BILEVEL PROGRAMMING ALGORITHMS FOR MACHINE LEARNING MODEL SELECTION

By

Gregory M. Moore

A Thesis Submitted to the Graduate

Faculty of Rensselaer Polytechnic Institute

in Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Major Subject: MATHEMATICS

Approved by the
Examining Committee:

Kristin P. Bennett, Thesis Adviser

Sanmay Das, Member

Joseph G. Ecker, Member

John E. Mitchell, Member

Rensselaer Polytechnic Institute
Troy, New York

September 2010
(For Graduation December 2010)

CONTENTS

LIST OF TABLES . . . iv

LIST OF FIGURES . . . v

ACKNOWLEDGMENT . . . vii

ABSTRACT . . . viii

1. INTRODUCTION . . . 1
   - 1.1 Thesis structure . . . 2
   - 1.2 Notation and definitions . . . 3

2. BACKGROUND . . . 7
   - 2.1 Learning models . . . 7
     - 2.1.1 Hard margin classification . . . 8
     - 2.1.2 Support vector machines . . . 9
     - 2.1.3 Ridge Regression . . . 12
     - 2.1.4 ϵ-insensitive SVR . . . 14
     - 2.1.5 Algorithms for solving SVM-type problems . . . 15
       - 2.1.5.1 Cutting plane algorithm . . . 16
   - 2.2 Cross-validation . . . 17
     - 2.2.1 Leave-one-out CV . . . 20
     - 2.2.2 Bilevel programming . . . 21
     - 2.2.3 A bilevel formulation for cross-validation . . . 22
   - 2.3 Implicit methods for solving bilevel CV . . . 23
     - 2.3.1 Grid search . . . 25
     - 2.3.2 Prior knowledge . . . 25
     - 2.3.3 Pattern search . . . 26
     - 2.3.4 Implicit gradient descent . . . 26
     - 2.3.5 Directional derivative of implicit problem . . . 29
   - 2.4 Explicit algorithms for solving bilevel CV . . . 31
   - 2.5 Other model selection algorithms for solving bilevel CV . . . 32
     - 2.5.1 Bayesian model selection . . . 32
     - 2.5.2 Generalized cross-validation . . . 33
     - 2.5.3 Path following . . . 33

3. PENALIZED BILEVEL FORMULATION 35
3.1 Nonsmooth penalty bilevel problem 35
3.2 Approximation algorithm with proximity control 38
3.2.1 Optimality and subgradient selection 41
3.3 Convergence of PBP 45
3.3.1 Strict decrease in stability centers 45
3.3.2 Convergence theorem 46
3.4 Properties of PBP 48
3.5 Model selection for ridge regression via the PBP algorithm 48
4. IMPLICIT BILEVEL PROPERTIES 51
4.1 Properties of the implicit problem 52
4.2 Solving twice-differentiable training functions 54
4.3 Solving once-differentiable training functions 56
4.4 Solving nondifferentiable training functions 59
4.5 Nonuniqueness of the implicit problem 63
5. IMPLICIT GRADIENT-BASED BILEVEL ALGORITHMS 64
5.1 The `ImpGrad` algorithm 65
5.2 The `ImpBundle` algorithm 66
5.2.1 Nonsmooth nonconvex bundle algorithm 66
5.2.2 Selection of directional derivative 68
6. LEAST-SQUARES SUPPORT VECTOR REGRESSION WITH MULTIPLE GROUPS 72
6.0.3 Implicit methods: `ImpGrad` and `ImpBundle` 73
6.0.4 The `PBP` algorithm 75
7. EXPERIMENTAL RESULTS 79
7.1 Experiment A: Small QSAR datasets 79
7.2 Experiment B: Large QSAR datasets 81
7.2.1 Hit identification task 82
7.2.2 Experimental design 84
7.2.3 Results 85
8. CONCLUSIONS 90

