and tractability; here sharp distinctions begin to arise within this class of estimators.

5.2.2 Robustness

Although robustness is an intuitive notion, it has not been given a unique technical definition in the literature. Several definitions have been proposed, with distinct advantages and disadvantages (Huber & Ronchetti, 2009). Some standard definitions consider the asymptotic invariance of estimators to an infinitesimal but arbitrary perturbation of the underlying distribution, e.g. the influence function (Hampel et al., 1986; Huber & Ronchetti, 2009). Although these analyses can be useful, we will focus on finite sample notions of robustness since these are most related to concerns of computational tractability. In particular, we focus on the following definition related to the *finite sample breakdown point* (Donoho & Huber, 1983; Davies & Gather, 2007).

Definition 1 (Bounded Response)**.** Assuming the parameter set Θ is metrizable, an estimator has *bounded response* if for any finite data sample its output remains in a bounded interior subset of the closed parameter set Θ (or respectively $\mathcal{H}$), no matter how a *single* observation pair is perturbed.

This is a much weaker definition than having a non-zero breakdown point: a breakdown of ϵ requires that bounded response be guaranteed when any ϵ fraction of the data is perturbed arbitrarily. Bounded response is obviously a far more modest requirement. However, importantly, the definition of bounded response allows the possibility of arbitrary *leverage*; that is, no bound is imposed on the magnitude of a perturbed input (i.e. $\|\mathbf{x}_1\| \to \infty$ or $\kappa(x_1, x_1) \to \infty$). Surprisingly, we find

that even such a weak robustness property is difficult to achieve while retaining computational tractability.

5.2.3 Computational Dilemma

The goals of robustness and computational tractability raise a dilemma: it is easy to achieve robustness (i.e. bounded response) or tractability (i.e. polynomial run-time) in a consistent estimator, but apparently not both.

Consider, for example, using a *convex* loss function. These are the best known class of functions that admit computationally efficient polynomial-time minimization (Nesterov & Nimirovskii, 1994) (see also (Nesterov, 2003)). It is sufficient that the objective be polynomial-time evaluable, along with its first and second derivatives, and that the objective be *self-concordant* (Nesterov & Nimirovskii, 1994). (A function ρ is *self-concordant* if $|\rho'''(r)| \leq 2\rho''(r)^{3/2}$; see e.g. (Boyd & Vandenberghe, 2004, Ch.9).) Since a Tikhonov regularizer is automatically self-concordant, the minimization problems outlined above can all be solved in polynomial time with Newton-type algorithms, provided $\rho(r)$, $\rho'(r)$, and $\rho''(r)$ can all be evaluated in polynomial time for a self-concordant ρ (Boyd & Vandenberghe, 2004, Ch.9). Standard loss functions, such as squared error or Huber's loss satisfy these conditions, hence the corresponding estimators are polynomial-time. Unfortunately, loss minimization with a (non-constant) convex loss yields unbounded response to even a single outlier (Maronna et al., 2006, Ch.5).

We extend this result to also account for regularization and RKHSs.

Theorem 3. Empirical risk minimization based on a (non-constant) convex loss cannot have bounded response if the domain (or kernel) is unbounded, even under Tikhonov regularization.

We will separately prove the two cases; first, assuming an explicit feature representation expressed in a data matrix X, and second, in the case of an RKHS.

Explicit Feature Case

Proof. Consider the L_2-norm regularized M-estimator:

$$\hat{\boldsymbol{\theta}} \quad = \quad \arg\min_{\boldsymbol{\theta}\in\Theta} \frac{\lambda}{2}\|\boldsymbol{\theta}\|_2^2 + \sum_{i=1}^{n} \rho(y_i - X_{i:}\boldsymbol{\theta}), \tag{5.17}$$

where ρ is some (non-constant) convex function. For simplicity, we assume ρ is differentiable (otherwise one can consider subdifferentials to arrive at the same conclusion).

Suppose the theorem is false, then $\hat{\boldsymbol{\theta}}$ remains in a bounded interior subset. From the first order optimality condition (see, for instance, (Boyd & Vandenberghe, 2004)), we know that

$$[\rho'(y_1 - X_{1:}\hat{\boldsymbol{\theta}}) + \lambda]X_{1:} + \sum_{i=2}^{n}[\rho'(y_i - X_{i:}\hat{\boldsymbol{\theta}}) + \lambda]X_{i:} \quad = \quad 0. \tag{5.18}$$

Now we perturb $(X_{1:}, y_1)$ to cause a contradiction. Since ρ is a univariate convex and non-constant function, we apparently have $\lim_{d\to\infty}\rho'(d) > 0$ or $\lim_{d\to-\infty}\rho'(d) < 0$ or both. Assume $\lim_{d\to\infty}\rho'(d) > 0$ (the other case can be proved similarly).

Let y_1 and $\|X_{1:}\|$ tend to infinity in a way that $\frac{y_1}{\|X_{1:}\|}$ also tends to infinity. Then

$$y_1 - X_{1:}\hat{\boldsymbol{\theta}} \quad \geq \quad y_1 - \|X_{1:}\| \cdot \|\hat{\boldsymbol{\theta}}\| \tag{5.19}$$

$$\quad = \quad \|X_{1:}\| \cdot \left(\frac{y_1}{\|X_{1:}\|} - \|\hat{\boldsymbol{\theta}}\|\right) \tag{5.20}$$

tends to infinity since $\hat{\boldsymbol{\theta}}$ is bounded. Therefore $\rho'(y_1 - X_{1:}\hat{\boldsymbol{\theta}})$ tends to a positive number. But then the first term in (5.18) is unbounded in norm while the second term is bounded in norm (for we did not perturb $\{(X_{i:}, y_i)\}_{i=2}^n$), contradiction. ∎

Intuitively the meaning of this theorem is clear: If the loss function is unbounded, then any sample that is far enough can drag the estimate $\hat{\boldsymbol{\theta}}$ outside of any bounded interior subset. Tikhonov regularization is not able to overcome this effect. Note that we need to perturb both $X_{1:}$ and y_1, in order to derive a contradiction (for instance, if one only perturbs y_1, then convex functions with bounded derivatives will survive the proof).

RKHS Case

Proof. Let us consider the standard regularized M-estimator:

$$\hat{f}_{MR} \in \arg\min_{f\in\mathcal{H}} \frac{1}{n}\sum_{i=1}^{n} \rho(y_i - \langle\phi(x_i), f\rangle) + \frac{\lambda}{2}\|f\|_{\mathcal{H}}^2, \tag{5.21}$$

where $\{(x_i, y_i)\}_{i=1}^n$ are samples from the domain $\mathcal{X} \times \mathbb{R}$; and $\mathcal{H}$ is the RKHS induced by some *unbounded* kernel κ, with its canonical feature map $\phi : \mathcal{X} \mapsto \mathcal{H}$. As above, we assume ρ is some non-constant convex loss function. For simplicity, we assume that the minimum in (5.21) is attained.

Suppose the theorem is false, then $\hat{f}_{MR}$ remains in a bounded set. The first order necessary condition for the optimality of $\hat{f}_{MR}$ yields

$$\begin{aligned}\rho'\left(y_1 - \langle\phi(x_1), \hat{f}_{MR}\rangle\right)\phi(x_1) + \sum_{i=2}^{n}\rho'\left(y_i - \langle\phi(x_i), \hat{f}_{MR}\rangle\right)\phi(x_i) &+ n\lambda\hat{f}_{MR} \\ &= 0, \end{aligned}\tag{5.22}$$

where ρ' denotes the subdifferential of ρ (whose existence is guaranteed by convexity, moreover since ρ is finite-valued on $\mathbb{R}$, ρ' is also finite-valued on $\mathbb{R}$). The second term and third term above are bounded since $\hat{f}_{MR}$ is assumed to be bounded. We will perturb (x_1, y_1) such that the first term is not bounded in norm, hence creating a contradiction.

Since ρ is convex and non-constant, we must have either $\lim_{y\to\infty}\rho'(y) > 0$ or $\lim_{y\to-\infty}\rho'(y) > 0$ (or both). Let us assume $\lim_{y\to\infty}\rho'(y) > 0$ (the other case can be proved similarly).

Let both y_1 and $\kappa(x_1, x_1)$ tend to infinity in a way such that $\frac{y_1}{\sqrt{\kappa(x_1,x_1)}}$ also tends to infinity. Then

$$\begin{aligned} y_1 - \langle\phi(x_1), \hat{f}_{MR}\rangle &\geq y_1 - \sqrt{\kappa(x_1, x_1)}\|\hat{f}_{MR}\|_{\mathcal{H}} \\ &= \sqrt{\kappa(x_1, x_1)}\left(\frac{y_1}{\sqrt{\kappa(x_1, x_1)}} - \|\hat{f}_{MR}\|_{\mathcal{H}}\right), \end{aligned}\tag{5.23}$$

hence

$$\left\|\rho'\left(y_1 - \langle\phi(x_1), \hat{f}_{MR}\rangle\right)\phi(x_1)\right\|_{\mathcal{H}} = \left|\rho'\left(y_1 - \langle\phi(x_1), \hat{f}_{MR}\rangle\right)\right|\sqrt{\kappa(x_1, x_1)} \to \infty \quad (5.24)$$

due to our assumptions. ∎

Note that again we need to perturb both $\kappa(x_1, x_1)$ and y_1 to reach a contradiction (for instance, if one only perturbs y_1, then convex functions with bounded derivatives will survive our proof).

It should be clear in both these proofs that one may replace the L_2-norm with other regularizers without affecting Theorem 3.

By contrast, consider the case of a (non-constant) *bounded* loss function. (A bounded function obviously cannot be convex over an unbounded domain unless it is constant.) Bounded loss functions are a common choice in robust regression because they not only ensure bounded response, trivially, they can also ensure a high breakdown point of $(n-p)/(2n)$ (Maronna et al., 2006, Ch.5). Unfortunately, estimators based on bounded losses are inherently intractable.

Theorem 4. Bounded (non-constant) loss minimization is NP-hard. *(Proof given in Appendix A.3.4.)*

These difficulties with empirical risk minimization have led the field of robust statistics to develop a variety of alternative estimators (Huber & Ronchetti, 2009, Ch.7). For example, (Rousseeuw & Leroy, 1987a) recommends subset-selection based regression estimators, such as Least Trimmed Loss

$$\hat{\boldsymbol{\theta}}_{LTL} \quad \in \quad \arg\min_{\boldsymbol{\theta}\in\Theta} \sum_{i=1}^{n'} \rho(r_{[i]}). \quad (5.25)$$

Here $r_{[i]}$ denotes sorted residuals $r_{[1]} \leq \cdots \leq r_{[n]}$ and $n' < n$ is the number of terms to consider. Traditionally $\rho(r) = r^2$ is used. These estimators are known to have high breakdown (Rousseeuw & Leroy, 1987a), and obviously demonstrate

bounded response to single outliers. Unfortunately, it is NP-hard (Bernholt, 2005) (when n' approaches $n/2$ the breakdown of (5.25) approaches $1/2$ (Rousseeuw & Leroy, 1987a)).

5.3 Variational M-estimation

To address the dilemma, we first adopt a form of adaptive M-estimator that allows flexibility while allowing a general approximation strategy. The key idea is a variational representation of M-estimation that can express a number of standard robust (and non-robust) methods in a common framework. In particular, consider the following adaptive form of loss function

$$\rho(r) \quad = \quad \min_{0 \leq \eta \leq 1} \eta \ell(r) + \psi(\eta). \tag{5.26}$$

where r is a residual value, ℓ is a closed *convex* base loss, η is an adaptive weight on the base loss, and ψ is a convex auxiliary function. The weight can choose to ignore the base loss if $\ell(r)$ is large, but this is balanced against a prior penalty $\psi(\eta)$. Different choices of base loss and auxiliary function will yield different results, and one can represent a wide variety of loss functions ρ in this way (Black & Rangarajan, 1996). For example, any convex loss ρ can be trivially represented in the form (5.26) by setting $\ell = \rho$, and $\psi(\eta) = \delta_{\{1\}}(\eta)$. (We use $\delta_C(\eta)$ to denote the indicator for the point set C; i.e., $\delta_C(\eta) = 0$ if $\eta \in C$, otherwise $\delta_C(\eta) = \infty$.) *Bounded* loss functions that can also be represented in this way are given in Appendix A.3.1.

Therefore, all of the previous forms of regularized empirical risk minimization, whether with a convex or bounded loss ρ, can be easily expressed using only convex base losses ℓ and convex auxiliary functions ψ, as follows

$$\hat{\boldsymbol{\theta}}_{VM} \quad \in \quad \arg\min_{\boldsymbol{\theta} \in \Theta} \min_{0 \leq \boldsymbol{\eta} \leq 1} \boldsymbol{\eta}' \boldsymbol{\ell}(\mathbf{y} - X\boldsymbol{\theta}) + \mathbf{1}'\boldsymbol{\psi}(\boldsymbol{\eta}) + \tfrac{\lambda}{2}\|\boldsymbol{\eta}\|_1 \|\boldsymbol{\theta}\|_2^2 \tag{5.27}$$

$$\hat{\boldsymbol{\alpha}}_{VM} \quad \in \quad \arg\min_{\boldsymbol{\alpha}} \min_{0 \leq \boldsymbol{\eta} \leq 1} \boldsymbol{\eta}' \boldsymbol{\ell}(\mathbf{y} - K\boldsymbol{\alpha}) + \mathbf{1}'\boldsymbol{\psi}(\boldsymbol{\eta}) + \tfrac{\lambda}{2}\|\boldsymbol{\eta}\|_1 \boldsymbol{\alpha}' K \boldsymbol{\alpha}. \tag{5.28}$$

Note that we have added a regularizer $\|\boldsymbol{\eta}\|_1/n$, which increases robustness by encouraging η weights to prefer small values. This particular form of regularization has two advantages: (i) it is a smooth function of $\boldsymbol{\eta}$ on $0 \leq \boldsymbol{\eta} \leq 1$ (since $\|\boldsymbol{\eta}\|_1 = \mathbf{1}'\boldsymbol{\eta}$ in this case), and (ii) it enables a tight convex approximation strategy, as we will see below. (Other forms of robust regression can be expressed in a similar framework; see Appendix A.3.3.)

These formulations are all convex in the parameters given the auxiliary weights, and vice versa. However, they are not jointly convex in the optimization variables (i.e. in $\boldsymbol{\theta}$ and $\boldsymbol{\eta}$, or in $\boldsymbol{\alpha}$ and $\boldsymbol{\eta}$). Therefore, one is not assured that the problems (5.27)–(5.28) have only global minima; in fact local minima exist and global minima cannot be easily found (or even verified).

5.4 Computationally Efficient Approximation

We present a general approximation strategy for the variational regression estimators above that can guarantee polynomial run-time while ensuring certain robustness and consistency properties. The approximation is significantly tighter than the existing work (Yu et al., 2010), which allows us to achieve stronger guarantees while providing better empirical performance. In developing our estimator we follow standard methodology from combinatorial optimization that was partly applied also in Chapter 3 and in Chapter 4: given an intractable optimization problem, first formulate a (hopefully tight) convex relaxation that provides a lower bound on the objective, then round the relaxed minimizer back to the feasible space, hopefully verifying that the rounded solution preserves desirable properties, and finally re-optimize the rounded solution to refine the result; see e.g. (Peng & Wei, 2007).

To maintain generality, we formulate the approximate estimator in the RKHS setting. Consider (5.28). Although the problem is obviously convex in $\boldsymbol{\alpha}$ given $\boldsymbol{\eta}$, and vice versa, it is not jointly convex (recall the assumption that ℓ and ψ are

both convex functions). This suggests that an obvious computational strategy for computing the estimator (5.28) is to alternate between $\boldsymbol{\alpha}$ and $\boldsymbol{\eta}$ optimizations (or use heuristic methods (Nunkesser & Morell, 2010)), but this cannot guarantee anything other than local solutions (and thus may not even achieve any of the desired theoretical properties associated with the estimator).

Reformulation: We first need to reformulate the problem to allow a tight relaxation. Let $\Delta(\boldsymbol{\eta})$ denote putting a vector $\boldsymbol{\eta}$ on the main diagonal of a square matrix, and let $\circ$ denote componentwise multiplication. Since ℓ is closed and convex by assumption, we know that $\ell(r) = \sup_\nu \nu r - \ell^*(\nu)$, where ℓ^* is the Fenchel conjugate of ℓ (Boyd & Vandenberghe, 2004). This allows (5.28) to be reformulated as follows.

Lemma 4.
$$\min_{0\leq\boldsymbol{\eta}\leq 1} \min_{\boldsymbol{\alpha}} \boldsymbol{\eta}^T\boldsymbol{\ell}(\mathbf{y} - K\boldsymbol{\alpha}) + \mathbf{1}^T\boldsymbol{\psi}(\boldsymbol{\eta}) + \tfrac{\lambda}{2}\|\boldsymbol{\eta}\|_1\boldsymbol{\alpha}^T K\boldsymbol{\alpha} \quad (5.29)$$

$$= \min_{0\leq\boldsymbol{\eta}\leq 1} \sup_{\boldsymbol{\nu}} \mathbf{1}^T\boldsymbol{\psi}(\boldsymbol{\eta}) - \boldsymbol{\eta}^T(\boldsymbol{\ell}^*(\boldsymbol{\nu}) - \Delta(\mathbf{y})\boldsymbol{\nu}) - \tfrac{1}{2\lambda}\boldsymbol{\nu}^T\left(K \circ (\boldsymbol{\eta}\|\boldsymbol{\eta}\|_1^{-1}\boldsymbol{\eta}^T)\right)\boldsymbol{\nu}, \quad (5.30)$$

where the function evaluations are componentwise.

Proof. The lemma amounts to dualizing the interior convex minimization problem

$$\min_{\boldsymbol{\alpha}} \boldsymbol{\eta}'\boldsymbol{\ell}(\mathbf{y} - K\boldsymbol{\alpha}) + \mathbf{1}'\boldsymbol{\psi}(\boldsymbol{\eta}) + \tfrac{\lambda}{2}\|\boldsymbol{\eta}\|_1\boldsymbol{\alpha}' K\boldsymbol{\alpha}. \quad (5.31)$$

Recall from the main body that the function ℓ is assumed to be closed and convex. Therefore, strong Fenchel duality holds:

$$\boldsymbol{\eta}'\boldsymbol{\ell}(\mathbf{y} - K\boldsymbol{\alpha}) \;=\; \sup_{\boldsymbol{\nu}} \boldsymbol{\nu}'\Delta(\boldsymbol{\eta})(\mathbf{y} - K\boldsymbol{\alpha}) - \boldsymbol{\eta}'\boldsymbol{\ell}^*(\boldsymbol{\nu}) \quad (5.32)$$

and we can therefore re-write (5.31) as

$$\begin{aligned}(5.31) \;&=\; \min_{\boldsymbol{\alpha}} \sup_{\boldsymbol{\nu}} \mathbf{1}'\boldsymbol{\psi}(\boldsymbol{\eta}) - \boldsymbol{\eta}'\boldsymbol{\ell}^*(\boldsymbol{\nu}) + \boldsymbol{\nu}'\Delta(\mathbf{y})\boldsymbol{\eta} - \boldsymbol{\nu}'\Delta(\boldsymbol{\eta})K\boldsymbol{\alpha} \\ &\qquad + \frac{\lambda}{2}\|\boldsymbol{\eta}\|_1\boldsymbol{\alpha}' K\boldsymbol{\alpha} \quad (5.33)\\ &=\; \sup_{\boldsymbol{\nu}} \min_{\boldsymbol{\alpha}} \mathbf{1}'\boldsymbol{\psi}(\boldsymbol{\eta}) - \boldsymbol{\eta}'\boldsymbol{\ell}^*(\boldsymbol{\nu}) + \boldsymbol{\nu}'\Delta(\mathbf{y})\boldsymbol{\eta} - \boldsymbol{\nu}'\Delta(\boldsymbol{\eta})K\boldsymbol{\alpha} \\ &\qquad + \frac{\lambda}{2}\|\boldsymbol{\eta}\|_1\boldsymbol{\alpha}' K\boldsymbol{\alpha}, \quad (5.34)\end{aligned}$$

where (5.34) follows by strong duality (since for all fixed $\boldsymbol{\nu}$ in (5.34) the sublevel sets in $\boldsymbol{\alpha}$ are bounded (Boyd & Vandenberghe, 2004, Ch.3)).

Finally, $\boldsymbol{\alpha}$ can be eliminated from the inner problem by solving for a critical point (the inner objective is convex in $\boldsymbol{\alpha}$). Taking the gradient in $\boldsymbol{\alpha}$ and setting to zero gives the system of equations $\nabla_{\boldsymbol{\alpha}} = K\Delta(\boldsymbol{\eta})\boldsymbol{\nu} + \lambda\|\boldsymbol{\eta}\|_1 K\boldsymbol{\alpha} = 0$, which is satisfied by $\boldsymbol{\alpha} = \Delta(\boldsymbol{\eta})\boldsymbol{\nu}/(\lambda\|\boldsymbol{\eta}\|_1)$. Substituting this solution back into (5.34) yields

$$
\begin{aligned}
(5.34) &= \sup_{\boldsymbol{\nu}} \mathbf{1}'\boldsymbol{\psi}(\boldsymbol{\eta}) - \boldsymbol{\eta}'(\boldsymbol{\ell}^*(\boldsymbol{\nu}) - \Delta(\mathbf{y})\boldsymbol{\nu}) - \frac{1}{2\lambda\|\boldsymbol{\eta}\|_1}\boldsymbol{\nu}'\Delta(\boldsymbol{\eta})K\Delta(\boldsymbol{\eta})\boldsymbol{\nu} \quad (5.35)\\
&= \sup_{\boldsymbol{\nu}} \mathbf{1}'\boldsymbol{\psi}(\boldsymbol{\eta}) - \boldsymbol{\eta}'(\boldsymbol{\ell}^*(\boldsymbol{\nu}) - \Delta(\mathbf{y})\boldsymbol{\nu}) - \frac{1}{2\lambda}\boldsymbol{\nu}'\left(K \circ (\boldsymbol{\eta}\|\boldsymbol{\eta}\|_1^{-1}\boldsymbol{\eta}')\right)\boldsymbol{\nu}, \quad (5.36)
\end{aligned}
$$

establishing the lemma. ∎

Although no relaxation has been introduced, the new form (5.30) has a more convenient structure.