LIST OF TABLES

7.1 Small QSAR dataset properties 79

7.2 Large QHTS dataset properties 83

LIST OF FIGURES

2.1 Hard margin classification 8

2.2 Soft margin classification 9

2.3 Linear hinge loss function 10

2.4 Ridge regression 12

2.5 ϵ-insensitive regression 14

2.6 Cutting plane algorithm *cuts* 16

2.7 Dividing data for cross-validation 18

2.8 Smoothed hinge loss function 19

2.9 The bilevel cross-validation problem. 21

2.10 A sample of grid points 24

2.11 Performance of grid search, ImpGrad, and PBP algorithms 30

3.1 The local approximation function 38

4.1 Nonsmooth classification 53

5.1 Nonconvex classification 64

7.1 Generalization error on QSAR datasets 81

7.2 Computational time on QSAR datasets 82

7.3 Generalization error on pyruvate kinase dataset 86

7.4 Generalization error on tau-fibril dataset 87

7.5 Scalability plot in modeling set size 88

7.6 Hyperparameter scalability plot 89

ACKNOWLEDGMENT

I would like to thank all those who have helped me make it where I am today. From my family, I would like to thank my parents, whom without your support and encouragement I would not be where I am today. In addition to my parents, I would like to acknowledge the remainder of my family who have played a critical role in supporting me. Finally from my family, Leila thank you for supporting me in good times and bad. Your support and encouragement helped tremendously, thank you.

My time at RPI would have been very different if it were not for my fellow graduate students. Thank you all for enriching my life both academically and socially. In particular of my fellow students, Charles, I do now know where I would be without your help and daily “thoughts”. Thank you for making my time here enjoyable academically and socially.

I would like thank the Department of Mathematical Sciences staff. Your help throughout my stay at RPI was much appreciated, especially from you Dawnmarie. Lastly I like to thank my professors, particularly my committee members. Your mentoring was greatly appreciated. I would like to conclude by acknowledging my advisor, Kristin Bennett. Your consistent, but realistic, pressure has tested me in many ways, all to develop the mathematician I am today. I truly enjoyed working with you and will miss your unique approach to research.

ABSTRACT

This thesis represents an advance towards the goal of self-tuning supervised data mining as well as a significant innovation in scalable bilevel programming algorithms. Novel nonsmooth bilevel programming methods for training linear learning models with hyperparameters optimized via T-fold cross-validation (CV) are considered. Determining the optimal values of these hyperparameters is the goal of *model selection*. Three algorithms are developed and tested against previous algorithms: `ImpGrad`, `ImpBundle` and `PBP`. Also a new generalized version of ϵ-insensitive regression is developed. This new modeling task, called multiSVR, allows use of data sets with varying levels of quality. Two data sets are used to test this new 10-hyperparameter model.

Current model selection practice constructs models over a predefined grid of hyperparameter combinations and selects the best one, an inefficient heuristic algorithm. The proposed methods formulate the model selection CV problem as a bilevel problem. A bilevel problem is an optimization problem with constraints that are also an optimization program. The overall objective is referred to as the outer-level objective, and the inner optimization constraints are referred to as the inner-level problem. The bilevel framework simultaneously solves for the model hyperparameters (outer-level) and training weights (inner-level). The new methods address hyperparameter selection for linear support vector machines (SVM) and related machine learning problems.

The proposed approaches are the first to address the hyperparameter selection of SVM models expressed as unconstrained (potentially nonsmooth) optimization problems. Applying nonsmooth mathematical programming methods to nonsmooth linear SVM models has produced the fastest training algorithms to date. The bilevel programming approaches to hyperparameter optimization can produce two types of algorithms: explicit and implicit. Explicit algorithms simultaneously optimize the model weights and hyperparameters by replacing the inner-level problems by their optimality conditions. The proposed explicit algorithm `PBP` utilizes the optimality conditions in nonsmooth form, contrasting with prior less scalable approaches that formulate the problem as a mathematical program with equilibrium constraints (MPEC). Implicit algorithms optimize only the hyperparameters by treating the model weights as implicit functions of the hyperparameters. The implicit algorithms work in a low dimensional space at the expense of having a very challenging nonsmooth nonconvex objective function. The proposed implicit algorithms work with the primal unconstrained SVM problem in contrast with prior implicit methods that use the smooth dual SVM problem.