Relaxation: Let $N = \boldsymbol{\eta}\|\boldsymbol{\eta}\|_1^{-1}\boldsymbol{\eta}^T$ and note that, since $0 \leq \boldsymbol{\eta} \leq 1$, N must satisfy a number of useful properties. We can summarize these by formulating a constraint set $N \in \mathcal{N}_{\boldsymbol{\eta}}$ given by:

$$
\begin{aligned}
\mathcal{N}_{\boldsymbol{\eta}} &= \{N : N \succcurlyeq 0, N\mathbf{1} = \boldsymbol{\eta}, \operatorname{rank}(N) = 1\} \quad (5.37)\\
\mathcal{M}_{\boldsymbol{\eta}} &= \{M : M \succcurlyeq 0, M\mathbf{1} = \boldsymbol{\eta}, \operatorname{tr}(M) \leq 1\}. \quad (5.38)
\end{aligned}
$$

Unfortunately, the set $\mathcal{N}_{\boldsymbol{\eta}}$ is not convex because of the rank constraint. However, relaxing this constraint leads to a set $\mathcal{M}_{\boldsymbol{\eta}} \supseteq \mathcal{N}_{\boldsymbol{\eta}}$ which preserves much of the key structure, as we verify below.

Lemma 5.

$$
\begin{aligned}
(5.30) &= \min_{0\leq\boldsymbol{\eta}\leq 1} \min_{N\in\mathcal{N}_{\boldsymbol{\eta}}} \sup_{\boldsymbol{\nu}} \mathbf{1}^T\boldsymbol{\psi}(\boldsymbol{\eta}) - \boldsymbol{\eta}^T(\boldsymbol{\ell}^*(\boldsymbol{\nu}) - \Delta(\mathbf{y})\boldsymbol{\nu}) - \tfrac{1}{2\lambda}\boldsymbol{\nu}^T\left(K \circ N\right)\boldsymbol{\nu} \quad (5.39)\\
&\geq \min_{0\leq\boldsymbol{\eta}\leq 1} \min_{M\in\mathcal{M}_{\boldsymbol{\eta}}} \sup_{\boldsymbol{\nu}} \mathbf{1}^T\boldsymbol{\psi}(\boldsymbol{\eta}) - \boldsymbol{\eta}^T(\boldsymbol{\ell}^*(\boldsymbol{\nu}) - \Delta(\mathbf{y})\boldsymbol{\nu}) - \tfrac{1}{2\lambda}\boldsymbol{\nu}^T\left(K \circ M\right)\boldsymbol{\nu} \quad (5.40)
\end{aligned}
$$

using the fact that $\mathcal{N}_{\boldsymbol{\eta}} \subseteq \mathcal{M}_{\boldsymbol{\eta}}$.

Proof. First, to prove the equality (5.39) consider any fixed $\boldsymbol{\eta} \in [0,1]^n$. We will show that $N = \boldsymbol{\eta}\|\boldsymbol{\eta}\|_1^{-1}\boldsymbol{\eta}' \Leftrightarrow N \in \mathcal{N}_{\boldsymbol{\eta}}$, which will immediately yield the equality. The direction $\Rightarrow$ can be verified by a simple check. To prove $\Leftarrow$, assume $N \in \mathcal{N}_{\boldsymbol{\eta}}$. Since $N \succcurlyeq 0$ and $\mathrm{rank}(N) = 1$ we know that $N = \mathbf{q}\lambda\mathbf{q}'$ for some $\lambda > 0$ and $\mathbf{q}$ such that $\|\mathbf{q}\| = 1$. But now, since $N\mathbf{1} = \boldsymbol{\eta}$, it must follow that $\mathbf{q} = \boldsymbol{\eta}/(\lambda\boldsymbol{\eta}'\mathbf{1})$, hence $\mathbf{q} = \boldsymbol{\eta}/\|\boldsymbol{\eta}\|$ and $\lambda\mathbf{q}'\mathbf{1} = \|\boldsymbol{\eta}\|$. Therefore, $\lambda = \|\boldsymbol{\eta}\|/\mathbf{q}'\mathbf{1} = \|\boldsymbol{\eta}\|^2/(\boldsymbol{\eta}'\mathbf{1})$. That is, N must have the form $N = \mathbf{q}\lambda\mathbf{q}' = \boldsymbol{\eta}\|\boldsymbol{\eta}\|_1^{-1}\boldsymbol{\eta}'$.

The inequality (5.40) then follows from the above argument, and the fact that $\|\boldsymbol{\eta}\|^2/(\boldsymbol{\eta}'\mathbf{1}) \leq 1$, which implies $\mathrm{tr}(N) \leq 1$ for any $N \in \mathcal{N}_{\boldsymbol{\eta}}$. Therefore $\mathcal{N}_{\boldsymbol{\eta}} \subseteq \mathcal{M}_{\boldsymbol{\eta}}$. ∎

Crucially, the constraint set $\{(\boldsymbol{\eta}, M) : 0 \leq \boldsymbol{\eta} \leq 1, M \in \mathcal{M}_{\boldsymbol{\eta}}\}$ is jointly convex in $\boldsymbol{\eta}$ and M, thus (5.40) is a convex-concave min-max problem. To see why, note that the inner objective function is jointly convex in $\boldsymbol{\eta}$ and M, and concave in $\boldsymbol{\nu}$. Since a pointwise maximum of convex functions is convex, the problem is convex in $(\boldsymbol{\eta}, M)$ (Boyd & Vandenberghe, 2004, Ch.3). We conclude that all local minima in $(\boldsymbol{\eta}, M)$ are global. Therefore, (5.40) provides the foundation for an efficiently solvable relaxation.

Rounding: Unfortunately the solution to M in (5.40) does not allow direct recovery of an estimator $\boldsymbol{\alpha}$ achieving the same objective value in (5.29), unless M satisfies $\mathrm{rank}(M) = 1$. In general we first need to round M to a rank 1 solution. Fortunately, a trivial rounding procedure is available: we simply use $\boldsymbol{\eta}$ (ignoring M) and re-solve for $\boldsymbol{\alpha}$ in (5.29). This is equivalent to replacing M with the rank 1 matrix $\tilde{N} = \boldsymbol{\eta}\|\boldsymbol{\eta}\|_1^{-1}\boldsymbol{\eta}^T \in \mathcal{N}_{\boldsymbol{\eta}}$, which restores feasibility in the original problem. Of course, such a rounding step will generally increase the objective value.

Reoptimization: Finally, the rounded solution can be locally improved by alternating between $\boldsymbol{\eta}$ and $\boldsymbol{\alpha}$ updates in (5.29) (or using any other local optimization method), yielding the final estimate $\tilde{\boldsymbol{\alpha}}$.

5.5 Properties

Although a tight *a priori* bound on the size of the optimality gap is difficult to achieve, a rigorous bound on the optimality gap can be recovered *post hoc* once after the re-optimized estimator is computed. Let R_0 denote the minimum value of (5.29) (not efficiently computable); let R_1 denote the minimum value of (5.40) (the relaxed solution); let R_2 denote the value of (5.29) achieved by freezing $\boldsymbol{\eta}$ from the relaxed solution but re-optimizing $\boldsymbol{\alpha}$ (the rounded solution); and finally let R_3 denote the value of (5.29) achieved by re-optimizing $\boldsymbol{\eta}$ and $\boldsymbol{\alpha}$ from the rounded solution (the re-optimized solution). Clearly we have the relationships $R_1 \leq R_0 \leq R_3 \leq R_2$. An upper bound on the relative optimality gap of the final solution (R_3) can be determined by $(R_3 - R_0)/R_3 \leq (R_3 - R_1)/R_3$, since R_1 and R_3 are both known quantities.

5.5.1 Tractability Properties

Under mild assumptions on ℓ and ψ, computation of the approximate estimator (solving the relaxed problem, rounding, then re-optimizing) admits a polynomial-time solution.

Recall that the approximate estimator is computed in three steps: First, the relaxed lower bound (5.40) is computed, recovering $\boldsymbol{\eta}$ (see Lemma 5). Second, the parameters $\boldsymbol{\alpha}$ are recovered by fixing $\boldsymbol{\eta}$ and re-solving for $\boldsymbol{\alpha}$ in (5.29) (see Lemma 4 in Section 5.4). Third, $\boldsymbol{\eta}$ and $\boldsymbol{\alpha}$ are locally reoptimized in (5.29) using the previous $(\boldsymbol{\eta}, \boldsymbol{\alpha})$ as the initial point. We discuss an efficient implementation strategy, and the conditions under which polynomial run-time can be ensured.

For clarity we consider the case $\psi(\eta) = 1-\eta$. (An efficient algorithm is possible for general ψ, but the details unnecessarily complicate the presentation.)

Relaxation: The first step of the approximate estimator is to compute the lower bound (5.40) given in Lemma 5, and recover the optimal $\boldsymbol{\eta}$ and M. This

optimization can be solved efficiently as follows. Recall the definition $M_{\boldsymbol{\eta}} = \{M \succcurlyeq 0, M\mathbf{1} = \boldsymbol{\eta}, \mathrm{tr}(M) \leq 1\}$, apply the definition $\psi(\eta) = 1 - \eta$, and observe

$$\begin{aligned}
(5.40) \quad &= \max_{\boldsymbol{\nu}} \min_{0 \leq \boldsymbol{\eta} \leq 1} \min_{M \in \mathcal{M}_{\boldsymbol{\eta}}} n - \mathbf{1}'\boldsymbol{\eta} - \boldsymbol{\eta}'(\boldsymbol{\ell}^*(\boldsymbol{\nu}) - \Delta(\mathbf{y})\boldsymbol{\nu}) \\
&\qquad -\frac{1}{2\lambda}\boldsymbol{\nu}'\left(K \circ M\right)\boldsymbol{\nu} && (5.41) \\
&= \max_{\boldsymbol{\nu}} \max_{\mathbf{a} \geq 0, \mathbf{b} \geq 0} \min_{\boldsymbol{\eta}} \min_{M \in \mathcal{M}_{\boldsymbol{\eta}}} n - \mathbf{1}'\boldsymbol{\eta} - \boldsymbol{\eta}'(\boldsymbol{\ell}^*(\boldsymbol{\nu}) - \Delta(\mathbf{y})\boldsymbol{\nu}) \\
&\qquad -\frac{1}{2\lambda}\boldsymbol{\nu}'\left(K \circ M\right)\boldsymbol{\nu} - \mathbf{a}'M\mathbf{1} + \mathbf{b}'(M\mathbf{1} - \mathbf{1}) && (5.42) \\
&= \max_{\boldsymbol{\nu}} \max_{\mathbf{a} \geq 0, \mathbf{b} \geq 0} \min_{M \succcurlyeq 0, \mathrm{tr}(M) \leq 1} n - \mathbf{1}'M\mathbf{1} - \mathbf{1}'M(\boldsymbol{\ell}^*(\boldsymbol{\nu}) - \Delta(\mathbf{y})\boldsymbol{\nu}) \\
&\qquad -\frac{1}{2\lambda}\boldsymbol{\nu}'\left(K \circ M\right)\boldsymbol{\nu} - \mathbf{a}'M\mathbf{1} + \mathbf{b}'(M\mathbf{1} - \mathbf{1}) && (5.43) \\
&= \max_{\boldsymbol{\nu}, \mathbf{a} \geq 0, \mathbf{b} \geq 0} n - \mathbf{b}'\mathbf{1} - \max_{M \succcurlyeq 0, \mathrm{tr}(M) \leq 1} \mathbf{1}'M\mathbf{1} + \mathbf{1}'M(\boldsymbol{\ell}^*(\boldsymbol{\nu}) - \Delta(\mathbf{y})\boldsymbol{\nu}) \\
&\qquad +\frac{1}{2\lambda}\boldsymbol{\nu}'\left(K \circ M\right)\boldsymbol{\nu} + \mathbf{a}'M\mathbf{1} - \mathbf{b}'M\mathbf{1}, && (5.44)
\end{aligned}$$

where (5.41) follows by Sion's minimax theorem (Rockafellar, 1970, Cor.37.3.2); (5.42) follows by Lagrange duality; (5.43) follows by substituting $M\mathbf{1} = \boldsymbol{\eta}$ to replace the constraint and eliminate $\boldsymbol{\eta}$; and (5.44) is simple regrouping. Therefore (5.44) can be solved as a nonsmooth maximization:

$$\max_{\boldsymbol{\nu}, \mathbf{a} \geq 0, \mathbf{b} \geq 0} n - \mathbf{b}'\mathbf{1} - f(\boldsymbol{\nu}, \mathbf{a}, \mathbf{b}), \tag{5.45}$$

where

$$\begin{aligned}
f(\boldsymbol{\nu}, \mathbf{a}, \mathbf{b}) &= \max_{M \succcurlyeq 0, \mathrm{tr}(M) \leq 1} \mathbf{1}'M\mathbf{1} + \mathbf{1}'M\mathbf{c} + \frac{1}{2\lambda}\boldsymbol{\nu}'\left(K \circ M\right)\boldsymbol{\nu} + \mathbf{a}'M\mathbf{1} - \mathbf{b}'M\mathbf{1} && (5.46) \\
&= \frac{1}{2}\max_{M \succcurlyeq 0, \mathrm{tr}(M) \leq 1} \mathrm{tr}(MC(\boldsymbol{\nu}, \mathbf{a}, \mathbf{b})), && (5.47)
\end{aligned}$$

such that

$$\begin{aligned}
C(\boldsymbol{\nu}, \mathbf{a}, \mathbf{b}) &= 2\mathbf{1}\mathbf{1}' + \mathbf{1}\mathbf{c}(\boldsymbol{\nu})' + \mathbf{c}(\boldsymbol{\nu})\mathbf{1}' + \frac{1}{\lambda}\left(K \circ \boldsymbol{\nu}\boldsymbol{\nu}'\right) + \mathbf{a}\mathbf{1}' + \mathbf{1}\mathbf{a}' - \mathbf{b}\mathbf{1}' - \mathbf{1}\mathbf{b}' && (5.48) \\
\mathbf{c}(\boldsymbol{\nu}) &= \boldsymbol{\ell}^*(\boldsymbol{\nu}) - \Delta(\mathbf{y})\boldsymbol{\nu}. && (5.49)
\end{aligned}$$

Note that in the maximization problem (5.45), $f(\boldsymbol{\nu}, \mathbf{a}, \mathbf{b})$ must be convex, since it is a pointwise maximum of linear functions (Boyd & Vandenberghe, 2004, Ch.3); hence (5.45) is a concave maximization problem. Each evaluation of $f(\boldsymbol{\nu}, \mathbf{a}, \mathbf{b})$ requires no more than $O(n^3)$ time, since (5.47) can be solved by computing the maximum eigenvector of $C(\boldsymbol{\nu}, \mathbf{a}, \mathbf{b})$ (Overton & Womersley, 1993). Moreover, a subgradient in $(\boldsymbol{\nu}, \mathbf{a}, \mathbf{b})$ can easily be recovered from the maximizer M, based on the fact that

$$\partial f \quad \ni \quad \begin{bmatrix} \Delta(M\mathbf{1})(\boldsymbol{\ell}'(\boldsymbol{\nu}) - \mathbf{y}) + \frac{1}{\lambda}(K \circ M)\boldsymbol{\nu} \\ M\mathbf{1} \\ -M\mathbf{1} \end{bmatrix} \tag{5.50}$$

by Danskin's theorem (Bertsekas, 1995, Ch.6). (Note that the maximizer M might not be unique, so f is nonsmooth at points where this occurs.) Finally, at a solution $(\boldsymbol{\nu}, \mathbf{a}, \mathbf{b}; M)$ the weight vector $\boldsymbol{\eta}$ is recovered via $\boldsymbol{\eta} = M\mathbf{1}$.

Therefore, computing the relaxed solution requires solving a nonsmooth concave maximization over $3n$ variables, where each function evaluation (and subgradient) can be computed in $O(n^3)$ time. An ellipsoid algorithm can therefore be used to solve (5.45) in polynomial-time (Nesterov, 2003) (Primak & Kheyfets, 1995).

Rounding: The rounding procedure involves a simple, smooth convex minimization of $\boldsymbol{\alpha}$ in (5.29), where $\boldsymbol{\eta}$ is fixed from the relaxation step. This problem can be solved in polynomial-time provided only that the base loss ℓ (assumed convex) is also self-concordant (Nesterov & Nimirovskii, 1994).

Reoptimization: Finally, in the reopimization step, both $\boldsymbol{\eta}$ and $\boldsymbol{\alpha}$ are jointly (and locally) optimized in (5.29), starting from the solution above. A local optimum can, once again, be recovered in polynomial-time adding only the assumption that ψ (in addition to being convex) is also self-concordant.

Therefore, the estimation procedure requires only polynomial-time under the stated assumptions.

Once $\boldsymbol{\eta}$ is recovered from the relaxed solution, the subsequent optimizations of

(5.29) can be solved efficiently under weak assumptions about ℓ and ψ; namely that they both satisfy the self-concordance and polynomial-time computation properties discussed in Section 5.2.

5.5.2 Robustness Properties

Despite the approximation, the relaxation remains sufficiently tight to preserve some of the robustness properties of bounded loss minimization. To establish the robustness (and consistency) properties, we will need to make use of a specific technical definition of *outliers* and *inliers*.

Definition 2 (Outliers and Inliers)**.** For an L-Lipschitz loss ℓ, an outlier is a point (x_i, y_i) that satisfies $\ell(y_i) > L^2 K_{ii}/(2\lambda) - \psi'(0)$, while an inlier satisfies $\ell(y_i) + L^2 K_{ii}/(2\lambda) < -\psi'(1)$.

Theorem 5. Assume the loss ρ is bounded and has a variational representation (5.26) such that ℓ is Lipschitz-continuous and ψ' is bounded. Also assume there is at least one (unperturbed) inlier, and consider the perturbation of a single data point (y_1, x_1). Under the following conditions, the rounded (re-optimized) estimator maintains bounded response:
(i) If either y_1 remains bounded, or $\kappa(x_1, x_1)$ remains bounded.
(ii) If $|y_1| \to \infty$, $\kappa(x_1, x_1) \to \infty$ and $\ell(y_1)/\kappa(x_1, x_1) \to \infty$.

Note that the latter condition causes *any* convex loss ℓ to demonstrate unbounded response. Therefore, the approximate estimator is strictly more robust (in terms of bounded response) than regularized empirical risk minimization with a convex loss ℓ.

To prove this theorem, first let

$$
\begin{aligned}
R_0 &= \min_{0\le\boldsymbol{\eta}\le 1}\min_{\boldsymbol{\alpha}} \boldsymbol{\eta}'\boldsymbol{\ell}(\mathbf{y}-K\boldsymbol{\alpha})+\mathbf{1}'\boldsymbol{\psi}(\boldsymbol{\eta})+\frac{\lambda}{2}\|\boldsymbol{\eta}\|_1\boldsymbol{\alpha}'K\boldsymbol{\alpha} && (5.51)\\
R_1 &= \min_{0\le\boldsymbol{\eta}\le 1}\min_{M\in\mathcal{M}_{\boldsymbol{\eta}}}\sup_{\boldsymbol{\nu}} \mathbf{1}'\boldsymbol{\psi}(\boldsymbol{\eta})+\boldsymbol{\eta}'(\Delta(\mathbf{y})\boldsymbol{\nu}-\boldsymbol{\ell}^*(\boldsymbol{\nu}))\\
&\qquad -\frac{1}{2\lambda}\boldsymbol{\nu}'\left(K\circ M\right)\boldsymbol{\nu} && (5.52)\\
R_2 &= \sup_{\boldsymbol{\nu}} \mathbf{1}'\boldsymbol{\psi}(\boldsymbol{\eta})+\boldsymbol{\eta}'(\Delta(\mathbf{y})\boldsymbol{\nu}-\boldsymbol{\ell}^*(\boldsymbol{\nu}))\\
&\qquad -\frac{1}{2\lambda}\boldsymbol{\nu}'\left(K\circ(\boldsymbol{\eta}\|\boldsymbol{\eta}\|_1^{-1}\boldsymbol{\eta}')\right)\boldsymbol{\nu} && (5.53)\\
&= \min_{\boldsymbol{\alpha}} \boldsymbol{\eta}'\boldsymbol{\ell}(\mathbf{y}-K\boldsymbol{\alpha})+\mathbf{1}'\boldsymbol{\psi}(\boldsymbol{\eta})+\frac{\lambda}{2}\|\boldsymbol{\eta}\|_1\boldsymbol{\alpha}'K\boldsymbol{\alpha}, && (5.54)
\end{aligned}
$$

where $\boldsymbol{\eta}$ in (5.53) is fixed at the optimal assignment determined by (5.52), and (5.54) follows by Lemma 4. Here, R_0 denotes the objective value obtained by the (intractable) oracle minimizer, R_1 is the objective value obtained by the relaxed solution, and R_2 denotes the objective value obtained by the rounded solution.

It is immediate that R_0 and R_1 must be bounded, since

$$R_1 \;\le\; R_0 \;\le\; n\psi(0) \;<\; \infty \tag{5.55}$$

where the first inequality follows from Lemma 5, and the second inequality is achieved by choosing $\boldsymbol{\eta} = \boldsymbol{\alpha} = 0$ in (5.52). The key question is whether R_2, the rounded objective value, remains bounded. Once this is established for each of the cases, we finally show that this will imply that $\|\hat{f}_2\|_{\mathcal{H}}$ remains bounded, and the theorem will be proved.