The first proposed explicit approach, a penalized bilevel program (`PBP`) algorithm, treats the lower-level problems as unconstrained optimization problems that are replaced with their optimality conditions. The key innovation is that the inner-level problem can be replaced with a nonsmooth nonlinear constraints without introducing extra variables or constraints. Previous methods only used the smoothed primal or dual formulations which results in nonlinear optimality conditions that grow exponentially in the sample size. A novel bilevel programming algorithm to solve this class of problems is developed by penalizing the nonsmooth optimality condition and then solving the resulting problem with an approximation algorithm. Convergence analysis of the algorithm shows `PBP` converges to a solution satisfying the necessary optimality conditions of the penalized bilevel program.

The `ImpGrad` and `ImpBundle` algorithms are implicit gradient-based approaches. Similar to `PBP`, these algorithms treat the lower-level problems as unconstrained optimization problems, unlike the previous implicit methods which use the dual training formulation. This yields a linear optimality condition from which the gradient of the training problem can be computed. Similar to the previous implicit gradient-based algorithm, `ImpGrad` assumes that the bilevel program is locally smooth, and selects an arbitrary subgradient. `ImpBundle` expands on `ImpGrad` by not assuming smoothness. This improved algorithm uses directional derivatives in the search direction of interest to ensure that the subgradient chosen is not only valid, but also informative. This prevents searching in assent directions.

A new outer algorithm is developed to solve the `ImpBundle` algorithm. Rather than use the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method of `ImpGrad` and previous implicit gradient-based approaches, this method uses a novel nonsmooth nonconvex bundle method that allows for bounds constraints. This algorithm explicitly deals with the nonsmoothness and nonconvexity of the model selection problem.

Computational results compare `PBP`, `ImpGrad` and `ImpBundle` against previous bilevel algorithms. Results on cheminformatics problems are presented for the traditional ϵ-insensitive regression problem and a new multiSVR model with 10 hyperparameters. Model selection of the later problem is beyond the ability of most algorithms to solve efficiently. In fact a thorough grid search on this problem would require roughly 250 million training problems to be solved. The proposed implicit and algorithms are shown to scale well in the sample size and number of hyperparameters. Experimental results show than the proposed algorithms outperform grid search and prior smooth bilevel CV methods in terms of modeling performance and computational time. PBP produces the best overall generalization results but with larger computational times that the implicit algorithms. This increased speed permits modeling with an increased number of hyperparameters on massive datasets. The implicit algorithms are faster and applicable to a larger class of problems than PBP. But `ImpBundle` and particularly `ImpGrad` yielded slightly less robust generalization results on a few problems.

CHAPTER 1
INTRODUCTION

Machine learning (ML) focuses on methods and algorithms for computers to learn complex and hidden structures from large data. Successful and popular methods include support vector machines [43] and kernel methods [39]. For the purpose of this work, the learning models construct a linear function that minimizes a regularized convex loss function. By changing the loss function, this learning model can address many learning problems including regression, classification, novelty detection and ranking. The resulting optimization problem is convex, but typically involves *hyperparameters* ($\boldsymbol{\gamma}$) that must be selected by the user. The focus of this thesis will be choosing these important hyperparameters; this task is also known as *model selection*. The selected hyperparameters determine model accuracy and robustness.

This thesis (1) develops novel nonsmooth bilevel programming cross-validation (CV) algorithms applicable to a wide variety of existing and novel machine learning problems and (2) investigates novel multi-parametric learning models enabled by the new bilevel CV paradigm.