Proof for Case (i), Bounded y

Proof. Assume that $\mathbf{y}$ remains bounded. We will need to make use of the fact that, since ℓ is closed and convex, we have

$$\max_{\nu}\eta(y\nu-\ell^*(\nu)) \;=\; \eta\max_{\nu} y\nu-\ell^*(\nu) \;=\; \eta\ell(y). \tag{5.56}$$

Therefore, from (5.53) and (5.56) it follows that

$$\begin{aligned} R_2 &\leq \sup_{\boldsymbol{\nu}} \mathbf{1}'\boldsymbol{\psi}(\boldsymbol{\eta}) - \boldsymbol{\eta}'(\boldsymbol{\ell}^*(\boldsymbol{\nu}) - \Delta(\mathbf{y})\boldsymbol{\nu}) && (5.57)\\ &= \mathbf{1}'\boldsymbol{\psi}(\boldsymbol{\eta}) + \boldsymbol{\eta}'\boldsymbol{\ell}(\mathbf{y}) && (5.58)\\ &\leq n\gamma && (5.59) \end{aligned}$$

such that $\gamma = \max_{0\leq\eta\leq 1}\psi(\eta) + \max_{|y|\leq B}\ell(y) < \infty$. ∎

Proof for Case (i), Bounded K

Proof. Assume K remains bounded; in particular that $|K_{ij}| \leq B$ for some $B < \infty$. We will need to make use of the fact that ℓ is Lipschitz-continuous. In particular, let ℓ have a Lipschitz-constant of L. Then it follows from Lemma 4 and the definition of Fenchel conjugate (Rockafellar, 1970) that

$$\begin{aligned} R_2 &= \sup_{-L\leq\boldsymbol{\nu}\leq L} \mathbf{1}'\boldsymbol{\psi}(\boldsymbol{\eta}) - \boldsymbol{\eta}'(\boldsymbol{\ell}^*(\boldsymbol{\nu}) - \Delta(\mathbf{y})\boldsymbol{\nu})\\ &\quad -\frac{1}{2\lambda}\boldsymbol{\nu}'\left(K\circ(\boldsymbol{\eta}\|\boldsymbol{\eta}\|_1^{-1}\boldsymbol{\eta}')\right)\boldsymbol{\nu} && (5.60)\\ &\leq \sup_{-L\leq\boldsymbol{\nu}\leq L} \mathbf{1}'\boldsymbol{\psi}(\boldsymbol{\eta}) - \boldsymbol{\eta}'(\ell^*(\boldsymbol{\nu}) - \Delta(\mathbf{y})\boldsymbol{\nu}) && (5.61)\\ &= \sup_{-L\leq\boldsymbol{\nu}\leq L} \mathbf{1}'\boldsymbol{\psi}(\boldsymbol{\eta}) - \boldsymbol{\eta}'(\ell^*(\boldsymbol{\nu}) - \Delta(\mathbf{y})\boldsymbol{\nu}) - \frac{1}{2\lambda}\boldsymbol{\nu}'(K\circ M)\boldsymbol{\nu}\\ &\quad +\frac{1}{2\lambda}\boldsymbol{\nu}'(K\circ M)\boldsymbol{\nu} && (5.62)\\ &\leq \sup_{-L\leq\boldsymbol{\nu}\leq L}\left\{\mathbf{1}'\boldsymbol{\psi}(\boldsymbol{\eta}) - \boldsymbol{\eta}'(\ell^*(\boldsymbol{\nu}) - \Delta(\mathbf{y})\boldsymbol{\nu}) - \frac{1}{2\lambda}\boldsymbol{\nu}'(K\circ M)\boldsymbol{\nu}\right\}\\ &\quad + \sup_{-L\leq\boldsymbol{\nu}\leq L}\frac{1}{2\lambda}\boldsymbol{\nu}'(K\circ M)\boldsymbol{\nu} && (5.63)\\ &= R_1 + \sup_{-L\leq\boldsymbol{\nu}\leq L}\frac{1}{2\lambda}\boldsymbol{\nu}'(K\circ M)\boldsymbol{\nu} && (5.64)\\ &\leq n\psi(0) + \sup_{-L\leq\boldsymbol{\nu}\leq L}\frac{1}{2\lambda}\boldsymbol{\nu}'(K\circ M)\boldsymbol{\nu}. && (5.65) \end{aligned}$$

Therefore it suffices to bound

$$\begin{aligned} & \sup_{-L\leq\boldsymbol{\nu}\leq L} \boldsymbol{\nu}'(K\circ M)\boldsymbol{\nu} \\ &\leq L^2\,\mathrm{tr}(KM) && (5.66)\\ &\leq L^2Bn, && (5.67)\end{aligned}$$

since $M \succcurlyeq 0$, $\mathrm{tr}(M) \leq 1$, and $\lambda_{max}(K) \leq Bn$ (Horn & Johnson, 1985, p.292). ∎

Proof for Case (ii)

Proof. The proof in this case proceeds differently than the previous cases. Here we assume we are given a fixed data set $(x_1, y_1), ..., (x_n, y_n)$, where only the first data point (x_1, y_1) is perturbed (without loss of generality), so that $|y_1| \to \infty$, $\kappa(x_1, x_1) \to \infty$ and $\ell(y_1)/\kappa(x_1, x_1) \to \infty$. Note that such a point must eventually become an outlier. We will show that this forces the corresponding weight η_1 to eventually satisfy $\eta_1 = 0$ in the relaxed solution (5.52), which will automatically imply that the rounded value R_2 stays at the same finite value thereafter (since no other data point is perturbed).

Consider the inner objective in (5.52):

$$\begin{aligned} & \mathbf{1}'\boldsymbol{\psi}(\boldsymbol{\eta}) + \boldsymbol{\eta}'(\Delta(\mathbf{y})\boldsymbol{\nu} - \boldsymbol{\ell}^*(\boldsymbol{\nu})) - \frac{1}{2\lambda}\boldsymbol{\nu}'(K\circ M)\boldsymbol{\nu} && (5.68)\\ &= \mathbf{1}'\boldsymbol{\psi}(M\mathbf{1}) + \boldsymbol{\eta}'(\Delta(\mathbf{y})\boldsymbol{\nu} - \boldsymbol{\ell}^*(\boldsymbol{\nu})) - \frac{1}{2\lambda}\boldsymbol{\nu}'(K\circ M)\boldsymbol{\nu}. && (5.69)\end{aligned}$$

The gradients with respect to the parameters for this objective are given by

$$\begin{aligned} \nabla_{\boldsymbol{\nu}} &= \Delta(\boldsymbol{\eta})(\mathbf{y} - \boldsymbol{\ell}^*(\boldsymbol{\nu})') - \frac{1}{\lambda}(K\circ M)\boldsymbol{\nu} && (5.70)\\ \nabla_M &= \boldsymbol{\psi}'(M\mathbf{1})\mathbf{1}' + (\Delta(\mathbf{y})\boldsymbol{\nu} - \boldsymbol{\ell}^*(\boldsymbol{\nu}))\mathbf{1}' - \frac{1}{2\lambda}(K\circ\boldsymbol{\nu}\boldsymbol{\nu}') && (5.71)\\ \frac{d}{d\eta_1} &= \psi'(\eta_1) + y_1\nu_1 - \ell^*(\nu_1) - \frac{1}{2\lambda n}K_{1:}\boldsymbol{\nu}\nu_1. && (5.72)\end{aligned}$$

Since ℓ is Lipschitz-continuous with Lipschitz constant L, we know that $|\nu| \leq L$ (Rockafellar, 1970), hence

$$\left|\frac{1}{2\lambda n}K_{1:}\boldsymbol{\nu}\nu_1\right| \leq \frac{\max|K_{1:}|}{2\lambda n}L^2 n = \frac{L^2}{2\lambda}K_{11}. \quad (5.73)$$

Now consider the tentative assignment $M_{1:} = \mathbf{0}'$, $\eta_1 = 0$. At this assignment, all of the quadratic terms in ν_1 have been nullified in (5.68) and (5.69), leaving the optimization over ν_1 as

$$\max_{\nu_1} y_1\nu_1 - \ell^*(\nu_1) \quad = \quad \ell(y_1) \tag{5.74}$$

by (5.56). Now note that at this solution for ν_1 we have

$$\frac{d}{d\eta_1} \quad = \quad \psi'(\eta_1) + \ell(y_1) - \frac{1}{2\lambda n} K_{1:}\boldsymbol{\nu}\nu_1 \tag{5.75}$$

$$\frac{d}{d\eta_1}\bigg|_{\eta_1=0} \quad = \quad \psi'(0) + \ell(y_1) - \frac{1}{2\lambda n} K_{1:}\boldsymbol{\nu}\nu_1. \tag{5.76}$$

Therefore, if $\ell(y_1) > \frac{L^2}{2\lambda}K_{11} - \psi'(0)$ (the outlier condition) then $\frac{d}{d\eta_1}\Big|_{\eta_1=0} > 0$, which implies that η_1 stays at 0. We conclude that once the outlier condition is achieved (guaranteed by the assumptions), R_2 retains the same finite value after all subsequent perturbations of (x_1, y_1), independent of $\ell(y_1)$ and $\kappa(x_1, x_1)$. ∎

Final Step: Bounding $\|\hat{f}\|_{\mathcal{H}}$ Consider the rounded estimate $\hat{f}_2$, corresponding to the solution to (5.54). It remains to bound $\|\hat{f}_2\|_{\mathcal{H}}$ in all the three cases discussed above.

Proof. Observe that

$$\|\hat{f}_2\|_{\mathcal{H}} \quad = \quad \boldsymbol{\alpha}' K \boldsymbol{\alpha} \quad \leq \quad \frac{2R_2}{\lambda \|\boldsymbol{\eta}\|_1}. \tag{5.77}$$

Since R_2 has been proved bounded under the stated assumptions, we only need to consider the behavior of $\|\boldsymbol{\eta}\|_1$, which we do in two cases: $\boldsymbol{\eta} = 0$ and $\boldsymbol{\eta} \neq 0$.

First, assume $\boldsymbol{\eta} = 0$. Then $\boldsymbol{\alpha} = 0$ is an optimal solution of (5.54), implying that $\|\hat{f}_2\|_{\mathcal{H}} = 0$, which is obviously bounded.

So it only remains to consider $\boldsymbol{\eta} \neq 0$. In this case it suffices to bound $\|\boldsymbol{\eta}\|_1$ away from zero. We achieve this by appealing to the assumption that there is at least one *inlier*; that is, an unperturbed point (x_i, y_i), $i \neq 1$, such that $\ell(y_i) + L^2 K_{ii}/(2\lambda) < -\psi'(1)$. For any such point, we can establish that $\eta_i = 1$, and the result will follow.

To show that any inlier gets weight η_i in the relaxed solution (5.52), tentatively consider the assignment $\eta_i = 1$ then recall from (5.72) that

$$\frac{d}{d\eta_i} = \psi'(\eta_i) + y_i\nu_i - \ell^*(\nu_i) - \frac{1}{2\lambda n}K_{i:}\boldsymbol{\nu}\nu_i \tag{5.78}$$

$$\left.\frac{d}{d\eta_i}\right|_{\eta_i=1} = \psi'(1) + y_i\nu_i - \ell^*(\nu_i) - \frac{1}{2\lambda n}K_{i:}\boldsymbol{\nu}\nu_i. \tag{5.79}$$

in the relaxed problem. By (5.74) above we know that $y_i\nu_i - \ell^*(\nu_i) \leq \ell(y_i)$. Furthermore, by (5.73) we know that $|K_{i:}\boldsymbol{\nu}\nu_i/(2\lambda n)| \leq L^2K_{ii}/(2\lambda)$. Therefore, if the condition $\ell(y_i) + L^2K_{ii}/(2\lambda) < -\psi'(1)$ is satisfied then $\left.\frac{d}{d\eta_i}\right|_{\eta_i=1} < 0$, which implies that η_i stays at 1. ∎

5.5.3 Consistency Properties

Finally, we can establish consistency of the approximate estimator in a limited albeit non-trivial setting, although we have yet to establish it generally.

Theorem 6. Assume ℓ is Lipschitz-continuous and $\psi(\eta) = 1 - \eta$. Assume that the data is generated from a mixture of *inliers* and *outliers*, where $P(\text{inlier}) > P(\text{outlier})$. Then the estimate $\hat{\boldsymbol{\theta}}$ produced by the rounded (re-optimized) method is loss consistent.

Proof. From the proof of Theorem 5, we know that the relaxed solution will set $\eta_i = 0$ for all outliers, and $\eta_i = 1$ for all inliers. Since only outliers and inliers are present in the data, the solution $\boldsymbol{\eta}$ will be discrete $\{0, 1\}^n$, and all outliers will be ignored. Thus the reoptimized estimator $\hat{f}_2$ is based only on the inliers, and on such data the estimator is solved by a standard regularized empirical risk minimization. The consistency then follows from the standard results on regularized empirical risk minimization above. ∎

5.6 Experimental Evaluation

We conducted a set of experiments to evaluate the effectiveness of the proposed method compared to standard methods from the literature. Our experimental evaluation was conducted in two parts: first a synthetic experiment where we could control data generation, then an experiment on real data.

The first synthetic experiment was conducted as follows. A target weight vector $\boldsymbol{\theta}$ was drawn from $N(\mathbf{0}, I)$, with $X_{i:}$ sampled uniformly from $[0, 1]^m$, $m = 5$, and outputs y_i computed as $y_i = X_{i:}\boldsymbol{\theta} + \epsilon_i$, $\epsilon_i \sim N(0, \frac{1}{2})$. We then seeded the data set with outliers by randomly re-sampling each y_i and $X_{i:}$ from $N(0, 10^8)$ and $N(0, 10^4)$ respectively, governed by an outlier probability p. Then we randomly sampled 100 points as the training set and another 10000 samples are used for testing. We implemented the proposed method with two different base losses, L_2 and L_1, respectively; referring to these as CvxBndL2 and CvxBndL1. We compared to standard L_2 and L_1 loss minimization, as well as minimizing the Huber minimax loss (Huber) (Huber & Ronchetti, 2009). We also considered standard methods from the robust statistics literature, including the least trimmed square method (LTS) (Rousseeuw & Leroy, 1987a; Rousseeuw & Van Driessen, 2006), and bounded loss minimization based on the Geman-McClure loss (GemMc) (Black & Rangarajan, 1996). Finally we also compared to the alternating minimization strategies outlined at the end of Section 5.3 (AltBndL2 and AltBndL1 for L_2 and L_1 losses respectively), and implemented the strategy described in (Yu et al., 2010).

We added the Tikhonov regularization to each method and the regularization parameter λ was selected (optimally for each method) on a separate validation set. Note that LTS has an extra parameter n', which is the number of inliers. The ideal setting $n' = (1 - p)n$ was granted to LTS. We also tried 30 random restarts for LTS and picked the best result.

All experiments are repeated 10 times and the average root mean square errors (RMSE) (with standard deviations) on the clean test data are reported in Table 5.1.

Methods	Outlier Probability					
	$p = 0.4$		$p = 0.2$		$p = 0.0$	
L2	43.5	(13)	57.6	(21.21)	0.52	(0.01)
L1	4.89	(2.81)	3.6	(2.04)	0.52	(0.01)
Huber	4.89	(2.81)	3.62	(2.02)	0.52	(0.01)
LTS	6.72	(7.37)	8.65	(14.11)	0.52	(0.01)
GemMc	0.53	(0.03)	0.52	(0.02)	0.52	(0.01)
(Yu et al., 2010)	0.52	(0.01)	0.52	(0.01)	0.52	(0.01)
AltBndL2	0.52	(0.01)	0.52	(0.01)	0.52	(0.02)
AltBndL1	0.73	(0.12)	0.74	(0.16)	0.52	(0.01)
CvxBndL2	0.52	(0.01)	0.52	(0.01)	0.52	(0.01)
CvxBndL1	0.53	(0.02)	0.55	(0.05)	0.52	(0.01)

Table 5.1: RMSE on clean test data for an artificial data set with 5 features and 100 training points, with outlier probability p, and 10000 test data points. Results are averaged over 10 repetitions. Standard deviations are given in parentheses.

For $p = 0$ (*i.e.* no outliers), all methods perform well; their RMSEs are close to optimal (1/2, the standard deviation of ϵ_i). However, when outliers start to appear, the result of least squares is significantly skewed, while the results of classic robust statistics methods, Huber, L1 and LTS, indeed turn out to be more robust than the least squares, but nevertheless are still affected significantly. Both implementations of the new method performs comparably to the the non-convex Geman-McClure loss while substantially improving the alternating strategy under the L1 loss. Note that the latter improvement clearly demonstrates that alternating can be trapped in poor local minima. The proposal from (Yu et al., 2010) was not effective in this setting (which differed from the one investigated there).

Next, we conducted an experiment on four real datasets taken from the StatLib repository (sta, 2010) and DELVE (del, 1996). For each data set, we randomly selected 108 points as the training set, and another random 1000 points as the test set. Here the regularization constant is tuned by 10-fold cross validation. To seed outliers, 5% of the training set are randomly chosen and their X and y values are multiplied by 100 and 10000, respectively.

All of these data sets have 8 features, except pumadyn which has 32 features. We also estimated the scale factor on the training set by the mean absolute deviation method, a common method in robust statistics (Maronna et al., 2006). Again, the ideal parameter $n' = (1 - 5\%)n$ is granted to LTS and 30 random restarts are performed. The RMSE on test set for all methods are reported in Table 5.2. It is clear that all methods based on convex losses (L2, L1, Huber) suffer significantly from the added outliers. The method proposed in this section consistently outperform all other methods with a noticeable margin, except on the abalone data set where GemMc performs slightly better (Note that we obtain different results than (Yu et al., 2010) arising from a very different outlier process). Again, we observe evidence that the alternating strategy can be trapped in poor local minima, while the method from (Yu et al., 2010) was less effective. We also measured the relative op-

Methods	Datasets							
	cal-housing		abalone		pumadyn		bank-8fh	
L2	1185	(124.59)	7.93	(0.67)	1.24	(0.42)	18.21	(6.57)
L1	1303	(244.85)	7.30	(0.40)	1.29	(0.42)	6.54	(3.09)
Huber	1221	(119.18)	7.73	(0.49)	1.24	(0.42)	7.37	(3.18)
LTS	533	(398.92)	755.1	(126)	0.32	(0.41)	10.96	(6.67)
GemMc	28	(88.45)	2.30	(0.01)	0.12	(0.12)	0.93	(0.80)
(Yu et al., 2010)	967	(522.40)	8.39	(0.54)	0.81	(0.77)	3.91	(6.18)
AltBndL2	967	(522.40)	8.39	(0.54)	0.81	(0.77)	7.74	(9.40)
AltBndL1	1005	(603.00)	7.30	(0.40)	1.29	(0.42)	1.61	(2.51)
CvxBndL2	9	(0.64)	7.60	(0.86)	0.07	(0.07)	0.20	(0.05)
CvxBndL1	8	(0.28)	2.98	(0.08)	0.08	(0.07)	0.10	(0.07)
Gap(Cvx2)	2e-12	(3e-12)	3e-9	(4e-9)	0.025	(0.052)	0.001	(0.003)
Gap(Cvx1)	0.005	(0.01)	0.001	(0.001)	0.267	(0.269)	0.011	(0.028)

Table 5.2: RMSE on clean test data for 108 training data points and 1000 test data points, with 10 repeats. Standard deviations shown parentheses. The mean gap values of CvxBndL2 and CvxBndL1, Gap(Cvx2) and Gap(Cvx1) respectively, are given in the last two rows.

timality gaps for the approximate CvxBnd procedures. The gaps were quite small in most cases (the gaps were very close to zero in the synthetic case, and so are not shown), demonstrating the tightness of the proposed approximation scheme.

5.7 Conclusion

We first out pointed out that there is a dilemma between ensuring robustness and tractability, by proving that a bounded estimator cannot be convex while convex losses cannot be robust. In response, we developed a new robust regression method that can guarantee a form of robustness (bounded response) while ensuring tractability (polynomial run-time). To do so, we first designed an adaptive form of M-estimation and applied a convex relaxation to this form. The estimator was proved consistent under some restrictive but non-trivial conditions, although we have not established general consistency. We also gave results that established the robustness of the estimator, and developed an efficient optimization algorithm on the relaxation. The empirical evaluation revealed that the method meets or surpasses the generalization ability of state-of-the-art robust regression methods in experimental studies. Although the method is more computationally involved than standard approaches, it achieves reasonable scalability in real problems.

Chapter 6

Conclusion and Future Work

I summarize the main contributions of the previous chapters and discuss relevant future work for each.

In Chapter 3, I presented a reformulation of a non-convex two-layer model as a joint optimization problem, by explicitly defining the latent layers with free variables and relaxing the strict nonlinear transformation in consecutive layers using losses that are convex in first argument. This relaxed reformulation leads to an appealing strategy that decouples layers. We showed that for some particular losses, one can obtain an equivalent reformulation that has an objective function that is jointly convex in all variables. However, the constraint set of the equivalent reformulation is non-convex, we therefore relaxed the non-convex constraints. Although the final problem is a fully convex formulation for a conditional latent model with one hidden layer, it yields an expensive optimization problem that prevent off-the-shelf solvers from achieving satisfactory scaling properties. We then designed an ADMM-based algorithm to separate the constraints, and also adopted a hybrid conditional gradient algorithm with a low-rank strategy to handle the SDP constraint. The low-rank approach improved space complexity by incrementally adding bases and eliminating storage of the full SDP matrix, which can also be significantly accelerated by using local updates. We observed that this boosting style algorithm

converges in a few iterations and is stable in practice. To demonstrate that the relaxation can still capture the latent structure in original model, we first provided an empirical sanity check which shows that the relaxation can separate classes that one layer models cannot. The proposed model also outperforms baseline one layer models and a local alternating version on some real datasets. The misclassification results indicate that relaxation can still capture the predictive latent structure. Important directions for future work include replacing the step and indmax transfers with more traditional sigmoid and softmax transfers, while also replacing the margin losses with more traditional Bregman divergences; refining the relaxation to allow more control over the structure of the latent representations and extending the two-layer model to a model with multiple number of hidden layers which we studied in Chapter 4. Another direction is proving the tightness of the relaxation bounds by analyzing the gap between original problem and relaxation.

In Chapter 4, I presented a significant extension of the approach in Chapter 3. It is well-known that the depth is one of the key components of the current state of the art models in deep learning. Consequently, even though achieving a convex formulation of conditional latent modeling with one hidden layer is an exciting result, extending the convex approach to more than one hidden layer is an essential step. I first showed in Chapter 4 that this extension is not a trivial task. Then, using the similar decoupling of layers strategy, we developed a new output kernel type and a novel loss for connecting hidden layers, which yielded a jointly convex objective. We then relaxed the nonconvex constraints. To cope with the resulting optimization problem, we then developed a novel algorithm that eliminates the nested optimization bottleneck that was present in the two-layer model. As in Chapter 3, we conducted some sanity check experiments on artificial data to show that the relaxation preserves much of the important structure of the original model. We also report promising results on real data for which we compare the three-layer model to a local version and two-layer models. One essential direction for future

work is to improve the current efficiency of the algorithm, so that it can scale to the sizes of current large datasets. Aside from further improvements in algorithmic efficiency, an interesting direction for future investigation is to also capture unsupervised stage-wise training principles via auxiliary autoencoder objectives within a convex framework, rather than treating input reconstruction as a mere heuristic training device.

In Chapter 5, I presented a study of the nonconvexity of bounded loss functions in M-estimation, proposing a tractable robust estimation scheme with certain robustness, consistency and tractability guarantees. I first posed the dilemma that bounded losses cannot be convex while convex losses cannot be robust. We then proposed an adaptive form for robust estimation, which was then reformulated to achieve a novel convex relaxation. We showed certain robustness and consistency properties of the approximation. We then developed an efficient learning strategy, and empirically validated the effectiveness of the relaxed estimation scheme on artificial and real datasets. One direction for future work would be to investigate whether the proposed estimator achieves stronger robustness properties, such as high breakdown or bounded influence. It would also be interesting to extend the approach to estimate scale in a robust and tractable manner. Finally, investigating whether other techniques from the robust statistics and machine learning literatures can be incorporated in the general framework, while preserving desired properties, would also be interesting.