For example, in support vector regression (SVR), the training problem solves for the predictive hyperplane parameters $(\mathbf{w}, b)$. The model selection problem solves the tradeoff hyperparameter C (that determines the relative weight of the empirical risk and structural risk terms) and the insensitivity hyperparameter ϵ (half-width of losses' unpenalized tube). Typically, this problem of selecting the hyperparameters is left to the user, and a policy-based approach or a simple grid strategy is used. This work focuses on novel bilevel programming methods for machine learning that solve for both the model selection hyperparameters $(C,\ \epsilon)$ and the training parameters $(\mathbf{w},\ b)$.

A bilevel programming paradigm that can solve real-world problems will allow novel and more accurate machine learning models and enable people from many disciplines to utilize powerful machine learning techniques. Learning from data is a powerful technique for solving huge varieties of pattern recognition tasks such as target identification, drug development, image recognition, sound recognition, robotics and prediction tasks.

A popular method for model selection is CV grid search of hyperparameters along with expert choices. An expert user commonly makes model selection choices from experience, creates a discrete grid of the possible remaining hyperparameters, and then trains and validates a model at each grid point. Unfortunately, this is not efficient and there is no guarantee of solution quality. Non-experts may skip or skimp on model selection, producing models with suboptimal generalization. The bilevel formulation largely removes the expert user from the problem and efficiently solves for model parameters and hyperparameters simultaneously.

1.1 Thesis structure

The remainder of the thesis is outlined as follows: First definitions and notation used throughout this thisis are given. Following this, Chapter 2 presents background information used throughout this thesis. The background material is separated into three sections. The first section presents learning models, including support vector machines, the model selection problem, and concludes with algorithms to solve for the training parameters. The second section presents cross-validation as a framework to solve the model selection problem. This chapter also details the bilevel framework and how model selection can be formulated as a bilevel program via CV. The final background section concludes with a description of previous algorithms that solve the model section problem. This section is further broken into implicit methods, explicit methods, and algorithms designed to solve problems with one or two hyperparameters. Each class of algorithms has advantages and disadvantages that will be discussed, and for each class relevant algorithms will be presented.

Chapter 3 presents an algorithm to solve the bilevel model selection problem. This algorithm transforms the bilevel program by replacing the unconstrained lower level program with its optimality condition. This optimality condition is then penalized and forms the basis for this penalized bilevel programming (PBP) method. An approximation algorithm is then developed to solve this PBP problem. This method for solving the model selection problem is refered to as the `PBP` algorithm. This chapter concludes by presenting the model selection algorithm for the ridge regression problem.

Chapter 4 explores the mathematical challenges of the implicit gradient descent algorithm proposed by Keerthi et al.[25]. Unlike the previous algorithm, no assumptions are made about the differentiability of the problem. First the nonsmooth nature of this implicit gradient algorithm is examined in detail for twice-differentiable training functions, once-differentiable functions and nondifferentiable functions. Following this, Chapter 5 then discusses the nonconvexity properties of the implicit bilevel problem. Finally this chapter develops two nonsmooth, nonconvex algorithms with linear constraints (`ImpGrad` and `ImpBundle`) to solve the model selection problem.

Chapter 6 introduces and implements a new generalized version of ϵ-insensitive regression where the data is clustered into multiple groups. Implementation details for the implicit gradient descent methods and PBP methods are given. Chapter 7 details the datasets used to compare the `ImpGrad`, `ImpBundle` and `PBP` algorithms against previous algorithms and then gives accompanying results. Chapter 8 serves as a conclusion.

1.2 Notation and definitions

Let bolded $\mathbf{x}$ be a column vector. The transpose of vector $\mathbf{x}$ is denoted $\mathbf{x}'$. Let $\mathbf{e}$ and $\mathbf{0}$ be the vectors of ones and zeros of appropriate size. Let I be the identity matrix of appropriate size. We define the component-wise maximum of the vector $\mathbf{x}$ and $\mathbf{0}$ as:

$$\mathbf{x}_+ := \max(\mathbf{x}, \mathbf{0}). \tag{1.1}$$

If $\mathbf{x}$ and $\mathbf{y}$ satisfy $\mathbf{x}'\mathbf{y} = 0$, then we say $\mathbf{x}$ is perpendicular to $\mathbf{y}$, or $\mathbf{x} \perp \mathbf{y}$. Further, we define:

$$0 \leq \mathbf{x} \perp \mathbf{y} \geq 0 \tag{1.2}$$

as shorthand notation for the following three equations:

$$0 \le \mathbf{x},\ \mathbf{x} \perp \mathbf{y},\ \text{and } \mathbf{y} \ge 0. \tag{1.3}$$