References

(1996). Delve dataset.

URL `http://www.cs.toronto.edu/~delve/data/datasets.html`

(2010). Cmu statlib datasets.

URL `http://lib.stat.cmu.edu/datasets`

Andersen, E. D., & Andersen, K. D. (2000). The mosek interior point optimizer for linear programming: an implementation of the homogeneous algorithm. *High Performance Optimization*, (pp. 197–232).

Argyriou, A., Evgeniou, T., & Pontil, M. (2008). Convex multi-task feature learning. *Machine Learning*, *73*.

Arora, S., Bhaskara, A., Ge, R., & Ma, T. (2014). Bounds for learning deep representations. In *Proceedings of the International Conference on Machine Learning*.

Aslan, Ö., Cheng, H., Zhang, X., & Schuurmans, D. (2013). Convex two-layer modeling. In *Advances in Neural Information Processing Systems*.

Aslan, Ö., Zhang, X., & Schuurmans, D. (2014). Convex deep learning via normalized kernels. In *Advances in Neural Information Processing Systems*.

Baldi, P., & Hornik, K. (1989). Neural networks and principal component analysis: Learning from examples without local minima. *Neural networks*, *2(1)*, 5358.

Banerjee, A., Merugu, S., Dhillon, I., & Ghosh, J. (2005). Clustering with Bregman divergences. *Journal of Machine Learning Research*, *6*, 1705–1749.

Barron, A. R. (1993). Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, *39(3)*, 930945.

Bartlett, P. L., & Mendelson, S. (2003). Rademacher and gaussian complexities: risk bounds and structural results. *Journal of Machine Learning Research*, *3*, 463–482.

Ben-David, S., Eiron, N., & Long, P. M. (2003). On the difficulty of approximately maximizing agreements. *Journal of Computer and System Sciences*, *66*(3), 496 – 514.
URL http://www.sciencedirect.com/science/article/pii/S0022000003000382

Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, *2*, 1–127.

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(2), 183–238.

Bengio, Y., Le Roux, N., Vincent, P., & Delalleau, O. (2005). Convex neural networks. In *Advances in Neural Information Processing Systems*.

Bennett, K. P., & Hernandez, E. P. (2006). The interplay of optimization and machine learning research. *Journal of Machine Learning Research*, *7*, 12651281.

Bernholt, T. (2005). Robust estimators are hard to compute. Tech. Rep. 52/2005, SFB475, U. Dortmund.

Bertsekas, D. (1995). *Nonlinear Programming*. Athena.

Bie, T. D., & Cristianini, N. (2003). Convex methods for transduction. In *Advances in Neural Information Processing Systems*.

Black, M., & Rangarajan, A. (1996). On the unification of line processes, outlier rejection, and robust statistics with applications in early vision. *International Journal of Computer Vision*, *19*(1), 57–91.

Blum, A. L., & Rivest, R. L. (1992). Training a 3-node neural network is np-complete. *Neural Networks*, *5(1)*, 117127.

Bottou, L., Curtis, F. E., & Nocedal, J. (2016). Optimization methods for large scale machine learning. https://arxiv.org/pdf/1606.04838v1.pdf.

Boyd, S., Parikh, N., Chu, E., Peleato, B., & Eckstein, J. (2010). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, *3*(1), 1–123.

Boyd, S., & Vandenberghe, L. (2004). *Convex Optimization*. Cambridge U. Press.

Bradley, D., & Bagnell, J. (2009). Convex coding. In *Conference on Uncertainty in Artificial Intelligence*.

Bray, A. J., & Dean, D. S. (2007). Statistics of critical points of gaussian fields on large-dimensional spaces. *Physics Review Letter*, *98*.

Burer, S., & Monteiro, R. (2003). A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical Programming*, *95*(2), 329–357.

Candes, E., Li, X., Ma, Y., & Wright, J. (2009). Robust principal component analysis? ArXiv:0912.3599.

Carreira-Perpiñán, M., & Lu, Z. (2010). dimensionality reduction by unsupervised regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Chapelle, O. (2007). Training a support vector machine in the primal. *Neural Computation*, *19*(5), 1155–1178.

Chapelle, O., Schölkopf, B., & Zien, A. (Eds.) (2006). *Semi-Supervised Learning*. MIT Press.

Cheng, H., Zhang, X., & Schuurmans, D. (2013). Convex relaxations of Bregman divergence clustering. In *Conference on Uncertainty in Artificial Intelligence*.

Cho, Y., & Saul, L. (2010). Large margin classification in infinite neural networks. *Neural Computation.*, *22*.

Choromanska, A., Henaff, M., Mathieu, M., Arous, G. B., & LeCun, Y. (2015a). The loss surfaces of multilayer networks. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*.

Choromanska, A., LeCun, Y., & Arous, G. B. (2015b). Open problem: The landscape of the loss surfaces of multilayer networks. In *Proceedings of the Conference on Learning Theory*.

Christmann, A., & Steinwart, I. (2007). Consistency and robustness of kernel-based regression in convex risk minimization. *Bernoulli*, *13*(3), 799–819.

Christmann, A., Van Messem, A., & Steinwart, I. (2009). On consistency and robustness properties of support vector machines for heavy-tailed distributions. *Statistics and Its Interface*, *2*, 311–327.

Crammer, K., & Singer, Y. (2001). On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, (pp. 265–292).

Cucker, F., & Zhou, D. (2007). *Learning Theory: An Approximation Theory Viewpoint*. Cambridge.

Cun, Y. L., Boser, B., Denker, J. S., Howard, R. E., Habbard, W., Jackel, L. D., & Henderson, D. (1990). Advances in neural information processing systems 2. chap. Handwritten Digit Recognition with a Back-propagation Network, (pp. 396–404).
URL `http://dl.acm.org/citation.cfm?id=109230.109279`

Dahl, G., Yu, D., Deng, L., & Acero, A. (2012). On the problem of local minima in backpropagation. *IEEE Trans. ASLP*, *20*(1), 30–42.

d'Aspremont, A., & Boyd, S. (2003). Relaxations and randomized methods for nonconvex qcqps. *EE392o*.
URL `http://archive.ics.uci.edu/ml`

d'Aspremont, A., & Karoui, N. E. (2014). A stochastic smoothing algorithm for semidefinite programming. *SIAM Journal on Optimization*, *24(3)*, 1138–1177.

Dauphin, Y., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., & Bengio, Y. (2014). Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in Neural Information Processing Systems*.

Davies, P., & Gather, U. (2007). The breakdown point—examples and counterexamples. *REVSTAT Statistical Journal*, *5*(1), 1–17.

Davis, G., Mallat, S., & Avellaneda, M. (1997). Greedy adaptive approximation. *Journal of Constructive Approximation*, *13*, 57–98.

Dinuzzo, F., Ong, C. S., Gehler, P., & Pillonetto, G. (2011). Learning output kernels with block coordinate descent. In *Proceedings of the International Conference on Machine Learning*.

Donoho, D., & Huber, P. (1983). The notion of breakdown point. In *A Festschrift for Erich L. Lehmann*, (pp. 157–184). Wadsworth.

El Ghaoui, L., & Lebret, H. (1997). Robust solutions to least-squares problems with uncertain data. *SIAM Journal on Matrix Analysis and Applications*, *18*, 1035–1064.

Elad, M., & Aharon, M. (2006). Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Trans. on Image Processing*, *15*, 3736–3745.

Erhan, D., Bengio, Y., Courville, A., Manzagol, P., & Vincent, P. (2010). Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, *11*, 625–660.

Feldman, V., Guruswami, V., Raghavendra, P., & Wu, Y. (2009). Agnostic learning of monomials by halfspaces is hard. In *Proceedings of the 2009 50th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '09, (pp. 385–394). Washington, DC, USA: IEEE Computer Society.
URL `http://dx.doi.org/10.1109/FOCS.2009.26`

Feng, J., Xu, H., & Yan, S. (2013). Online robust pca via stochastic optimization. In *Advances in Neural Information Processing Systems*.

Frank, M., & Wolfe, P. (1956). An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, *3*, 149–154.

Fuernkranz, J., Huellermeier, E., Mencia, E., & Brinker, K. (2008). Multilabel classification via calibrated label ranking. *Machine Learning*, *73*(2), 133–153.

Fyodorov, Y. V., & Williams, I. (2007). Replica symmetry breaking condition exposed by random matrix calculation of landscape complexity. *Journal of Statistical Physics*, *129 (5-6)*, 10811116.

Garber, D., & Hazan, E. (2011). Approximating semidefinite programs in sublinear time. In *Advances in Neural Information Processing Systems*.

Garber, D., & Hazan, E. (2014). Faster rates for the frank-wolfe method over strongly-convex sets. Tech. rep. Http://arxiv.org/abs/1406.1305.

Garey, M., & Johnson, D. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company.

Gens, R., & Domingos, P. (2012). Discriminative learning of sum-product networks. In *Advances in Neural Information Processing Systems*.

Goemans, M. X., & Williamson, D. (1995). Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, *42*, 1115–1145.

Goldberg, A., Zhu, X., Recht, B., Xu, J., & Nowak, R. (2010). Transduction with matrix completion: Three birds with one stone. In *Advances in Neural Information Processing Systems*.

Goodfellow, I. J., Vinyals, O., & Saxe, A. M. (2015). Qualitatively characterizing neural network optimization problems. In *Proceedings of International Conference on Learning Representations*.

Gori, M., & Tesi, A. (1992). On the problem of local minima in backpropagation. *IEEE PAMI*, *14*, 76–86.

Grant, M., & Boyd, S. (2008). Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, & H. Kimura (Eds.) *Recent Advances in Learning and Control*, Lecture Notes in Control and Information Sciences, (pp. 95–110). Springer-Verlag Limited. `http://stanford.edu/~boyd/graph_dcp.html`.

Grant, M., & Boyd, S. (2014). CVX: Matlab software for disciplined convex programming, version 2.1. `http://cvxr.com/cvx`.

Grippo, L., Palagi, L., Piacentini, M., Piccialli, V., & Rinaldi, G. (2012). Speedp: an algorithm to compute sdp bounds for very large max-cut instances. *Mathematical Programming*, *136*, 353–373.

Grippo, L., Palagi, L., & Piccialli, V. (2008). Necessary and sufficient global optimality conditions for nlp reformulations of linear sdp problems. *Journal of Global Optimization*.

Grippoa, L., & Sciandrone, M. (2000). On the convergence of the block nonlinear Gauss-Seidel method under convex constraints. *Operations Research Letters*, *26*, 127–136.

Guo, Y., & Schuurmans, D. (2007). Convex relaxations of latent variable training. In *Advances in Neural Information Processing Systems*.

Guo, Y., & Schuurmans, D. (2011). Adaptive large margin training for multilabel classification. In *Proceedings of theAssociation for the Advancement of Artificial Intelligence conference on Artificial Intelligence*.

Hajnal, A. (1993). Threshold circuits of bounded depth. *Journal of Computer and System Sciences*, *46*(2), 129–154.

Hampel, F., Ronchetti, E., Rousseeuw, P., & Stahel, W. (1986). *Robust Statistics: The Approach Based on Influence Functions*. Wiley.

Hazan, E. (2008). Sparse approximate solutions to semidefinite programs. In *Latin American Conference on Theoretical Informatics*.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Helmberg, C., & Rendl, F. (2000). A spectral bundle method for semidefinite programming. *SIAM Journal on Optimization*, *10*, 673–696.

Hinton, G. (2007). Learning multiple layers of representations. *Trends in Cognitive Sciences*, *11*, 428–434.

Hinton, G., Osindero, S., & Teh, Y. (2006). A fast algorithm for deep belief nets. *Neur. Comp.*, *18*(7).

Hinton, G., Srivastava, N., Krizhevsky, A., Sutskever, A., & Salakhutdinov, R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. ArXiv:1207.0580.

Hoeffgen, K., Simon, H., & Van Horn, K. (1995). Robust trainability of single neurons. *JCSS*, *52*, 114–125.

Horn, R., & Johnson, C. (1985). *Matrix Analysis*. Cambridge.

Håstad, J. (2001). Some optimal inapproximability results. *J. ACM*, *48*(4), 798–859.

Hsieh, C.-J., Sustik, M. A., Dhillon, I. S., & Ravikumar, P. (2013). Big & quic: Sparse inverse covariance estimation for a million variables. In *Advances in Neural Information Processing Systems*.

Huber, P., & Ronchetti, E. (2009). *Robust Statistics*. Wiley, 2nd ed.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the International Conference on Machine Learning*.

Jaggi, M. (2013). Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *Proceedings of the International Conference on Machine Learning*.

Jain, P., Netrapalli, P., & Sanghavi, S. (2013). Low-rank matrix completion using alternating minimization. In *Proceedings of the ACM symposium on Theory of computing*.

Janzamin, M., Sedghi, H., & Anandkumar, A. (2015). Beating the perils of nonconvexity: Guaranteed training of neural networks using tensor methods. http://arxiv.org/abs/1506.08473.

Joachims, T. (1999). Transductive inference for text classification using support vector machines. In *Proceedings of the International Conference on Machine Learning*.

Joulin, A., & Bach, F. (2012). A convex relaxation for weakly supervised classifiers. In *Proceedings of the International Conference on Machine Learning*.

Joulin, A., Bach, F., & Ponce, J. (2010). Efficient optimization for discriminative latent class models. In *Advances in Neural Information Processing Systems*.

Joulin, A., Bach, F., & Ponce, J. (2012). Multi-class cosegmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Journée, M., Bach, F., Absil, P.-A., & Sepulchre, R. (2010). Low-rank optimization on the cone of positive semidefinite matrices. *SIAM Journal on Optimization*, *20*, 2327–2351.

Kawaguchi, K. (2016). Deep learning without poor local minima. In *Advances in Neural Information Processing Systems*.

Kimeldorf, G., & Wahba, G. (1970). A correspondence between Bayesian estimation on stochastic processes and smoothing by splines. *Annals of Mathematical Statistics*, *41*(2), 495–502.

Kimeldorf, G., & Wahba, G. (1971). Some results on Tchebycheffian spline functions. *Journal of Mathematical Analysis and Applications*, *33*, 82–95.

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. URL `http://www.cs.toronto.edu/~kriz/cifar.html`

Krizhevsky, A., Sutskever, A., & Hinton, G. (2012). ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*.

Kulis, B., Surendran, A. C., & Platt, J. C. (2007). Fast low-rank semidefinite programming for embedding and clustering. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*.

Lanckriet, G., Cristianini, N., Bartlett, P., Ghaoui, L., & Jordan, M. (2004). Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*.

Lanckriet, G., Cristianini, N., Bartlett, P., Ghaoui, L., & Jordan, M. (2007). A direct formulation for sparse pca using semidefinite programming. *SIAM review*, *49(3)*, 434–448.

Laue, S. (2012). A hybrid algorithm for convex semidefinite optimization. In *Proceedings of the International Conference on Machine Learning*.

Lawrence, N. (2005). Probabilistic non-linear principal component analysis. *JMLR*, *6*, 1783–1816.

Le, Q., Ranzato, M., Monga, R., Devin, M., Corrado, G., Chen, K., Dean, J., & Ng, A. (2012). Building high-level features using large scale unsupervised learning. In *Proceedings of the International Conference on Machine Learning*.

LeCun, Y. (2007). Who is afraid of non-convex loss functions? Http://videolectures.net/eml07_lecun_wia.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep Learning. *Nature*, *521*, 436444.

LeCun, Y., & Cortes, C. (2010). MNIST h. digit database.
URL `http://yann.lecun.com/exdb/mnist/`

Lee, H., Grosse, R., Ranganath, R., & Ng, A. Y. (2011). Unsupervised learning of hierarchical representations with convolutional deep belief networks. *Communications of the ACM*, *54*(10), 95–103.
URL `http://doi.acm.org/10.1145/2001269.2001295`

Lemaréchal, C., & Oustry, F. (1999). Semidefinite relaxations and lagrangian duality with application to combinatorial optimization. *INRIA, Rapport de recherche*.

Lichman, M. (2013). UCI machine learning repository.
URL `http://archive.ics.uci.edu/ml`

Livni, R., Shalev-Shwartz, S., & Shamir, O. (2014a). An algorithm for training polynomial networks. ArXiv:1304.7045v2.

Livni, R., Shalev-Shwartz, S., & Shamir, O. (2014b). On the computational efficiency of training neural networks. In *Advances in Neural Information Processing Systems*.

Lovász, L. (1979). On the shannon capacity of a graph. *IEEE Transactions on Information Theory*, *25*(1), 1–7.

Lovász, L., & Schrijver, A. (1991). Cones of matrices and set-functions and 0-1 optimization. *SIAM J. Optim*, *1*, 166–190.

Luo, Z., Ma, W., So, A., Ye, Y., & Zhang, S. (2010). Semidefinite relaxation of quadratic optimization problems. *IEEE Signal Processing Magazine*, *27*, 20–34.

Maronna, R., Martin, R., & Yohai, V. (2006). *Robust Statistics: Theory and Methods*. Wiley.

Martens, J. (2016). Second-order optimization for neural networks. Ph.D. Thesis, Dept. of Computer Science, University of Toronto.

McCoy, M., & Tropp, J. (2011). Two proposals for robust pca using semidefinite programming. *Electronic Journal of Statistics*, *5*, 1123–1160.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., &

Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533.
URL http://dx.doi.org/10.1038/nature14236

Montufar, G., Pascanu, R., Cho, K., & Bengio, Y. (2014). On the number of linear regions of deep neural networks. In *Advances in Neural Information Processing Systems*.

Mukherjee, S., Niyogi, P., Poggio, T., & Rifkin, R. (2006). Learning theory: Stability is sufficient for generalization and necessary and sufficient for consistency of empirical risk minimization. *Advances in Computational Mathematics*, *25*(1-3), 161–193.

Murty, K. G., & Kabadi, S. N. (1987). Some np-complete problems in quadratic and nonlinear programming. *Mathematical programming*, *39(2)*, 117129.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the International Conference on Machine Learning*.

Natarajan, B. (1995). Sparse approximate solutions to linear systems. *Society for Industrial and Applied Mathematics (SIAM) Journal on Computing*, *13*, 227–234.

Neal, R. (1992). Connectionist learning of belief networks. *Artificial Intelligence*, *56*(1), 71–113.

Nemirovski, A. (2007). Prox-method with rate of convergence o(1/t) for variational inequalities with lipschitz continuous monotone operators and smooth convex-concave saddle point problems. *SIAM Journal on Optimization*, *15*, 229–251.

Nesterov, Y. (1983). A method of solving a convex programming problem with convergence rate o(1/sqr(k)). *Soviet Mathematics Doklady*, *27*, 372–376.

Nesterov, Y. (2003). *Introductory Lectures on Convex Optimization: A Basic Course*. Kluwer.

Nesterov, Y. (2007). Smoothing technique and its applications in semidefinite optimization. *Mathematical Programming*, *110(2)*, 245–259.

Nesterov, Y., & Nimirovskii, A. (1994). *Interior-Point Polynomial Algorithms in Convex Programming*.

Nesterov, Y., Wolkowicz, H., & Ye, Y. (2000). Semidefinite programming relaxations of nonconvex quadratic optimization. In *Handbook of semidefinite programming*, vol. 27. Kluwer Acad. Publ.

Netrapalli, P., Jain, P., & Sanghavi, S. (2013). Phase retrieval using alternating minimization. In *Advances in Neural Information Processing Systems*.

Netrapalli, P., Niranjan, U., Sanghavi, S., Anandkumar, A., & Jain, P. (2014). Non-convex robust pca. In *Advances in Neural Information Processing Systems*.

Nowozin, S., & Bakir, G. (2008). A decoupled approach to exemplar-based unsupervised learning. In *Proceedings of the International Conference on Machine Learning*.

Nunkesser, R., & Morell, O. (2010). An evolutionary algorithm for robust regression. *Computational Statistics and Data Analysis*, *54*, 3242–3248.

Overton, M., & Womersley, R. (1993). Optimality conditions and duality theory for minimizing sums of the largest eigenvalues of symmetric matrices. *Mathematical Programming*, *62*, 321–357.

Peng, J., & Wei, Y. (2007). Approximating k-means-type clustering via semidefinite programming. *Society for Industrial and Applied Mathematics(SIAM) Journal on Optimization*, *18*, 186–205.

Primak, M., & Kheyfets, B. (1995). A modification of the inscribed ellipsoid method. *Mathematical and Computer Modelling*, *21*(11), 69–76.

Razborov, A. A. (1992). On small depth threshold circuits. In *Proceedings of the Third Scandinavian Workshop on Algorithm Theory*.

Renegar, J. (2014). Efficient first order methods for linear programming and semidefinite programming. *Optimization-online.org*.

Rifai, S., Vincent, P., Muller, X., Glorot, X., & Bengio, Y. (2011). Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the International Conference on Machine Learning*.

Rifkin, R., & Lippert, R. (2007). Value regularization and Fenchel duality. *Journal of Machine Learning Research*, *8*, 441–479.

Robert, V., & Hande Yurttan, B. (2000). On formulating semidefinite programming problems as smooth convex nonlinear optimization problems. Tech. rep.

Rockafellar, R. T. (1970). *Convex Analysis*, vol. 28 of *Princeton Mathematics Series*. Princeton, NJ: Princeton University Press.

Rockafellar, R. T. (1993). Lagrange multipliers and optimality. *SIAM Review*, *35*(2), 183–238.
URL `http://www.jstor.org/stable/2133143`

Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., & Bengio, Y. (2015). Fitnets: Hints for thin deep nets. In *In Proceedings of ICLR*.

Rousseeuw, P., & Leroy, A. (1987a). *Robust Regression and Outlier Detection*. Wiley.

Rousseeuw, P., & Van Driessen, K. (2006). Computing LTS regression for large data sets. *Data Mining and Knowledge Discovery*, *12*(1), 29–45.

Rousseeuw, P. J., & Leroy, A. M. (1987b). *Robust Regression and Outlier Detection*. New York: John Wiley & Sons.

Sanjeev, A., Hazan, E., & Kale, S. (2005). Fast algorithms for approximate semidefinite programming using the multiplicative weights update method. In *Proceedings of the IEEE Annual Symposium on Foundations of Computer Science*.

Saxe, A., McClelland, J. L., & Ganguli, S. (2014). Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *Proceedings of International Conference on Learning Representations*.

Schoelkopf, B., & Smola, A. (2002). *Learning with Kernels*. MIT Press.

Sedghi, H., & Anandkumar, A. (2014). Provable methods for training neural networks with sparse connectivity. https://arxiv.org/abs/1412.2693.

Shalev-Shwartz, S., Shamir, O., Srebro, N., & Sridharan, K. (2010). Learnability, stability and uniform convergence. *J. Mach. Learn. Res.*, *11*, 2635–2670.
URL `http://dl.acm.org/citation.cfm?id=1756006.1953019`

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., & Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, *529*(7587), 484–489.
URL `http://dx.doi.org/10.1038/nature16961`

Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *Proceedings of International Conference on Learning Representations*.

Sindhwani, V., & Keerthi, S. (2006). Large scale semi-supervised linear SVMs.

In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Socher, R., Lin, C., Ng, A., & Manning, C. (2011). Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the International Conference on Machine Learning*.

Sra, S., Nowozin, S., & Wright, S. J. (2012). *Optimization for Machine Learning*. MIT Press.

Srivastava, N., & Salakhutdinov, R. (2012). Multimodal learning with deep Boltzmann machines. In *Advances in Neural Information Processing Systems*.

Steinwart, I., & Christmann, A. (2008). *Support Vector Machines*. Springer.

Strang, G. (1996). *Introduction to Linear Algebra*. Wellesley-Cambridge Press.

Sturm, J. F. (1999). Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones. *Optimization Methods and Software*, *11*(1-4), 625–653. URL http://www.tandfonline.com/doi/abs/10.1080/10556789908805766

Sutskever, I., Martens, J., Dahl, G., & Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In *Proceedings of the International Conference on Machine Learning*.

Swersky, K., Ranzato, M., Buchman, D., Marlin, B., & de Freitas, N. (2011). On autoencoders and score matching for energy based models. In *Proceedings of the International Conference on Machine Learning*.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Tishby, N., Pereira, F., & Bialek, W. (1999). The information bottleneck method. In *Proceedings of the Annual Allerton Conference on Communication, Control and Computing*.

Toh, K., Todd, M., & Tutuncu, R. (1999). Sdpt3 — a matlab software package for semidefinite programming. *Optimization Methods and Software*, *11*, 545–581. URL `http://www.math.nus.edu.sg/~mattohkc/sdpt3.html`

van der Vaart, A. W., & Wellner, J. A. (1996). *Weak Convergence and Empirical Processes*. Springer.

Vandenberghe, L., & Boyd, S. (1996). Semidefinite programming. *SIAM Review*, (pp. 49–95).

Vapnik, V. N. (1998). *Statistical Learning Theory*. Wiley-Interscience.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., & Manzagol, P.-A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, *11*(3), 3371–3408.

Wen, Z., Goldfarb, D., & K.Scheinberg (2012). Block coordinate descent methods for semidefinite programming. *Handbook on Semidefinite, Conic and Polynomial Optimization*, (pp. 533–564).

Wen, Z., Goldfarb, D., & Yin, W. (2010). Alternating direction augmented lagrangian methods for semidefinite programming. *Mathematical Programming Computation*, *2*, 20–230.

Xu, H., Caramanis, C., & Mannor, S. (2008). Robust regression and Lasso. In *Advances in Neural Information Processing Systems*.

Xu, H., Caramanis, C., & Mannor, S. (2009). Robustness and regularization of support vector machines. *Journal of Machine Learning Research*, *10*, 1485–1510.

Xu, L., Neufeld, J., Larson, B., & Schuurmans, D. (2004). Maximum margin clustering. In *Advances in Neural Information Processing Systems*.

Xu, L., & Schuurmans, D. (2005). Unsupervised and semi-supervised multi-class support vector machines. In *National Conference on Artificial Intelligence*.

Yu, Y., Aslan, O., & Schuurmans, D. (2012). A polynomial-time form of robust regression. In *Advances in Neural Information Processing Systems*.

Yu, Y., Yang, M., Xu, L., White, M., & Schuurmans, D. (2010). Relaxed clipping: A global training method for robust regression and classification. In *Advances in Neural Information Processing Systems*.

Zhang, X., Yu, Y., & Schuurmans, D. (2012). Accelerated training for matrix-norm regularization: A boosting approach. In *Advances in Neural Information Processing Systems*.

Zhang, X., Yu, Y., White, M., Huang, R., & Schuurmans, D. (2011). Convex sparse coding, subspace learning, and semi-supervised extensions. In *Annual Conference on Artificial Intelligence*.

Zhang, Y., Lee, J. D., & Jordan, M. I. (2015a). ℓ_1-regularized neural networks are improperly learnable in polynomial time. CoRR abs/1510.03528.

Zhang, Y., Martin, J. D. L., Wainwright, J., & Jordan, M. I. (2015b). Learning halfspaces and neural networks with random initialization. https://arxiv.org/pdf/1511.07948v1.pdf.

Zhang, Z. (1997). Parameter estimation techniques: A tutorial with application to conic fitting. *Image and Vision Computing Journal*, *15*, 59–76.

Zhuang, J., Tsang, I., & Hoi, S. (2011). Two-layer multiple kernel learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*.

Zou, H., Hastie, T., & Tibshirani, R. (2006). Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, *15*, 265–286.

Appendix A

Supplementary Material

A.1 Details for Training a Two-layer Model

A.1.1 Faster Gradient Calculation by Low Rank Decomposition

Note that the gradient $\nabla \mathcal{G}_1(N)$ can be readily computed given C^*. Applying the same technique to the second layer yields an efficient procedure for evaluating $\mathcal{G}(N)$ and $\nabla \mathcal{G}(N)$. Finally note that many of the matrix-vector multiplications in this procedure can be further accelerated by exploiting the low rank factorization of N maintained by the boosting algorithm.

Recall that $\mathcal{G}$ consists mainly of two parts: the first layer objective in (3.11) and the second layer objective in (3.7). We just show how to solve the first layer, while the technique can be applied directly to the second layer too. For simplicity, we only consider linear kernel on X, and extension to nonlinear kernel is also straightforward.

For convenience we copy (3.11) to here, using N_T and H_T:

$$\min_W L_1(WX, H_T) + \tfrac{\alpha}{2}\|W\|_F^2 = \min_W \tilde{L}_1(H_T'WX, H_T'H_T) + \tfrac{\alpha}{2}\|W\|_F^2 \tag{A.1}$$

$$= \min_C \tilde{L}_1(H_T'H_TCX'X, H_T'H_T) + \tfrac{\beta}{2}\operatorname{tr}(XC'H_T'H_TCX') \tag{A.2}$$

$$= \min_{D \in \mathrm{Im}(N_T)} \tilde{L}_1(DK, N_T) + \tfrac{\alpha}{2}\operatorname{tr}(D'N_T^\dagger DK). \tag{A.3}$$

Denote this objective as $\mathcal{G}_1(N)$ or $\mathcal{H}_1(H)$. The boosting Step 4 indeed only requires the gradient $\nabla\mathcal{H}_1(H_T)$, while the gradient $\nabla\mathcal{G}_1(N_T)$ is needed only in Step 3. So we focus on the efficient computation of these two gradients.

To compute $\nabla\mathcal{H}_1(H_T)$, it suffices to optimize over W in (A.1). This is advantageous because a) the objective is strongly convex which is in favor of LBFGS; b) the size of W is nT, where T is the iteration index of boosting and is often quite small; c) the gradient in W can be computed in $O(tnT)$ time, which can also benefit from the low value of T.

To compute $\nabla\mathcal{G}_1(N_T)$, one possible approach is to solve for C in (A.2):

$$\min_C \tilde{L}_1(N_TCX'X, N_T) + \tfrac{\beta}{2}\operatorname{tr}(XC'N_TCX'). \tag{A.4}$$

However, the cost for computing the gradient in C is $O(t^2n)$, which is expensive if done at each iteration of optimization. Therefore we introduce one more change of variable: $E = CX'$ and then the problem becomes

$$\min_E \tilde{L}_1(N_TEX, N_T) + \tfrac{\beta}{2}\operatorname{tr}(E'N_TE). \tag{A.5}$$

So our final strategy is:

- Find the optimal W^* for (A.1) as in computing $\nabla\mathcal{H}_1(H_T)$,
- Recover the optimal E^* for (A.5) by finding any E that satisfies $W^* = H_TE$,
- Use E^* to compute the gradient $\nabla\mathcal{G}_1(N_T)$ via (A.5).

The first and second steps can make use of the low value of T as in computing $\nabla\mathcal{H}_1(H_T)$. The last step does cost $O(t^2n)$, but it needs to be done only once, rather than in each iteration of solving for C in (A.4). So in summary, the total computational cost is $O(tnT)$ per iteration in optimizing W, followed by $O(t^2n)$ for one time to recover E^* and to compute $\nabla\mathcal{G}_1(N_T)$ via (A.5).

A.1.2 Experimental Details for Two-layer Model

For the "real" experiments we used a collection of binary labeled data sets: USPS (241×1500) and G241N (241×1500) from Chapelle et al. (2006), Letter (vowel letters A-E vs non vowel letters B-F 16×3098) from (Lichman, 2013), MNIST ($\{1, 9\}$vs$\{4, 8\}$: 784×28484), and CIFAR-100 (bicycle and motorcycle vs lawnmower and tank 256×1526 where red channel features are preprocessed by averaging pixels) from (LeCun & Cortes, 2010) and (Krizhevsky, 2009).

The weight, latent matrix and response matrix for "Xor' dataset are given in Figure A.1, Figure A.2 and Figure A.3.

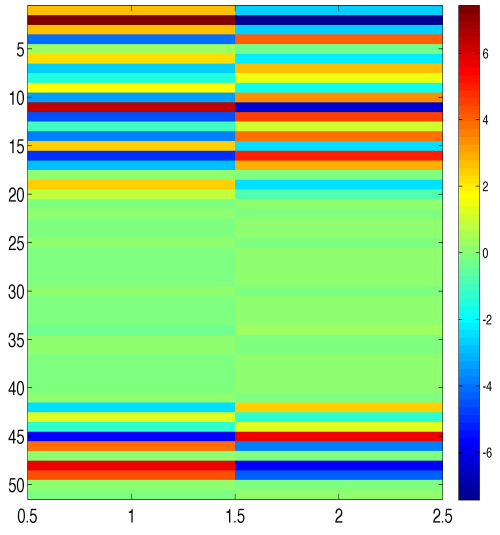


Figure A.1: Weight matrix V for second layer

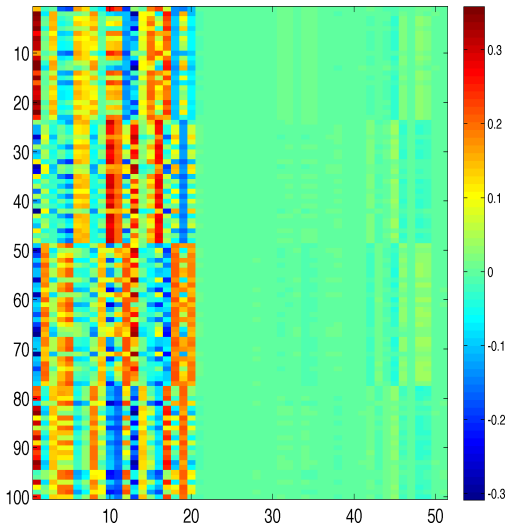


Figure A.2: The latent matrix H

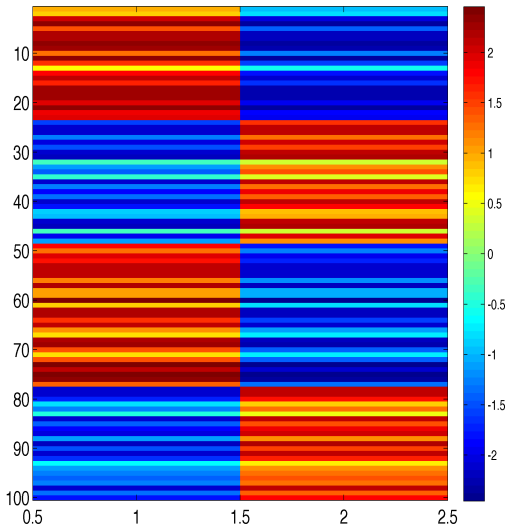


Figure A.3: Response matrix VH for second layer

A.2 Details for Training a Multi-layer Model

A.2.1 Proof that the Feasible Region of (4.12) is Convex

Here we prove that the set

$$V := \{(M, S, K) : M \succeq \mathbf{0}, K \succeq \mathbf{0}, S \in M\mathbb{R}K\} \tag{A.6}$$

is convex.

Recall that $\mathbb{R}\Phi$ consists of matrices in the row span of Φ. Since M and K are PSD, the constraint $S \in M\mathbb{R}K$ involves the intersection of two convex constraints. Let

$$V_1 = \{(M, S, K) : M \succeq \mathbf{0}, K \succeq \mathbf{0}, S \in M\mathbb{R}\} \tag{A.7}$$

$$V_2 = \{(M, S, K) : M \succeq \mathbf{0}, K \succeq \mathbf{0}, S \in \mathbb{R}K\}. \tag{A.8}$$

First, we show that the set V_1 is convex. Suppose $S_1 \in M_1\mathbb{R}$ and $S_2 \in M_2\mathbb{R}$ where $M_1 \succeq \mathbf{0}$ and $M_2 \succeq \mathbf{0}$. To prove that $S_1 + S_2 \in (M_1 + M_2)\mathbb{R}$, it suffices to show that $M_1\mathbb{R} \cup M_2\mathbb{R} \subseteq (M_1 + M_2)\mathbb{R}$. To this end, consider its contrapositive, *i.e.*, there exists $\mathbf{r}_1$, $\mathbf{r}_2$, $\mathbf{x}$ such that $\mathbf{x}'(M_1\mathbf{r}_1 + M_2\mathbf{r}_2) \neq 0$ while $\mathbf{x}'(M_1 + M_2) = \mathbf{0}$. However this is impossible, because the latter implies $\mathbf{x}'M_1 = \mathbf{x}'M_2 = \mathbf{0}'$ when $M_1 \succeq \mathbf{0}$ and $M_2 \succeq \mathbf{0}$.

V_2 is convex for isomorphic reasons. It remains only to show $V = V_1 \cap V_2$, since an intersection of convex sets is convex. The $\subseteq$ relationship is straightforward. To show $\supseteq$, let $S = MP = QK$ for some P and Q; then note that $SK^\dagger K = S = MM^\dagger S$. Hence $MM^\dagger SK^\dagger K = S$, *i.e.* $S \in V$.

A.3 Details for Training a Robust Model

A.3.1 Variational form for bounded loss functions

Some of *bounded* loss function examples represented in variational form are

(Geman-McClure) (Black & Rangarajan, 1996) $\rho(r) = \frac{r^2}{1+r^2} \quad \ell(r) = r^2$

$$\psi(\eta) = (\sqrt{\eta} - 1)^2, \tag{A.9}$$

(Geman-Reynolds) (Black & Rangarajan, 1996) $\rho(r) = \frac{|r|}{1+|r|} \quad \ell(r) = |r|$

$$\psi(\eta) = (\sqrt{\eta} - 1)^2, \tag{A.10}$$

(LeClerc) (Black & Rangarajan, 1996) $\rho(r) = 1 - \exp(-\ell(r)) \quad \ell(\cdot)$ convex

$$\psi(\eta) = \eta \log \eta - \eta + 1, \tag{A.11}$$

(Clipped-loss) (Yu et al., 2010) $\rho(r) = \max(1, \ell(r)) \quad \ell(\cdot)$ convex

$$\psi(\eta) = 1 - \eta. \tag{A.12}$$

A.3.2 Least Trimmed Loss

(Rousseeuw & Leroy, 1987a) recommends subset-selection based regression estimators, such as Least Trimmed Loss:

$$\hat{\boldsymbol{\theta}}_{LTL} \quad \in \quad \arg\min_{\boldsymbol{\theta}\in\Theta} \sum_{i=1}^{n'} \rho(r_{[i]}). \tag{A.13}$$

Here $r_{[i]}$ denotes sorted residuals $r_{[1]} \leq \cdots \leq r_{[n]}$ and $n' < n$ is the number of terms to consider. Traditionally $\rho(r) = r^2$ is used. These estimators are known to have high breakdown (Rousseeuw & Leroy, 1987a),when n' approaches $n/2$ the breakdown of (A.13) approaches $1/2$ (Rousseeuw & Leroy, 1987a). And obviously demonstrate bounded response to single outliers. Unfortunately, (A.13) is NP-hard (Bernholt, 2005).

A.3.3 Variational form of other Robust regression forms

Generalized M-estimation (GM-estimation) can be formulated simply by forcing each η_i to take on a specific value determined by $\|x_i\|$ or r_i (Rousseeuw & Leroy, 1987a), ignoring the auxilary function ψ. Least Trimmed Loss (A.13) can be expressed in the form (5.26) provided only that we add a shared constraint over $\boldsymbol{\eta}$:

$$\hat{\boldsymbol{\theta}}_{LTL} \quad \in \quad \arg\min_{\boldsymbol{\theta}\in\Theta} \quad \min_{0\leq\boldsymbol{\eta}\leq 1:\mathbf{1}'\boldsymbol{\eta}=n'} \quad \boldsymbol{\eta}'\boldsymbol{\ell}(\mathbf{r}) + \boldsymbol{\psi}(\boldsymbol{\eta}) \tag{A.14}$$

where $\psi(\eta_i) = 1 - \eta_i$ and $n' < n$ specifies the number of terms to consider in the sum of losses. Since $\boldsymbol{\eta} \in \{0,1\}^n$ at a solution (see e.g. (Yu et al., 2010)), (A.14) is equivalent to (A.13) if ψ is the clipped loss (A.12).

A.3.4 Proofs of NP-hardness

Our goal is to prove that minimizing a bounded loss function is NP-hard in general. After establishing the preliminary definitions required to be sufficiently precise about the results, we first prove that a special case—*clipped-loss* minimization—is strongly NP-hard in Section A.3.4. Then the NP-hardness of bounded loss minimization can be easily established by a reduction from clipped-loss minimization in Section A.3.4.

Hardness of Clipped Loss Minimization

Definition 3 (Loss function)**.** A function $\ell : \mathbb{R} \to \mathbb{R}$ is a *loss function* if (i) $\ell(r) \geq 0$ for all r; (ii) $\ell(0) = 0$; and (iii) $\ell(r)$ is nondecreasing in r away from $r = 0$; that is, $r_1 \leq r_2 \leq 0$ implies $\ell(r_1) \geq \ell(r_2)$, and $0 \leq r_1 \leq r_2$ implies $\ell(r_1) \leq \ell(r_2)$.

Definition 4 (τ-minimal two-sided loss)**.** A loss function ℓ is a *τ-minimal two-sided loss* if there exists a finite $B\tau > 0$ such that (i) $r \leq B\tau$ implies $\ell(r) \geq \tau$; and (ii) $r \geq B\tau$ implies $\ell(r) \geq \tau$.

Definition 5 (β-bounded loss)**.** A loss function ρ is a *β-bounded loss* if $\rho(y-\hat{y}) \leq \beta$ for all y and $\hat{y}$.

Definition 6 (weakly τ-minimal loss)**.** A loss function ℓ is a *weakly τ-minimal loss* if it is a $(\tau - \epsilon)$-minimal loss for all $\epsilon > 0$.

Definition 7 (Clipped loss minimization)**.**
Instance: $t \times n$ matrix X of training data, $t \times 1$ vector $\mathbf{y}$ of training labels, 1-minimal two-sided loss function ℓ, nonnegative threshold number c.
Question: Is there an $n \times 1$ vector $\boldsymbol{\theta}$ such that $\sum_{i=1}^{t} \min(1, \ell(y_i - X_{i:}\boldsymbol{\theta})) \leq c$?

Definition 8 (Bounded loss minimization)**.**
Instance: $t \times n$ matrix X of training data, $t \times 1$ vector $\mathbf{y}$ of training labels, 1-bounded and weakly 1-minimal two-sided loss function ℓ, nonnegative threshold number b.
Question: Is there an $n \times 1$ vector $\boldsymbol{\theta}$ such that $\sum_{i=1}^{t} \ell(y_i - X_{i:}\boldsymbol{\theta}) \leq b$?

Definition 9 (Maximum 2-satisfiability)**.**
Instance: Set U of variables, collection C of clauses over U such that each clause $c \in C$ has $|c| = 2$, positive integer $K \leq |C|$.
Question: Is there a truth assignment for U that simultaneously satisfies at least K of the clauses in C?

Note that MAX2SAT is known to be NP-complete in general (Garey & Johnson, 1979). It is solvable in polynomial-time if $K = |C|$ (Garey & Johnson, 1979), but NP-hard to approximate within a multiplicative constant better than 21/22 = 0.95454 (Håstad, 2001).

Theorem 7. Clipped loss minimization is strongly NP-hard.

Proof. We transform MAX2SAT to clipped-loss minimization. Let (U, C, K) constitute an instance of MAX2SAT.

Let $\mathbf{1}_i$ denote a boolean vector with a single 1 in position i. We will also use a scale factor s that will be specified below (we will be able to choose a value for s that is polynomial in $|U|$). **Widget construction:** For each variable in $u_j \in U$ associate a feature $X_{:j}$ with corresponding weight θ_j. For each clause $c \in C$ construct three training examples in the form $(\mathbf{x}^\top, y)$ as follows:

- For $u_i \vee u_j$ clauses, construct three examples $(s\mathbf{1}_i^\top, s)$, $(s\mathbf{1}_j^\top, s)$ and $(s(\mathbf{1}_i + \mathbf{1}_j)^\top, s)$.
- For $u_i \vee \neg u_j$ clauses, construct three examples $(s\mathbf{1}_i^\top, s)$, $(s\mathbf{1}_j^\top, 0)$ and $(s(\mathbf{1}_i - \mathbf{1}_j)^\top, 0)$.
- For $\neg u_i \vee u_j$ clauses, construct three examples $(s\mathbf{1}_i^\top, 0)$, $(s\mathbf{1}_j^\top, s)$ and $(s(\mathbf{1}_i - \mathbf{1}_j)^\top, 0)$.
- For $\neg u_i \vee \neg u_j$ clauses, construct three examples $(s\mathbf{1}_i^\top, 0)$, $(s\mathbf{1}_j^\top, 0)$ and $(s(\mathbf{1}_i + \mathbf{1}_j)^\top, s)$.