Together these equations imply that $\mathbf{x}$ and $\mathbf{y}$ are non-negative and *complementary*. $\mathbf{x}$ and $\mathbf{y}$ are complementary if for each component j, $\mathbf{x}_j$ or $\mathbf{y}_j$ equals zero. This shorthand notation will be used within the constraint set of optimization problems as well as part of the Karush-Kuhn-Tucker (KKT) optimality conditions.

Let the set of vectors $\mathbf{x}_j$ for $j = 1 \ldots n$ be denoted $\mathcal{X}$. The convex hull of $\mathcal{X}$ is $\text{conv}(\mathcal{X})$.

We let $f(\mathbf{x})$ denote a function of vector $\mathbf{x}$ and we assume that all functions are locally Lipschitz functions and differentiable almost everywhere. If f is a smooth function, we define $\nabla f(\mathbf{x})$ to be the gradient of f at the point $\mathbf{x}$. It is important to note that if f is differentiable with gradient $\mathbf{g}$ at point $\mathbf{x}$, then direction $-\mathbf{g}$ yields a direction of descent or $\mathbf{x}$ satisfies the first order optimality conditions. This is easily seen using a first order Taylor series approximation:

$$f(\mathbf{x}+\mathbf{d}) = f(\mathbf{x}) + \nabla f(\mathbf{x})'\mathbf{d} + \mathrm{O}(\|\mathbf{d}\|^2) = f(\mathbf{x}) + \mathbf{g}'\mathbf{d} + \mathrm{O}(\|\mathbf{d}\|^2). \tag{1.4}$$

Let $\mathbf{d} = -\beta\mathbf{g}$, and assume that $\mathbf{g}$ does not satisfy the first order optimality condition ($\|\mathbf{g}\| \neq 0$). Then the Taylor series approximation is:

$$f(\mathbf{x}-\beta\mathbf{g}) = f(\mathbf{x}) - \beta\mathbf{g}'\mathbf{g} + \beta^2\,\mathrm{O}(\|\mathbf{g}\|^2) < f(\mathbf{x}), \quad \text{for } \beta \to 0^+. \tag{1.5}$$

If f is a nonsmooth function, the subdifferential of $f(\mathbf{x})$ at $\bar{\mathbf{x}}$ is defined by the set [37]:

$$\partial f(\bar{\mathbf{x}}) = \text{conv}(\mathbf{g} \in \mathbb{R}^n | \nabla f(\mathbf{x}) \to \mathbf{g}, \mathbf{x} \to \bar{\mathbf{x}}). \tag{1.6}$$

Let $\mathbf{g} \in \partial f(\bar{\mathbf{x}})$ be a subgradient of f at $\bar{\mathbf{x}}$. In the case of nonsmooth functions, the Taylor series approximation does not hold, and a given subgradient can lead to directions of ascent, not descent. For the remainder of the the thesis we use subgradient notation for both smooth and nonsmooth functions.