To illustrate how this construction will work, consider a training example $(s\mathbf{1}_i^\top, s)$. For a weight vector $\boldsymbol{\theta}$ one will obtain a prediction $\hat{y} = s\mathbf{1}_i^\top \boldsymbol{\theta} = s\theta_i$, which is compared to the target value $y = s$. Note that if $\theta_i = 1$, then the loss must be zero on this example, since $\hat{y} = s = y$. However, as θ_i moves away from 1 the loss must increase until it "saturates". Let

$$\rho(y - \hat{y}) \;\; = \;\; \min(1, \ell(y - \hat{y})) \tag{A.15}$$

denote the "clipped" loss. Since ℓ is a 1-minimal two-sided loss there must exist a finite $B_1 > 0$ such that for any y, $\hat{y} \leq y - B_1$ implies $\rho(y - \hat{y}) = 1$ and $\hat{y} \geq y + B_1$ implies $\rho(y - \hat{y}) = 1$. Conversely, if $\rho(y - \hat{y}) < 1$ then it must follow that $y - B_1 < \hat{y} < y + B_1$. The role of the scale factor s, therefore, is to control the width of the "trough" where the losses remain strictly less than 1: Note that for any scale factor $s > 0$, if $\rho(sy - s\hat{y}) < 1$ then $sy - B_1 < s\hat{y} < sy + B_1$,

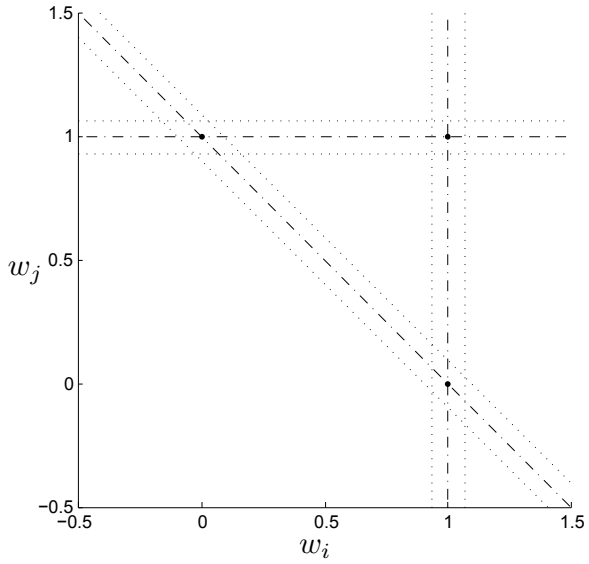


Figure A.4: Depiction of the error surface for the $u_i \vee u_j$ type of clause (for $i \neq j$)

which holds if and only if $y - B_1/s < \hat{y} < y + B_1/s$. Thus, the larger the choice of s, the narrower the trough where losses less than 1 can be achieved for a given training example.

Figure A.4, Figure A.5, Figure A.6 and Figure A.7 depict the error surfaces in (θ_i, θ_j) created by the set of three training examples for each of the four clause types (for sufficiently large s) when the variables are distinct, $i \neq j$(The trough widths are controlled by the scale factor s.). Note that the minimum loss any weight vector can achieve on a clause (i.e., on its associated set of three training examples) is always 1, and this can only be achieved by assigning boolean weights that "satisfy" the clause.

Choosing the scale factor s: The scale factor s needs to be fixed to a sufficiently large value so that, for any weight vector $\boldsymbol{\theta}$, there exists a "gap" between 0 and $\frac{1}{2}$ (and a corresponding mirror gap between $\frac{1}{2}$ and 1) that contains no individual weight component θ_i. In particular, consider a partition of the interval $[\frac{B_1(1+\sqrt{2})}{s}, \frac{1}{2}]$ into disjoint bins of size $\frac{B_1\sqrt{2}}{s}$. There are at least $\lfloor \left(\frac{1}{2} - \frac{B_1(1+\sqrt{2})}{s}\right) / \frac{B_1\sqrt{2}}{s} \rfloor =$

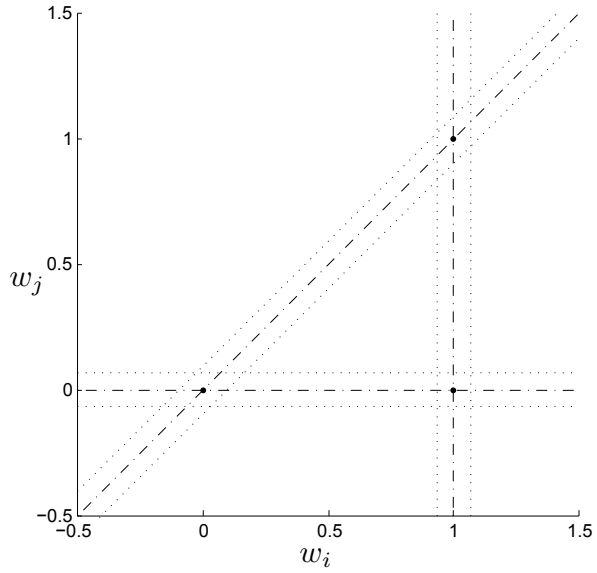


Figure A.5: Depiction of the error surface for the $u_i \vee \neg u_j$ type of clause (for $i \neq j$)

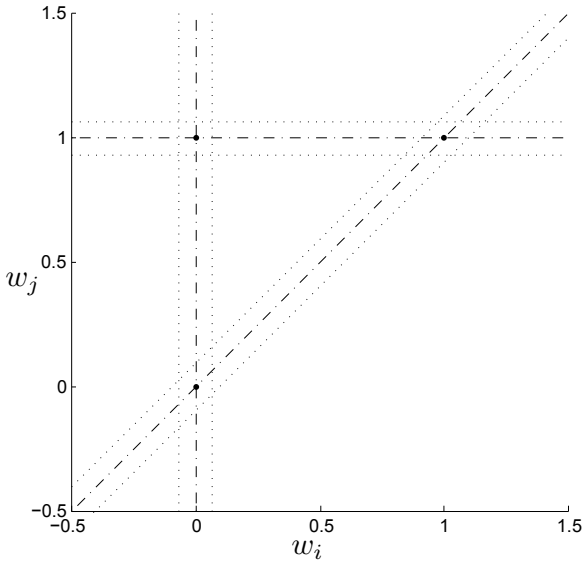


Figure A.6: Depiction of the error surface for the $\neg u_i \vee u_j$ type of clause (for $i \neq j$)

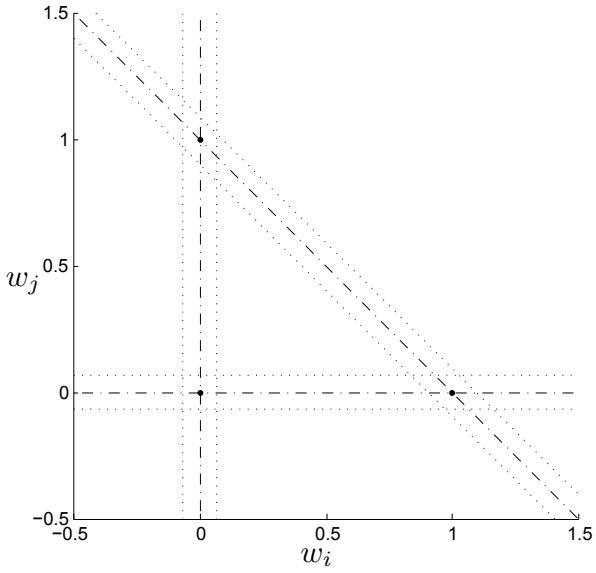


Figure A.7: Depiction of the error surface for the $\neg u_i \vee \neg u_j$ type of clause (for $i \neq j$)

$\lfloor \frac{s}{B_1 2\sqrt{2}} - \frac{1}{\sqrt{2}} - 1 \rfloor$ such bins that fit entirely within the interval.[1] So, by setting

$$s \quad = \quad B_1 2\sqrt{2}(|U| + 3) \tag{A.16}$$

it follows that there must be at least $|U| + 1$ disjoint bins of width $\frac{B_1\sqrt{2}}{s}$ within the interval $[\frac{B_1(1+\sqrt{2})}{s}, \frac{1}{2}]$, plus a mirror set of $|U| + 1$ disjoint bins within the interval $[\frac{1}{2}, 1 - \frac{B_1(1+\sqrt{2})}{s}]$; see Figure A.8. Given that the weight vector only has $|U|$ components, by construction, we know that for any particular $\boldsymbol{\theta}$ there must be at least one (bin, mirror bin) pair such that neither bin contains any θ_i value; see Figure A.9. We will use these empty bins to define which θ_i values are considered to be "nearly discrete" versus "close to $\frac{1}{2}$", as shown in Figure A.9.

Main argument: We now proceed to show that there is always a boolean weight

[1] The reason for considering the interval $[\frac{B_1(1+\sqrt{2})}{s}, \frac{1}{2}]$ instead of $[0, \frac{1}{2}]$ is that we also need to ensure the empty bin does not intersect the parallelogram at the interesction between an axis-parallel and diagonal trough. For a given s this will be true for any bin that is at least $\frac{B_1(1+\sqrt{2})}{s}$ away from 0 or 1.

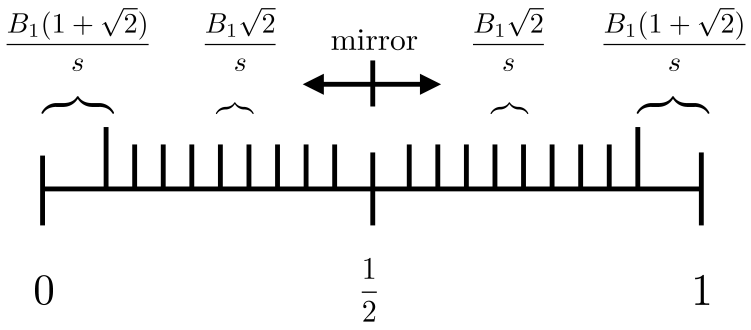


Figure A.8: Illustrating bins of width $\frac{B_1\sqrt{2}}{s}$ between $\frac{B_1(1+\sqrt{2})}{s}$ and $\frac{1}{2}$, and their mirror images between $\frac{1}{2}$ and $1 - \frac{B_1(1+\sqrt{2})}{s}$.

vector that is a global minimizer of the total clipped loss. Let

$$\rho(\boldsymbol{\theta}) \quad = \quad \sum_{i=1}^{t} \rho(y_i - X_{i:}\boldsymbol{\theta}) \tag{A.17}$$

denote the total clipped loss obtained by a weight vector $\boldsymbol{\theta}$ on the constructed set of training examples. Let $\boldsymbol{\theta}^{(0)}$ be a global minimizer of (A.17). Then we know that there must be some (bin, mirror bin) pair that contains no component of $\boldsymbol{\theta}^{(0)}$, which we can use to define "nearly discrete" versus "close to $\frac{1}{2}$" weight values (Figure A.9). Then by Lemma 6 below we know there must be some weight vector $\boldsymbol{\theta}^{(1)}$ which contains only boolean or "close to $\frac{1}{2}$" values such that $\rho(\boldsymbol{\theta}^{(1)}) \leq \rho(\boldsymbol{\theta}^{(0)})$. Then by Lemma 7 below we know there must exist a pure boolean weight vector $\boldsymbol{\theta}^{(2)}$ such that $\rho(\boldsymbol{\theta}^{(2)}) \leq \rho(\boldsymbol{\theta}^{(1)})$. Finally given a boolean assignment $\boldsymbol{\theta}^{(2)}$, a corresponding truth assignment $\mathbf{u}^{(2)}$ can be directly recovered via the transformation

$$\begin{aligned} u_i = \text{true} \quad &\Leftrightarrow \quad \theta_i = 1 \\ u_i = \text{false} \quad &\Leftrightarrow \quad \theta_i = 0. \end{aligned} \tag{A.18}$$

Furthermore, for a boolean weight vector $\boldsymbol{\theta}^{(2)}$ we have that

$$\begin{aligned} \rho(\boldsymbol{\theta}^{(2)}) = 3 \times |\{\text{clauses falsified by } \mathbf{u}^{(2)}\}| & \\ + 1 \times |\{\text{clauses satisfied by } \mathbf{u}^{(2)}\}| & \qquad \text{(A.19)} \\ = 3|C| - 2 \times |\{\text{clauses satisfied by } \mathbf{u}^{(2)}\}|. & \qquad \text{(A.20)} \end{aligned}$$

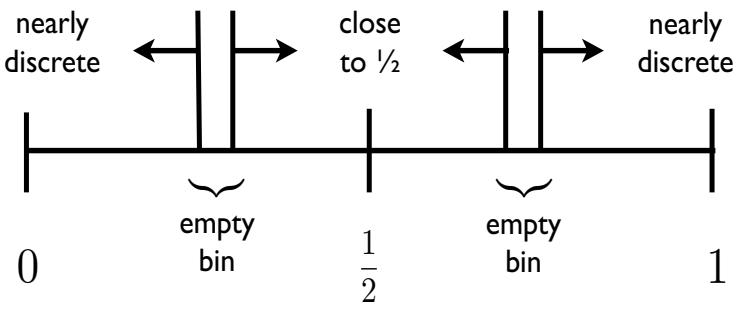


Figure A.9: Illustrating how an empty (bin, mirror bin) pair can be used to define which component weight values are "nearly discrete" versus "close to $\frac{1}{2}$".

Lemma 6. For any weight vector $\boldsymbol{\theta}$ there is always a weight vector $\tilde{\boldsymbol{\theta}}$ with only boolean components or components that are "close to $\frac{1}{2}$" such that $\rho(\tilde{\boldsymbol{\theta}}) \leq \rho(\boldsymbol{\theta})$.

Proof. Fix $\boldsymbol{\theta}$ and use one of its empty bins to define which component values are "nearly discrete" versus "close to $\frac{1}{2}$". Let $\tilde{\boldsymbol{\theta}}$ be the vector obtained by rounding all "nearly discrete" values of $\boldsymbol{\theta}$ to their nearest boolean value. We show that the total clipped loss obtained by $\tilde{\boldsymbol{\theta}}$ cannot be greater than that of $\boldsymbol{\theta}$.

Consider Figure A.10, which depicts the effect of rounding for a clause of the $u_i \vee u_j$ type (the other three cases are isomorphic). For any clause, there are two situations to consider: when both weights are rounded and when only one weight is rounded.

First, consider the case where both weights (θ_i, θ_j) are "nearly discrete" and hence both are rounded to their nearest boolean value. This corresponds to being in one of the *red* corner quadrants shown in Figure A.10. Weight pairs (θ_i, θ_j) in the four *red* corner quadrants are rounded to the boolean corners. Weight pairs (θ_i, θ_j)

in the four *blue* semi-rectangles on the edges are rounded to the line where the "nearly discrete" value becomes boolean. Weight pairs (θ_i, θ_j) in the *green* central square are not rounded. No weight pair occurs in the cross-hatch formed by the empty bins. By inspection, one can see that the boolean assignment within each quadrant achieves the minimum loss attainable within the quadrant. In particular, in the three "satisfying" quadrants the boolean assignment is the strict minimizer of the loss, whereas in the "falsifying" quadrant all assignments achieve the same loss of 3. Therefore, the total loss cannot have increased on such a clause due to rounding.

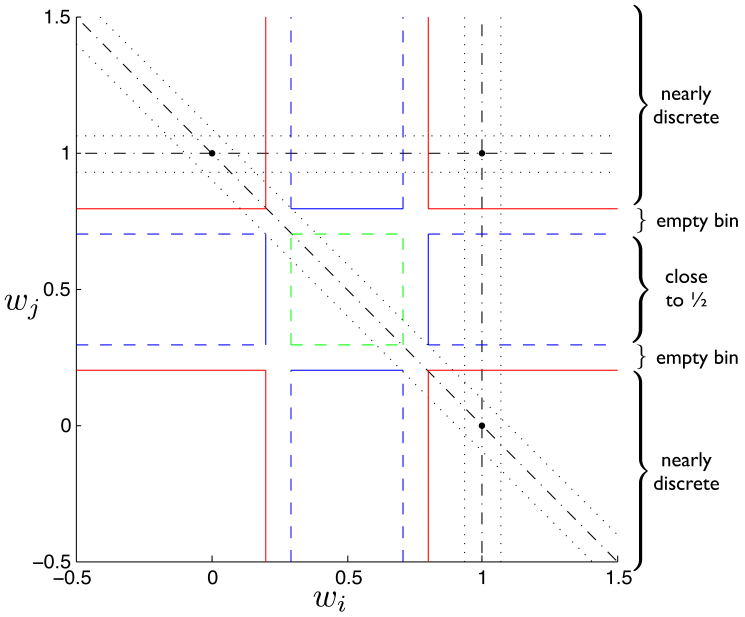


Figure A.10: Illustrating the rounding scheme of Lemma 6 for a $u_i \vee u_j$ clause. (The scheme for the other three clause types depicted in Figures A.4, A.5, A.6 and A.7 are isomorphic.

Second, consider the case where one of the weights is "nearly discrete" and hence rounded, while the other is "close to $\frac{1}{2}$" and hence not rounded. This corresponds to being in one of the *blue* semi-rectangles on the edges shown in Figure A.10. Once again, by inspection one can see that rounding the "nearly discrete"

weight to its nearest boolean value cannot increase the loss obtained. In particular, rounding the "nearly discrete" variable yields the minimum loss assignment in the top and right semi-rectangles, while every assignment in the bottom and left semi-rectangles obtains the same loss of 3. This latter property is precisely what is achieved by the bin definitions: the empty bin width of $\frac{B_1\sqrt{2}}{s}$ is sufficient to ensure that the *blue* semi-rectangles do not intersect the diagonal trough. Therefore, the total loss cannot have increased on such a clause due to rounding.

In clauses where both weights are "close to $\frac{1}{2}$" (the *green* square in the middle) neither weight is rounded, hence the loss does not change. The result follows. ∎

Lemma 7. For any weight vector $\boldsymbol{\theta}$ with only boolean components or components that are "close to $\frac{1}{2}$" there is always a boolean weight vector $\hat{\boldsymbol{\theta}}$ such that $\rho(\hat{\boldsymbol{\theta}}) \leq \rho(\boldsymbol{\theta})$.

Proof. Consider Figure A.11, which depicts the possible weight pairs (θ_i, θ_j) participating in a clause of type $u_i \vee u_j$. (The situation for any of the clause types is isomorphic.) The isolated *red* points are the only purely boolean values, the *blue* line segments indicate pairs where one value is boolean, and the *green* square shows the region where both weights are "close to $\frac{1}{2}$". Observe that any weight pair with one or more "close to $\frac{1}{2}$" values (i.e., the union of the *blue* lines and the *green* square) has loss at least 2.

Given the weight vector $\boldsymbol{\theta}$, patition the clauses into three subsets: the set of clauses where both weight values are boolean ("closed clauses"), the set of clauses where one weight is boolean and the other is not ("mixed clauses"), and the set of clauses where both weights are "close to $\frac{1}{2}$" ("open clauses"). Let the set of "non-closed clauses" consist of the union of the "mixed clauses" and the "open clauses". From above, we know that $\boldsymbol{\theta}$ achieves a loss of at least 2 on each "non-closed clause". Let c denote the loss per clause achieved by $\boldsymbol{\theta}$ on the "closed clauses". There are two cases to consider.

Case 1: ($c \geq 2$) In this case the overall loss per clause achieved by $\boldsymbol{\theta}$ must be

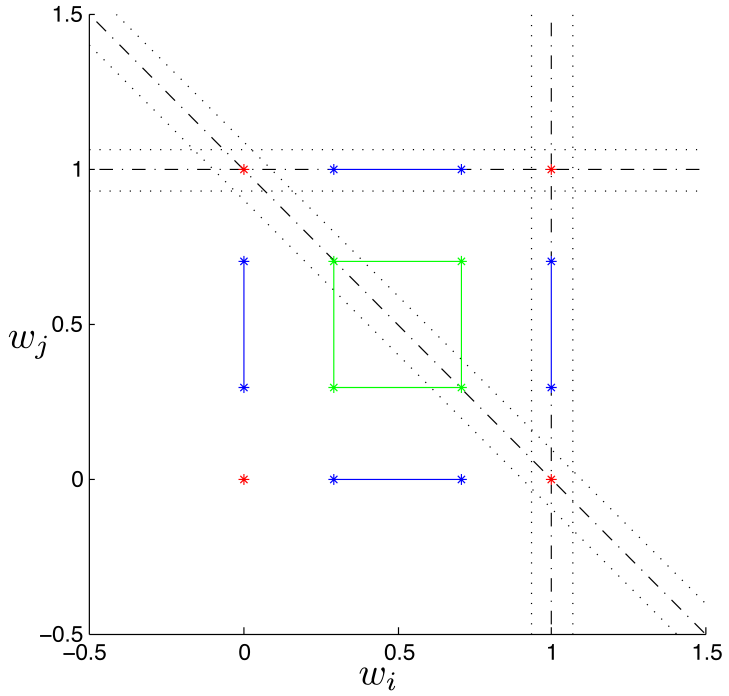


Figure A.11: Possible weight pairs that can occur after rounding.

at least 2, which implies the total loss is $\rho(\boldsymbol{\theta}) \geq 2|C|$. Therefore, by Lemma 8 below we know that there must exist some assignment $\hat{\mathbf{u}}$ to the variables in U that satisfies at least $\frac{|C|}{2}$ of the clauses in C. Let $\hat{\boldsymbol{\theta}}$ denote the corresponding boolean weight vector, recovered via the translation (A.18). By (A.20) we know that

$$\begin{aligned} \rho(\hat{\boldsymbol{\theta}}) &= 3|C| - 2 \times |\{\text{clauses satisfied by } \mathbf{u}^{(2)}\}| && \text{(A.21)} \\ &\leq 2|C| && \text{(A.22)} \\ &\leq \rho(\boldsymbol{\theta}). && \text{(A.23)} \end{aligned}$$

Case 2: ($c < 2$) Let B denote the indices where $\boldsymbol{\theta}$ is boolean, and let N denote the indices where $\boldsymbol{\theta}$ is "close to $\frac{1}{2}$". (Since $c < 2$ there must be at least one component θ_i that is boolean.) Fix the boolean components of $\boldsymbol{\theta}_B$, which in turn fixes the outcomes on the "closed clauses". For any "mixed clause" let θ_i denote the weight that is boolean. We would like to preserve the assignment of θ_i. Therefore, temporarily

replace such a mixed clause with a new singleton clause defined by substituting u_i with its partner u_j. Consider the new set of altered "mixed clauses", in union with the set of "open clauses". Denote this new set $\tilde{C}$. Note that $\tilde{C}$ is defined only on the subset of variables U_N corresponding to weights in $\boldsymbol{\theta}$ that are "close to $\frac{1}{2}$". By Lemma 8 below, there must be some assignment $\tilde{\mathbf{u}}_N$ that satisfies at least half of the clauses in $\tilde{C}$, hence there exists a corresponding boolean weight vector $\tilde{\boldsymbol{\theta}}_N$ that achieves a loss of at most $2|\tilde{C}|$ on $\tilde{C}$.

Now consider the boolean weight vector $\hat{\boldsymbol{\theta}}$ formed by conjoining $\boldsymbol{\theta}_B$ with $\tilde{\boldsymbol{\theta}}_N$. Note that for any "closed clause" $\boldsymbol{\theta}$ and $\hat{\boldsymbol{\theta}}$ behave identically and hence achieve the same loss. However, recall $\boldsymbol{\theta}$ achieves a loss of at least 2 on each "non-closed clause", while by the above construction, $\hat{\boldsymbol{\theta}}$ achieves a loss of at most 2 per clause over the "non-closed clauses". (In particular, even though $\hat{\mathbf{u}}_N$ was constructed by satisfying temporarily altered "mixed clauses" above, if it satisfies such a clause, then it must also satisfy the original "mixed clause".) Hence $\rho(\hat{\boldsymbol{\theta}}) \leq \rho(\boldsymbol{\theta})$.

Since in each of the two cases we were able to identify a boolean weight vector $\hat{\boldsymbol{\theta}}$ that achieves a loss no worse than $\boldsymbol{\theta}$, the result must follow. ∎

Lemma 8. For any MAX2SAT instance (U, C, K) and any assignment $\mathbf{u}$ to its variables U, either $\mathbf{u}$ or its negation $\neg\mathbf{u}$ must satisfy at least $\frac{|C|}{2}$ of the clauses in C.

Proof. Consider any assignment $\mathbf{u}$. Note that $\neg\mathbf{u}$ must satisfy each clause that $\mathbf{u}$ falsifies. Therefore, if $\mathbf{u}$ satisfies fewer than $\frac{|C|}{2}$ clauses, it must falsify at least $\frac{|C|}{2}$ of the clauses, hence $\neg\mathbf{u}$ would have to satisfy at least $\frac{|C|}{2}$ of the clauses. Otherwise, $\mathbf{u}$ satisfies at least $\frac{|C|}{2}$ of the clauses. ∎

Hardness of Bounded Loss Minimization

Finally, we are in a position to prove Theorem 2 from the main body. Recall the definition of bounded loss minimization (Definition 8).

Theorem 2. Bounded (non-constant) loss minimization is NP-hard.

Proof. We transform MAX2SAT to bounded-loss minimization. Let (U, C, K) constitute an instance of MAX2SAT. Use the same widget construction as in the proof of Theorem 7. Let

$$t = 3|C| \tag{A.24}$$

$$\ell(\boldsymbol{\theta}) = \sum_{i=1}^{t} \ell(y_i - X_{i:}\boldsymbol{\theta}) \tag{A.25}$$

$$\ell^* = \min_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}) \tag{A.26}$$

on the constructed training set.

To prove that minimizing the bounded loss ℓ can be used to decide whether K clauses can be satisfied, first consider an intermediate clipped-loss version of the problem. In particular, let

$$\hat{\ell}(y - \hat{y}) = \frac{1}{1 - \frac{1}{2t}} \ell(y - \hat{y}) \tag{A.27}$$

$$\hat{\rho}(y - \hat{y}) = \min(1, \hat{\ell}(y - \hat{y})) \tag{A.28}$$

$$\hat{\rho}(\boldsymbol{\theta}) = \sum_{i=1}^{t} \hat{\rho}(y_i - X_{i:}\boldsymbol{\theta}) \tag{A.29}$$

$$\hat{\rho}^* = \min_{\boldsymbol{\theta}} \hat{\rho}(\boldsymbol{\theta}) \tag{A.30}$$

on the constructed training set. Note that $\hat{\ell}$ is a 1-minimal loss with $\hat{B}_1 = B_{1-\frac{1}{2t}}$ (see Definition 4). Therefore, by Theorem 7, $\hat{\rho}^*$ can be used to decide whether K clauses can be satisfied in the original MAX2SAT instance. Recall also that $\hat{\rho}^*$ must be integer valued in this case.

Lemma 9 below shows that $\hat{\rho}^* - \frac{1}{2} \leq \ell^* \leq \hat{\rho}^*$. It then follows that for any integer n, $\hat{\rho}^* \leq n$ if and only if $\ell^* < n + \frac{1}{2}$. To see why this must hold, note: $(\Leftarrow)$

if $\ell^* < n + \frac{1}{2}$ then $\hat{\rho}^* \leq \ell^* + \frac{1}{2} < n + 1$, hence $\hat{\rho}^* \leq n$ since $\hat{\rho}^*$ is integer valued; and also ($\Rightarrow$) if $\ell^* \geq n + \frac{1}{2}$ then $\hat{\rho}^* \geq \ell^* \geq n + \frac{1}{2}$, hence $\hat{\rho}^* > n$ since $\hat{\rho}^*$ is integer valued. Therefore, ℓ^* can be used to decide what integer value $\hat{\rho}^*$ achieves, which in turn can be used to decide whether K clauses can be satisfied by Theorem 7. ∎

Note that bounded-loss minimization is strongly NP-hard if $B_{1-\frac{1}{2t}}$ is polynomial in $t = 3|C|$.

Lemma 9. $\hat{\rho}^* - \frac{1}{2} \leq \ell^* \leq \hat{\rho}^*$.

Proof. First, note that for any $(y, \hat{y})$

$$\begin{aligned}
\hat{\rho}(y - \hat{y}) - \tfrac{1}{2t} &\leq (1 - \tfrac{1}{2t})\hat{\rho}(y - \hat{y}) && \text{(A.31)}\\
&= (1 - \tfrac{1}{2t}) \min(1, \hat{\ell}(y - \hat{y})) && \text{(A.32)}\\
&= \min(1 - \tfrac{1}{2t}, \ell(y - \hat{y})) && \text{(A.33)}\\
&\leq \ell(y - \hat{y}) && \text{(A.34)}\\
&\leq \hat{\rho}(y - \hat{y}). && \text{(A.35)}
\end{aligned}$$

Therefore, for any $\boldsymbol{\theta}$

$$\hat{\rho}(\boldsymbol{\theta}) - \tfrac{1}{2} \leq \ell(\boldsymbol{\theta}) \leq \hat{\rho}(\boldsymbol{\theta}). \quad \text{(A.36)}$$

If we let $\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta})$ and $\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \hat{\rho}(\boldsymbol{\theta})$, it is then immedate that

$$\begin{aligned}
\hat{\rho}(\hat{\boldsymbol{\theta}}) - \tfrac{1}{2} &\leq \hat{\rho}(\boldsymbol{\theta}^*) - \tfrac{1}{2} && \text{(A.37)}\\
&\leq \ell(\boldsymbol{\theta}^*) && \text{(A.38)}\\
&\leq \ell(\hat{\boldsymbol{\theta}}) && \text{(A.39)}\\
&\leq \hat{\rho}(\hat{\boldsymbol{\theta}}). && \text{(A.40)}
\end{aligned}$$

∎

Appendix B

Some Basic Background

In this chapter, I present basic tutorial background on convexity, convex approximations in ML, deep learning, and robust estimation. In particular, I cover basic background on convexity in Section B.1 and then present background on deep learning and robust estimation in Sections B.3 and B.4, respectively.

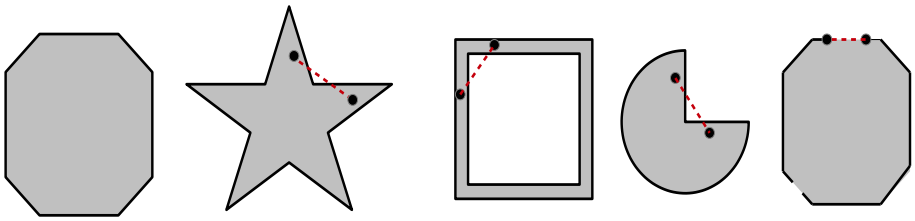
B.1 Convexity Background

This section covers some essential background on convex optimization for the sake of completeness. Let us start by giving the fundamental definitions.

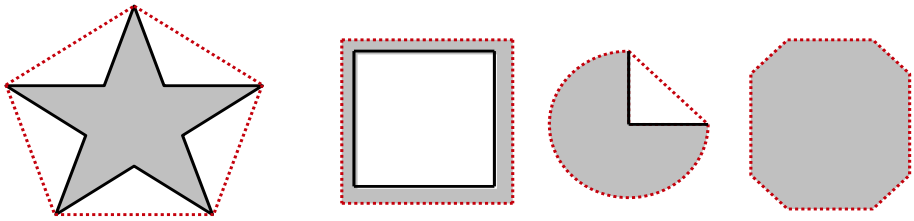
Definition B.1.1. Convex Combination. Given a set $S \subseteq \mathbb{R}^n$ and points $x_i, \ldots, x_k \in S$, $\sum_{i=1}^{k} \alpha_i x_i$ is called a *convex combination* where $\sum_{i=1}^{k} \alpha_i = 1$ and $\alpha_i \geq 0$ for all $i \in \{1, \ldots, k\}$ (Boyd & Vandenberghe, 2004).

Definition B.1.2. Convex sets. A set $S \subseteq \mathbb{R}^n$ is a *convex set* if and only if it contains every convex combination of points in S (Boyd & Vandenberghe, 2004).

Some examples for shapes of convexity of sets are shown in Figure B.1(a). Geometrically, a set is convex if every line between two points is inside the set. First set is convex whereas the rest are nonconvex. Red line in each nonconvex



(a) Convex and non convex sets.



(b) Convex hull.

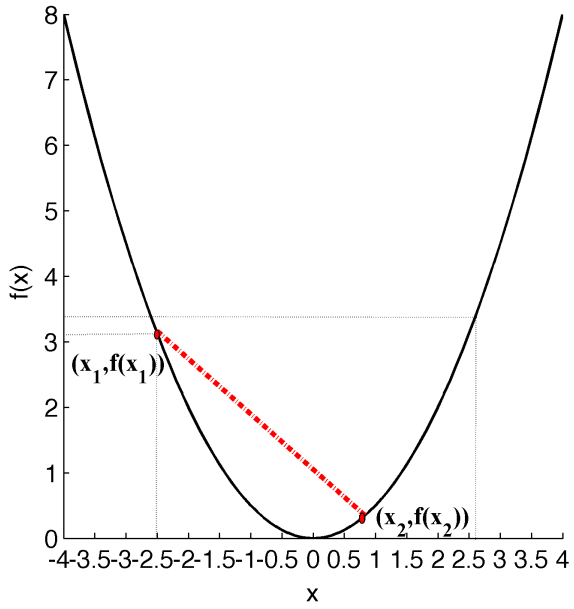
Figure B.1: Some convex and nonconvex sets and the illustration of convex hull (red dotted lines and inside) of nonconvex sets

set shows that some points in this line do not lie in the set hence these sets do not satisfy the requirement in the Definition B.1.2.

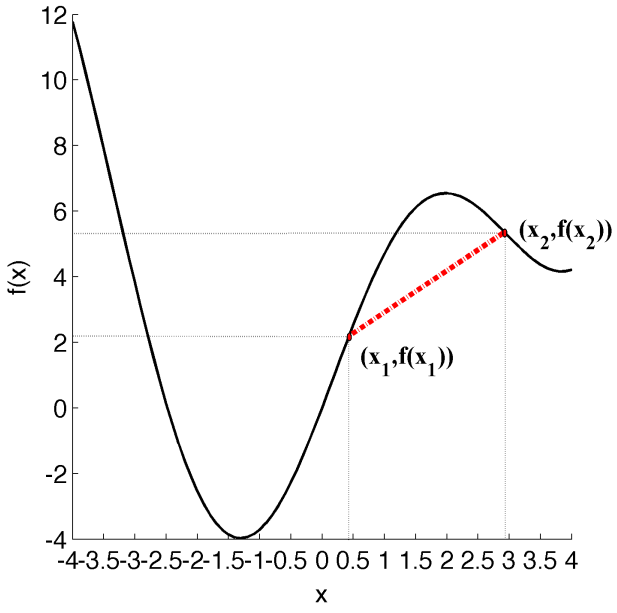
Definition B.1.3. Convex Hull. Given a set $S \subseteq \mathbb{R}^n$, *convex hull*, denoted as $\text{conv}(S)$ is a set constructed by all convex combinations of points in S, i.e. $\text{conv}(S) = \{\sum_{i=1}^{k} \theta_i x_i | x_i \in S, \alpha_i \geq 0, i = 1, \ldots, k, \sum_{i=1}^{k} \alpha_i = 1\}$.

It is not hard to see that convex hull of set S is the smallest convex set containing S. Hence, the convex hull of a convex set equals to itself. The convex hulls of the non-convex sets in Figure B.1(a) are shown in Figure B.1(b) where the convex hull is the red lines and interior. Convex hulls are a particularly important tool to approximate nonconvex sets with convex ones via convex relaxations, which will be covered in Section B.2.1.

Definition B.1.4. Convex functions. If a function $f : \mathbb{R}^n \mapsto \mathbb{R}$ that has a convex domain S satisfies $f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha) f(y)$ for all points $x, y \in S$ and for all $\alpha \in [0, 1]$, then the function f is a *convex function*.



(a) Convex function



(b) Nonconvex function

Figure B.2: The shapes of some convex and nonconvex functions.

Geometrically, if the line segments between every two points in the domain of a function is above the graph of this function between these two points, it is a convex function. The quadratic function is a common convex function and it is given in Figure B.2(a). A nonconvex function is given in Figure B.2(b) and a red line segment shown is below the graph. For common problems, the function can contain many local minima. Some well-known convex functions are exponential, power function, logarithm, negative entropy, max function, norms, log-sum-exp and log-determinant (Boyd & Vandenberghe, 2004).

Definition B.1.5. Local minima.. If a point $x^* \in S$ where S is the domain of a function f and if there exists some $\epsilon > 0$ such that $f(x^*) \leq f(x)$ for all $x \in S$ with $||x^* - x|| \leq \epsilon$, x^* is called *local minimizer (or minima)*.

Definition B.1.6. Global minima.. If a point $x^* \in S$ where S is the domain of a function f and $f(x^*) \leq f(x)$ for all $x \in S$, x^* is called *global minimizer (or minima)*.

The definition of convex functions and convex sets are used to assess the convexity of to objective function and constraint set respectively in mathematical programs. A mathematical program with the form

$$\min_{x \in S} f(x) \tag{B.1}$$

is the minimization of an objective function f over a constraint set S. If f is a convex function and S is a convex set, then (B.1) is called a *convex optimization problem* and all local minima are global minima. If either f is not a convex function or S is not a convex set, (B.1) is not a convex problem and the algorithms that try to find a locally optimal point without any guarantee to find a global minimum are usually classified in *local optimization* (Boyd & Vandenberghe, 2004). There are well-known drawbacks of local methods. First, initialization has a significant effect on the results thereby local methods may need careful initialization. Second, there

is no guarantee for solution quality. Third, they can be very sensitive to parameters of the algorithm. However, convex problems do not suffer from any of these drawbacks since every local minimum is the global minimum. Hence solving convex optimization problems is much more convenient and straightforward.

B.2 Semidefinite Relaxations

B.2.1 Semidefinite Programming

Semidefinite programming (SDP), is the optimization problem that minimizes a linear function of a variable subject to the constraint that an affine combination of symmetric matrices with coefficient vector equivalent to the variable is positive semidefinite (Vandenberghe & Boyd, 1996).

The SDP that minimizes a linear function of $x \in \mathbb{R}^m$ is

$$\begin{aligned} \min_{x} \quad & c'x \\ \text{s.t.} \quad & F(x) \succeq 0, \ i = 1, \ldots, m. \\ & Ax = b \end{aligned} \tag{B.2}$$

where $F(x) = F_0 + \sum_{i=1}^{m} x_i F_i$, $c \in \mathbb{R}^m$ and symmetric matrices $F_i \in \mathbb{R}^{n \times n}$, $A \in \mathbb{R}^{p \times m}$ and $b \in \mathbb{R}^p$ (Boyd & Vandenberghe, 2004). The inequality symbol $\succeq$ is used for positive semidefinite constraint. The positive semidefinite constraint is nonlinear and non-smooth however it is convex and this yields a convex optimization problem (Vandenberghe & Boyd, 1996; Robert & Hande Yurttan, 2000).

SDP can take two specific forms: standard form and inequality form (Boyd & Vandenberghe, 2004). Standard form SDP for $X \in \mathbb{S}^m$ is formulated as

$$\begin{aligned} \min_{x} \quad & tr(CX) \\ \text{s.t.} \quad & tr(A_i X) = b_i, \ i = 1, \ldots, p. \\ \text{s.t.} \quad & X \succeq 0 \end{aligned} \tag{B.3}$$

where $C, A_i \in \mathbb{S}^m$ for $i = 1, \ldots, p$. And inequality form SDP is formulated as

$$\begin{aligned} \min_{x} \quad & c'x \\ \text{s.t.} \quad & F(x) \succeq 0.\ i = 1, \ldots, m. \end{aligned} \tag{B.4}$$

SDP's have been widely used as a useful tool to deal with NP-hard combinatorial optimization problems (Vandenberghe & Boyd, 1996). In particular, they have been mostly applied to nonconvex quadratically constrained quadratic program (denoted as QCQP or Q^2P) because many well-known NP-hard combinatorial problems can easily be reformulated as non-convex QCQP (d'Aspremont & Boyd, 2003). A QCQP is formulated as

$$\begin{aligned} \min_{x} \quad & x'P_0x + q_0x + r_0 \\ \text{s.t.} \quad & x'P_ix + q_ix + r_i \leq 0.\ i = 1, \ldots, m. \end{aligned} \tag{B.5}$$

where $P_i \in \mathbb{S}^n_+, i = 0, 1 \ldots m$ where $\mathbb{S}$ denotes set of positive semidefinite matrices (Boyd & Vandenberghe, 2004; Nesterov et al., 2000). If all of the P_i's are positive semidefinite, then the QCQP is a convex problem. If at least one of the P_i's is not positive semidefinite, then it is not a convex problem. The non-convex QCQP is NP-hard (d'Aspremont & Boyd, 2003). While there is no complete proof about the hardness of them, they have been considered as hard since the complexity of all the algorithms known to solve them are exponential in terms of problem dimensions.

Here a well-known known hard optimization problem that is encountered frequently and we introduce that it can be formulated as a QCQP.

Example B.2.1. MAX-CUT. *Consider an undirected graph $G = (V, E)$ with n nodes and a corresponding nonnegative symmetric weight matrix W for arcs of the graph where $W_{ij} = 0$ if $(i, j) \notin E$. Given a partition of nodes of G to disjoint sets S and $\overline{S}$, $cut(S, \overline{S})$ denotes the edges that have one end point in S and the other in $\overline{S}$ (Goemans & Williamson, 1995). The problem of finding the cut with largest total weights is called the* maximum cut *(MAX-CUT) problem. The maximum cut*

problem can be formulated as the following integer program

$$\begin{aligned} \max_{x} \quad & \frac{1}{2}\sum_{i<j} W_{ij}(1 - x_i x_j) \\ \textit{s.t.} \quad & x_i \in \{-1, 1\} \quad \forall i \in V. \end{aligned} \tag{B.6}$$

where x_i determines the partition node i belongs to. Letting $\tilde{W}_{ij} = -W_{ij},\ \forall\ i \neq j$ where $i, j \in V$ and $\tilde{W}_{ii} = \sum_{j=1,\ldots,n} W_{ij}$, Equation B.6 can be written as the following nonconvex QCQP (d'Aspremont & Boyd, 2003),

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & x'\tilde{W}x \\ \text{s.t.} \quad & x_i^2 = 1, \quad i = 1, \ldots, n, \end{aligned} \tag{B.7}$$

where $\tilde{W}$ is a positive definite matrix. MAX-CUT has long been known as NP-complete problem (Goemans & Williamson, 1995). Rewriting the hard integer programming problem as a non-convex QCQP has been a crucial step to deal with many other hard combinatorial optimization problems since it allows applying a powerful technique to find a convex approximation of non-convex problem which will be covered in Section B.2.1 in more detail.

The *semidefinite relaxation* (SDR) of combinatorial problems is performed after rewriting them first as the QCQP given in Equation B.5. Recall that $x'P_i x = \mathrm{tr}(xx'P_i)$, substituting new variable $X = xx'$ allows the following exact reformulation

$$\begin{aligned} \min_{x\mathbb{R}^n, X \in \mathcal{S}^n} \quad & \mathrm{tr}(XP_0) + q_0 x + r_0 \\ \text{s.t.} \quad & \mathrm{tr}(XP_i) + q_i x + r_i \leq 0,\ i = 1, \ldots, m. \\ & X = xx' \end{aligned} \tag{B.8}$$

This has been called the *lifting procedure* (Lemaréchal & Oustry, 1999). Every term except the last constraint in the last problem is now linear, hence convex. The next key observation is that $X = xx'$ is equivalent to $X \succeq xx'$ and $rank(X) = 1$ (Vandenberghe & Boyd, 1996). The $X \succeq xx'$ is PSD constraint hence convex. However the rank constraint is non-convex (Natarajan, 1995; Davis et al., 1997).

Simply dropping the rank constraint and writing $X \succeq xx'$ as the equivalent the Schur complement yields the following SDR

$$\begin{aligned} \min_{x\mathbb{R}^n, X\in\mathcal{S}^n} \quad & \mathrm{tr}(XP_0) + q_0 x + r_0 \\ \text{s.t.} \quad & \mathrm{tr}(XP_i) + q_i x + r_i \leq 0, \ i = 1, \ldots, m. \\ & \begin{bmatrix} X & x' \\ x & 1 \end{bmatrix} \succeq 0, \end{aligned} \tag{B.9}$$

which is an instance of SDP described in Section B.2.1. This procedure describes the essence of SDR. The optimal value of the SDR gives a lower bound on the optimal value of the original non convex problem. Since non convex QCQP formulation of many hard combinatorial problems is an instance of this general QCQP form, SDR in Equation B.9 has been directly applied to non convex QCQP's of non-convex problems. For example, it is easy to see that SDP relaxation of the non convex QCQP of MAX-CUT problem given in Equation B.7 is the following SDP

$$\begin{aligned} \max_{X} \quad & \mathrm{tr}(X\tilde{W}) \\ \text{s.t.} \quad & X_{ii} = 1, \quad i = 1, \ldots, n. \\ & X \succeq 0 \end{aligned} \tag{B.10}$$

where the x disappears from MAX-CUT SDR.