The directional derivative of function $f(\mathbf{x})$ in unit direction $\mathbf{d}$ at $\bar{\mathbf{x}}$ is defined as:

$$f'(\bar{\mathbf{x}};\mathbf{d}) = \lim_{\beta\rightarrow 0^+} \frac{f(\bar{\mathbf{x}}+\beta\mathbf{d}) - f(\bar{\mathbf{x}})}{\beta}. \tag{1.7}$$

The directional derivative gives the first order approximation of the the expected change of function value in given direction $\mathbf{d}$. This is important because if $f'(\bar{\mathbf{x}};\mathbf{d}) < 0$, then the direction $\mathbf{d}$ is a descent direction. For the special case of which $f(\mathbf{x})$ is differentiable, then:

$$f'(\bar{\mathbf{x}};\mathbf{d}) = \nabla f(\bar{\mathbf{x}})'\mathbf{d}. \tag{1.8}$$

Given an optimization problem:

$$\min_{\mathbf{x}} f(\mathbf{x}), \tag{1.9}$$

the set of optimal solutions $\mathbf{x}$ is denoted by

$$\mathbf{x} \in \operatorname*{argmin}_{\mathbf{x}} f(\mathbf{x}). \tag{1.10}$$

We let $|\mathbf{x}|$ be the vector valued absolute value function, that is it returns a vector containing the magnitude of each respective component of $\mathbf{x}$. The family of vector q-norms for $1 \leq q \leq \infty$ is defined as $\|\mathbf{x}\|_q = \sqrt[q]{\sum_{i=1}^{n} |\mathbf{x}_i|^q}$. When q is not given, the Euclidean norm ($q = 2$) is assumed.

We define an algorithm to be *linear-time* with respect to descriptor n, if the algorithm's complexity with respect to n is $\mathrm{O}(n)$. In practice this is significant because if n doubles then the computational time is expected to double as well. Algorithms that exhibit this scalability, particularity in the dataset size allow efficient modeling on ever-increasingly large datasets.

Consider a dataset used for learning. Throughout this thesis we will assume that we have ℓ data and response pairs, denoted $(\mathbf{x}_j, y_j)$ for $j = 1..\ell$. We define $\mathbf{x}_j \in \mathbb{R}^n$, the n-dimensional set of real numbers and $y \in \mathbb{R}$. If the learning task is classification, then y_j is binary such that $y_j \in \{-1, 1\}, j = 1 \ldots \ell$. We assume a linear learning function, that is $f(\mathbf{x}) = \mathbf{x}'\mathbf{w} - b$, where $\mathbf{w}$ is the normal vector of the hyperplane of interest and b is the offset from the origin of the hyperplane.

Finally, for convenience of notation, we define the sum of piecewise defined functions in the following manner:

$$f(\mathbf{x}_1, \ldots, \mathbf{x}_\ell) = \sum_{j=1}^{\ell} \begin{cases} 0 & \text{if } \mathbf{x}_j < 0 \\ 1 & \text{if } \mathbf{x}_j \geq 0. \end{cases} \tag{1.11}$$

With notation and definitions given, we now present the background material that forms a basis for this thesis.

CHAPTER 2
BACKGROUND

This chapter presents the context for this thesis, broken up into three sections. The first section describes separable classification. This leads to the more general classification model discussed next: support vector machines (SVM). Then regularized regression is introduced and the unconstrained, primal and dual SVM formulations are discussed. Following this, existing algorithms to solve the SVM formulation are given.

With SVM understood, the second section focuses on the selection of hyperparameters via the classical model selection criteria cross-validation (CV). The cross-validation criteria is then setup as a bilevel program.

Finally existing bilevel algorithms to solve CV are given, broken up further into three classes. The first class is implicit algorithms which alternate the training of models and selection of hyperparameters. The next class considered is explicit algorithms. These algorithms use the bilevel formulation of cross-validation and solve the problem directly without alternation. Finally other algorithms applicable only to one or two hyperparameters are given as the last class. These algorithms are not directly comparable to the above algorithms as they do apply to general model selection problems.

We begin with a background on formulating learning tasks classification and regression as optimization problems.