B.2.2 Recovering a Solution from a SDR

Although the using SDR seems an natural and straightforward step to obtain a convex approximation for some nonconvex problems, the main bottleneck arises after finding the global minimum of SDR problem. Even though the optimum objective value of the SDR problem gives a lower bound for the original non convex problem, recovering a good feasible solution for the original problem from the global optimum X^* of the SDR problem is not trivial and indeed is the most crucial step of convex approximations (d'Aspremont & Boyd, 2003; Luo et al., 2010). We have

seen that SDR formulation is reached by dropping the rank constraint. Therefore, if X^* is of rank one, then the relaxation is tight which means x_0^* where $X^* = x_0^*(x_0^*)'$ is also optimal for the original problem. However, if X^* has rank more than one, then more work needs to be done to get some good feasible solutions of the original problem. An intuitively reasonable way to get such feasible solutions has been to use best rank-one approximations. One can show that the best rank one approximation of X^* with respect to 2-norm is $X_1^* = \lambda_1 q_1 q_1'$ where λ_1 is the maximum eigenvalue of X^* and q_1 is corresponding eigenvector (Luo et al., 2010). We could simply extract an approximation for x_0^* by $\tilde{x}_0 = \sqrt{\lambda_1 q_1}$ if it was feasible, however in general, $\tilde{x}_0$ may not be feasible. One possibility then is to "project" $\tilde{x}_0$ to the nearest feasible point using $\tilde{x}_0 = sign(\tilde{x}_0)$. The vector $\tilde{x}_0$ then is the *eigenvector approximation* (Luo et al., 2010).

An appealing alternative is to use *randomization* (d'Aspremont & Boyd, 2003; Luo et al., 2010). Recall that the last constraint in SDR in Equation B.9 is equivalent to $X - xx' \succeq 0$ by using Schur complement and then we can define Gaussian random variable $\tilde{x}$ with distribution $\tilde{x} \sim \mathcal{N}(x^*, X^* - x^*(x^*)')$ where X^* and $x*$ are solutions of SDR. The original non-convex QCQP can be solved in expectation with the following stochastic QCQP,

$$\begin{aligned} \min_{X - xx' \succeq 0} \quad & E_{\tilde{x} \sim \mathcal{N}(x, X - xx')}[\tilde{x}' P_0 \tilde{x} + q_0 \tilde{x} + r_0] \\ \text{s.t.} \quad & E_{\tilde{x} \sim \mathcal{N}(x, X - xx')}[\tilde{x}' P_i \tilde{x} + q_i \tilde{x} + r_i] \leq 0.\ i = 1, \ldots, m. \end{aligned} \tag{B.11}$$

Knowing the fact that $X - xx' = E_{\tilde{x} \sim \mathcal{N}(x, X - xx')}[\tilde{x}\tilde{x}'] - xx'$ and $x = E_{\tilde{x} \sim \mathcal{N}(x, X - xx')}[\tilde{x}]$ implies $X = E_{\tilde{x} \sim \mathcal{N}(x, X - xx')}[\tilde{x}\tilde{x}']$. This clearly shows the equivalence of stochastic QCQP to SDR which eventually allows an intuitive randomization procedure as follows. First we solve SDR and obtain X^*. Then sample solution candidates $\tilde{x}$ from the distribution $\mathcal{N}(x, X - xx')$. However, not all random samples can be feasible. A rounding procedure that is specific to problem structure can be applied in this case. For example, we can obtain discrete points from samples by simply using $\tilde{x} = \text{sign}(\tilde{x})$ (Luo et al., 2010). This stochastic setting allows a convenient

way to obtain a theoretical results on the accuracy of the SDR (e.g. bounds on the gap between the original problem and approximation) (Luo et al., 2010; Goemans & Williamson, 1995).

As a matter of course, SDR was first studied extensively in optimization field for a long time. SDR had its first significant impact in Signal Processing field long before sparking great interest in ML community (Luo et al., 2010). The idea of SDR has first mentioned in 1979 by Lovász (Lovász, 1979) and Nesterov et al. (2000) states that first SDR was proposed by Lovász & Schrijver (1991). However, the importance of SDR was not recognized until a well-known work by Goemans & Williamson (1995) which they have first shown the very promising approximation quality of SDR for MAX-CUT problem by using the randomization technique described above. More specifically, they proved that SDR can result in an objective value that is at least 0.8756 times the optimal value of original problem which was a clear evidence of the potential of high impact of SDR on dealing with NP hard problems by providing accurate approximations to them. In fact, this novel analysis of SDR stimulated a significant number of theoretical extensions for approximation quality analysis for different problems (even some analysis proving exact approximation of SDR under some conditions) (Luo et al., 2010). However elaborating the technical details of approximation guarantee proofs is beyond the scope of this thesis.

B.2.3 SDR in Machine Learning

SDR techniques were frequently applied to different problems in ML since the first related seminal work of Lanckriet et al. (2004) that quickly stimulated many other SDR applications in ML. Lanckriet et al. (2004) proposed a relaxation that learns the kernel matrix for two-class SVM. Bie & Cristianini (2003) developed a convex relaxation for transductive two-class SVM. Xu et al. (2004) used SDR for clustering that favors maximum margin hyperplanes for both unsupervised and semisuper-

vised setting. Xu & Schuurmans (2005) extended the convex SVM contributions of Xu et al. (2004); Bie & Cristianini (2003) to multi-class SVMs for both unsupervised and semisupervised setting. Lanckriet et al. (2007) et al. differently applied SDR to a dimension reduction problem, particularly to sparse principal component analysis (SPCA) which was proposed by Zou et al. (2006) and proposed a new convex version called direct SPCA(DSPCA) .

The early appealing progress in convex relaxations in ML has been very influential and opened new avenues to attack many other well known challenging nonconvex problems in ML. Guo & Schuurmans (2007) has sharply applied a novel SDR technique to a non-convex EM based latent variable model which was very inspirational for other works. Joulin & Bach (2012) extends this work to another convex relaxation with a soft-max loss for semisupervised setting. Argyriou et al. (2008) proposed a novel convex relaxation for learning representations that are common across many tasks in a multi-task setting. Zhang et al. (2011) developed convex formulations of representation learning problems such as sparse coding and dimension reduction. Cheng et al. (2013) presented convex relaxations for Bregman divergence clustering. Unfortunately, none of these results can handle predictive latent models with one hidden layer and this problem will be the focus of this thesis. Yu et al. (2010) propose convex approximation for bounded loss minimization without any result for robustness for the final estimator and any consistency result. In this thesis, we also develop a novel convex approximation for robust estimators which significantly improves the previous work.

B.2.4 SDP Solvers

Nowadays, ML applications require learning algorithms with less computational effort since dimensions of datasets are usually very large. Hence, it is critical to develop highly scalable algorithms.

SDP's sparked great interest after Nesterov and Nimirovskii had proposed first

high-impact algorithmic techniques to find solution of SDP's with polynomial runtime guarantees by extending *interior point* (or *barrier*) methods used for LP's and QP's to SDP's (Nesterov & Nimirovskii, 1994; Boyd & Vandenberghe, 2004). Interior point methods use a barrier function that is convex and ensures the inequality constraint is satisfied. This function is moved to the objective function to eliminate the inequality constraints by putting an appropriate Lagrangian coefficient in front of the barrier function. The well-known barrier function for the semidefinite constraint $X \succeq 0$ is $-\log(\det(X))$ which is convex. It has been used very frequently because it has the following nice and intuitive property. Semidefinite constraint is satisfied if and only if X has eigenvalues all positive (Strang, 1996). Recall that $-\log(\det(X)) = \sum_{i=1}^{n} \log(1/\lambda_i)$ where λ_i's are eigenvalues of X. Thus if one of the eigenvalues goes to zero, the barrier function goes to infinity, preventing the violation of the semidefinite constraint.

The most popular SDP toolboxes such as SeDuMi, SDPT3, CVX and MOSEK use interior point methods (Sturm, 1999; Toh et al., 1999; Grant & Boyd, 2014, 2008; Andersen & Andersen, 2000). Hence, they give solutions with high accuracy. On the other hand, interior point methods are second order methods and their complexity is a low-order polynomial of problem size which does not allow it to scale beyond small size problems. *first order methods* that tackled scaling bottleneck of SDP solvers even sometimes in the expense of less accurate solutions. Hence, first order methods was preferred over second order methods because of having low complexity per iteration.

As a matter of fact, many strategies that attempted to scale better is trading the high accuracy for better running times. The first-order methods proposed by Nesterov (2007), Nemirovski (2007) and Sanjeev et al. (2005) that adopt *proximal methods* to SDP's are well-known examples that fall into this category. A randomized version of the method proposed by Sanjeev et al. (2005) was developed (Garber & Hazan, 2011).

Another line of first order methods is *spectral bundle method* that exploited the non-negativeness of the smallest eigenvalue of the variable matrix (Helmberg & Rendl, 2000). Using the fact that all SDP problems can be written as a maximum eigenvalue problem given in Helmberg & Rendl (2000), a smooth stochastic algorithm that solves a smooth approximation to the eigenvalue problem achieving improvement in efficiency (d'Aspremont & Karoui, 2014). Another interesting advance in first-order methods was established with a key transformation trick discovered to turn the original problem to an equivalent saddle-point eigenvalue problem (Renegar, 2014) .

There are also other first order approaches applied to SDP e.g. the so called *Augmented Lagrangian based methods* (ADMM) and *block coordinate descent methods*. The ADMM based methods are usually computationally expensive because full eigen–decomposition is required in each iteration (Wen et al., 2010). Block coordinate descent based methods was also applied to SDP's by using Schur complements characterization of PSD constraints for updates on one row or column on solving subproblems (Wen et al., 2012). However, space complexity of $\mathcal{O}(n^2)$ makes it prohibitive for applying on large-scale problems. Another block coordinate descent based method scales to millions of dimensions for positive definite constraints by exploiting problem structure for a sparse inverse covariance selection (Hsieh et al., 2013).

Another category of first order methods which was prominent is *Frank Wolfe* (FW) algorithm which is also known as *Conditional Gradient* (CG) algorithm to solve SDP's (Frank & Wolfe, 1956; Hazan, 2008; Laue, 2012). FW was first introduced in a seminal paper by Frank and Wolfe for minimization of smooth convex function with respect to a convex set in 1956 (Frank & Wolfe, 1956). An intuitive illustration of the FW iteration taken from (Jaggi, 2013) is shown in the Figure B.3 . The need for expensive projections to feasible set in each iteration was considered as the main drawback of gradient descent methods. In each iteration, the vanilla

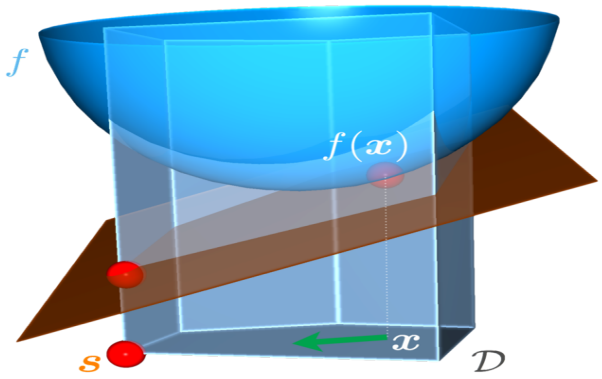


Figure B.3: A FW iteration illustration.

FW algorithm finds the direction to move by minimizing a linear approximation to the original objective function over the constraint set of interest. Hence, FW is projection-free compared to gradient descent methods and this made it attractive for large scale constrained convex optimization. On the other hand, FW has convergence rate on the order of $1/t$ and it's convergence rate is slow compared gradient descent methods and to some other first order methods such as Nesterovs accelerated gradient which has a general convergence rate of $1/t^2$ (Nesterov, 1983). However, it was shown that vanilla FW can converge at the rate $1/t^2$ for the special case of the strongly convex function and also strongly convex feasible sets (Garber & Hazan, 2014).

FW was first adopted for SDP by Hazan (2008) where the algorithm is crucially producing sparse solutions. Sparse solutions mean low rank solutions in case of SDP. The low rank solution is critical since it has advantages in terms of storage and computation. By applying the factorization $X = VV'$ where $V \in \mathbb{R}^{n \times r}$ and $r \leq n$, we can store V instead of X which saves much space for large dimensions. Multiplying X with a vector becomes also much cheaper when V is used and this can significantly improve running time. Hazan (2008) starts with a column vector for V and incremental rank-1 updates. However, the main drawback here is slow

convergence rate of FW. There are also nonlinear techniques that aim to take the advantage of low rank by optimizing directly over V. However the problem becomes no longer convex in V. Burer & Monteiro (2003) proposed such a method without a convergence guarantee to global solution trades the theoretical guarantees for practical efficiency.

The low rank approach in Burer & Monteiro (2003) sparked significant interest since it also provided practically efficient results for the SDP problems that do not have strict accuracy requirements, allowing one to enjoy the efficiency of a nonlinear non-convex approach (Kulis et al., 2007; McCoy & Tropp, 2011). This also opened a line of research that attempted to provide the analysis and conditions for stationary points that are globally optimal for this particular regime. First, Burer & Monteiro (2003) introduced a sufficient condition for global optimality in their original work, which was then extended by a crucial analysis of the necessary condition in (Grippo et al., 2008). Journée et al. (2010) also provided necessary and sufficient conditions for stationary points to be global minima under specific conditions. Their work was extended by developing an algorithm called *SpeeDP* that solves very large MAX-CUT problems by exploiting problem specific features of MAX-CUT (Grippo et al., 2012). In addition, some specific problems with a suitable structure were shown to give global optimum guarantees for this low rank approach (Jain et al., 2013; Netrapalli et al., 2013, 2014; Feng et al., 2013). For example, in case of matrix completion, it is already assumed that data is low rank and that a property called *restricted isometry property (RIP)* was satisfied. Under these conditions, one can show the convergence guarantee to global minimum (Jain et al., 2013).

It was explained above that the FW based method proposed by Hazan (2008) has theoretical guarantees whereas it does not offer practical efficiency. In contrast, nonlinear low rank approach of (Burer & Monteiro, 2003) does not have theoretical guarantees but practically efficient. A hybrid algorithm that crucially combines the

parts that are advantageous from two methods by adding a nonlinear local step in FW (Laue, 2012). This hybrid algorithm empirically improves performance of FW without sacrificing convergence guarantees for global optimum solution.

B.3 Deep Nonlinear Models

A deep model with fully connected layers is shown in Figure 1.2. The hidden layers Φ_i are calculated by nested activation functions

$$\phi_i(x) = \sigma_i(\phi_{i-1}(x)W_i). \tag{B.12}$$

where x is an instance of data, $i = 2, 3, \ldots, l$, l is the number of hidden layers, $\phi_1(x) = \sigma_1(xW_1)$. Mostly the same activation function is used in all layers. Some of the commonly used non-linear transfer functions are the following

$$\begin{aligned} \text{Sigmoid} &: \frac{1}{1+\exp(-x)} \\ \text{ReLu} &: \max(0, x) \\ \text{softplus} &: \log(1+\exp(x)) \\ \tanh &: \frac{\sinh(x)}{\cosh(x)} \end{aligned} \tag{B.13}$$

and they are shown in Figure B.4. The prediction function is calculated by softmax applied to the last hidden layer's values

$$f(x) = \frac{\exp(\phi_l(x)'W)}{\sum \exp(\phi_l(x)'W)} \tag{B.14}$$

The arguments of loss function are the prediction that is calculated by the output of last layer and the original output. The goal of the learning algorithm is to minimize an objective function that combines a classification loss and regularizer with respect to weights. Cross-entropy is the most common among other losses in deep learning and Tikhonov regularizer is the common regularizer which is known as *weight decay* in deep learning (Goodfellow et al., 2015). Although the objective

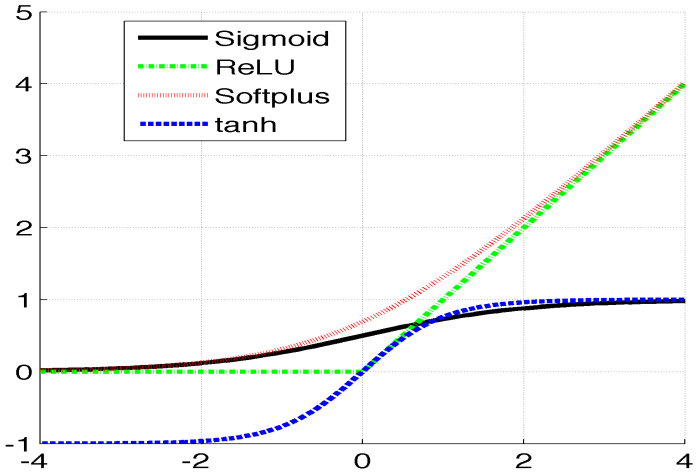


Figure B.4: Some well known activation functions for hidden layers.

function is convex at the last layer for the fully connected network in Figure 1.2, it is not jointly convex over all weights because of nested nonlinearities through layers that appears in loss function (Gori & Tesi, 1992).

B.4 Robust Models

A general approach to tackle sensitivity to outliers in robust statistics is called M-estimation (Zhang, 1997). M-estimation is essentially the minimization of sum of functions of the residual with respect to parameters and has the form

$$\hat{\boldsymbol{\theta}}_M \in \arg\min_{\boldsymbol{\theta}} \tfrac{1}{N} \textstyle\sum_{i=1}^{N} \rho(r_i) \tag{B.15}$$

where ρ is a symmetric, positive-definite function, $r_i = \mathbf{y}_i - \mathbf{x}_i\boldsymbol{\theta}$ is the residual, $\boldsymbol{\theta} \in \mathbb{R}^p$. Then estimator $\boldsymbol{\theta}$ is solution to the following p equations

$$\sum_{i=1}^{N} \psi(r_i) \frac{\partial r_i}{\partial \boldsymbol{\theta}_m}, \quad \text{for } m = 1, \ldots, p \tag{B.16}$$

where $\psi(r_i) = \frac{d\rho(r_i)}{dr_i}$ is the *influence* function. Influence function measures the sensitivity of the estimate with respect to residual. Ideal ρ should have a bounded influence function. Some M-estimators proposed in robust statistics and computer vision that have bounded influence function are listed in Figure B.5 and shown in Figure B.6 (Zhang, 1997). It can be observed that these losses have more resistance to outliers. For example, Huber and Geman-McClure are well known among them: Huber is piecewise function that changes a quadratic function for small residuals and linear for large residuals therefore is has some resistance to outliers. Huber is also convex which makes it tractable. Geman-McClure is a bounded estimator however it is not convex.

Ideally, we want the predictor to be both bounded and tractable. However, there is a fundamental dilemma that we prove in Chapter 5 for losses: convex losses in literature cannot be robust to outliers whereas bounded losses cannot be tractable.

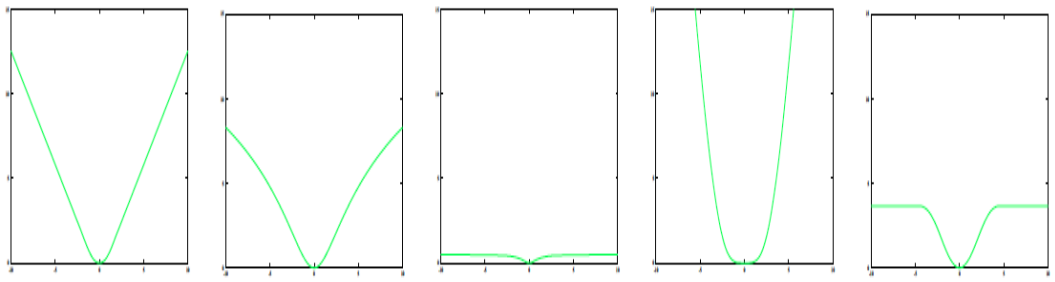
	ρ
Huber	$\begin{cases} \frac{(r)^2}{2} & \text{for } \lvert r\rvert \leq \delta, \\ \delta(\lvert r\rvert - \frac{\delta}{2}) & \text{otherwise.} \end{cases}$
Cauchy	$\frac{r}{(1+(r/\delta)^2)^2}$
Geman-McClure	$\frac{c^2}{2}\log(1+r^2)$
Welsch	$\frac{\delta^2}{2}[1-\exp(-(x/\delta)^2)]$
Tukey	$\begin{cases} \frac{\delta^2}{6}(1-[1-(r/\delta)^2]^3) & \text{for } \lvert r\rvert \leq \delta, \\ \frac{\delta^2}{6} & \text{otherwise.} \end{cases}$

(a) ρ-functions

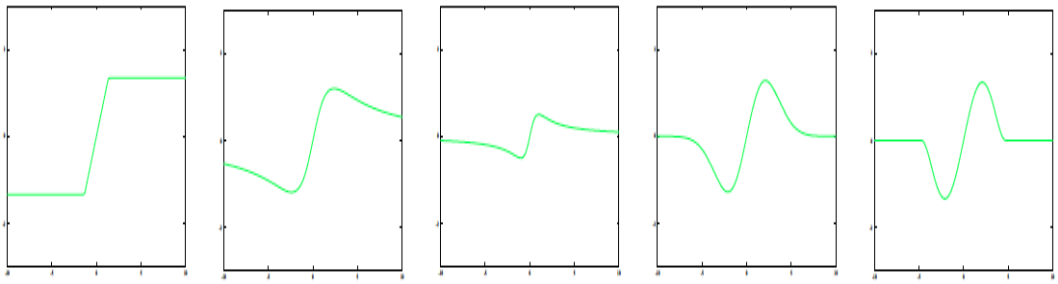
	ψ
Huber	$\begin{cases} r_i & \text{for } \lvert r\rvert \leq \delta, \\ \delta\text{sign}(r) & \text{otherwise.} \end{cases}$
Cauchy	$\frac{r}{(1+(r/\delta)^2)^2}$
Geman-McClure	$\frac{r}{(1+r^2)^2}$
Welsch	$x\exp(-(x/\delta)^2)$
Tukey	$\begin{cases} r[1-(r/\delta)^2]^2 & \text{for } \lvert r\rvert \leq \delta, \\ 0 & \text{otherwise.} \end{cases}$

(b) ψ influence functions

Figure B.5: The formulations of the losses and the influence functions for some M-estimators.



(a) ρ-functions



(b) ψ influence functions

Figure B.6: The figures of Huber, Cauchy, Geman-McClure, Welsch and Turkey from left to right.