2.1 Learning models

Given a set of $(\mathbf{x}_j, y_j)$, $j = 1..\ell$, data pairs, where $y_j \in \{-1, 1\}$, the goal of modeling task classification is to find a hyperplane that separates, or nearly separates, the data into two half-spaces, depending on the given label y_j. In this thesis we consider the linear function $y \approx f(\mathbf{x}) = \text{sign}(\mathbf{x}'\mathbf{w} - b)$. There are two algorithms of interest used in classification. *Hard margin classification* is the simpler case and is strictly for datasets which are linearly separable. A general classification

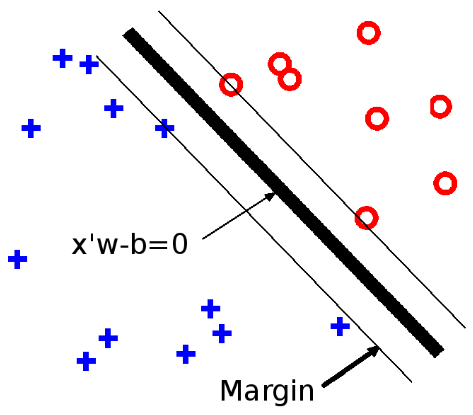

Figure 2.1: The separable classification problem. Using hard margin classification, a linear function (thick black line) is desired to exactly separate the blue plusses on the lower left from the red circles in the upper right. Hard margin classification maximizes the distance between two thin black lines, also called the *margin*.

model that allows misclassified points is referred to as a *soft margin classification*. We discuss each in turn.

2.1.1 Hard margin classification

The hard margin problem requires finding a hyperplane that correctly classifies *all* points, as shown in Figure 2.1. This hyperplane is not unique and the one that *maximizes the margin* is desired. The margin is the summed distance between the hyperplane and the nearest point in each class.

Maximizing the margin can be formulated [13] as an equivalent optimization problem that minimizes the norm of the normal vector of the hyperplane. This is a subtle change that later builds the foundation for support vector machines. The following simple optimization problem's objective is to minimize the normal vector of the hyperplane ($\|\mathbf{w}\|_2^2$) with constraints requiring all data in each half space share a common label ($y_j \in \{-1, 1\}$):

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \|\mathbf{w}\|_2^2 \\ \text{s.t.} \quad & y_j\left(\mathbf{x}_j'\mathbf{w} - b\right) \geq 1 \quad \text{for } j = 1..\ell. \end{aligned} \tag{2.1}$$

Unfortunately to train a model with this optimization problem the problem

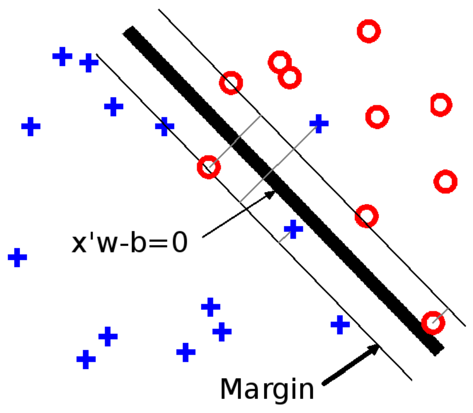

Figure 2.2: The soft margin classification problem. A linear function (black line) is desired to approximately separate the the blue plusses on the lower left with the red circles upper right while trading off model complexity. Here there are four points that given error, represented by the thin gray line. These points and those on the margin are called *support vectors*.

must be separable. Thus one issue with the hard margin model is determining whether a given dataset is known to be linearly separable *a priori*. The algorithm will not converge to a feasible point (as there are none) if the data are not linearly separable. Therefore a new framework was developed which would allow for non-separable data. *Soft margin classification* would solve this problem and form the basis of a new set of methods in machine learning: support vector machines (SVM).

2.1.2 Support vector machines

The support vector machine (SVM) framework provides a method to perform regularized classification, regression, ranking, and novelty detection. In particular, SVM allows classification to be performed on both separable and non-separable data, like in Figures 2.1 and 2.2 respectively. SVM is also known as *soft margin classification* [12].

The SVM classification goal is to find the hyperplane that minimizes the total distance from the respective margin to the misclassified (or nearly misclassified) points while not over fitting the data. Using optimization theory, SVM trades off model complexity, $\mathcal{S}(\mathbf{w}, b, \gamma)$, with training error, $\mathcal{L}_{trn}(\mathbf{w}, b, \gamma)$. Positive hyperpa-

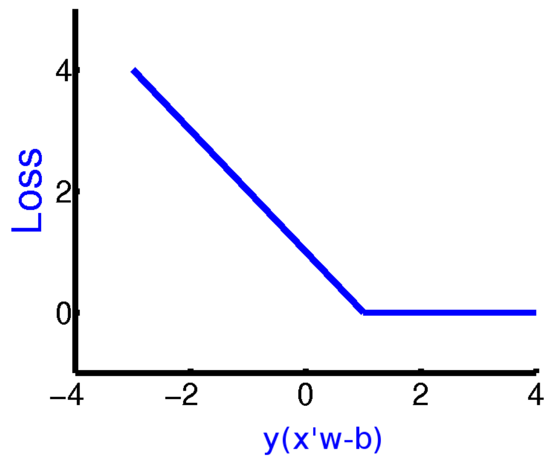

Figure 2.3: A linear hinge loss function for SVM. Penalizes points proportionally to the distance they are from the respective margin, but gives no penalty for those correctly classified with a value greater than 1.

rameter C controls this tradeoff. The general SVM optimization framework is:

$$\min_{\mathbf{w},b} \quad \mathcal{S}(\mathbf{w},b,\boldsymbol{\gamma}) + C\mathcal{L}_{trn}(\mathbf{w},b,\boldsymbol{\gamma}). \tag{2.2}$$

For soft margin classification, the margin regularization function is $\mathcal{S}(\mathbf{w},b,\boldsymbol{\gamma}) = \frac{1}{q}\|\mathbf{w}\|_q^q$, which is also used in the hard margin classification. The training error $\mathcal{L}_{trn}$ is softened to allow for misclassified points. In classical SVM, the training loss is a *hinge loss*, which as seen in Figure 2.3 penalizes points proportionally to the distance they are from their respective margin, but gives no penalty for those correctly classified with a value greater than 1. A standard choice is:

$$\mathcal{L}_{trn}(\mathbf{w},\boldsymbol{\gamma}) = \sum_{j=1}^{\ell} \min(0, 1 - y_j(\mathbf{x}_j'\mathbf{w} - b)). \tag{2.3}$$

This gives a nonsmooth optimization problem of the following form:

$$\min_{\mathbf{w},b} \quad \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{j=1}^{\ell} \min(0, 1 - y_j(\mathbf{x}_j'\mathbf{w} - b)). \tag{2.4}$$

The soft margin SVM formulation has $m+1$ variables and no constraints. Nonsmooth bundle algorithms to solve this problem typically exhibit O($m\ell$) training-time complexity [24]. This has the advantage that the SVM algorithm solves this in linear-time with-respect-to the dataset size. This means that if the dataset size doubles, then the training time also doubles.

The unconstrained soft margin problem is convex and nonsmooth like many SVM problems, but must be transformed before traditional simple linear programming (LP) or convex quadratic programming (QP) algorithms can be applied. The classical approach for smoothing most nonsmooth SVM problems is to introduce dummy variables and constraints which convert the SVM problem into an equivalent LP or QP. The j^{th} maximization term can be replaced by dummy variable $\boldsymbol{\xi}_j$ by adding the following two constraints to the problem:

$$y_j(\mathbf{x}_j'\mathbf{w} - b) \geq 1 - \boldsymbol{\xi}_j \tag{2.5}$$

$$\boldsymbol{\xi}_j \geq 0. \tag{2.6}$$

Therefore the soft margin problem can be rewritten as the following equivalent QP:

$$\begin{array}{lrcll} \min\limits_{\mathbf{w},b,\boldsymbol{\xi}} & \frac{1}{2}\|\mathbf{w}\|_2^2 + C \sum\limits_{j=1}^{\ell} \boldsymbol{\xi}_j & & & \\ \text{s.t.} & y_j(\mathbf{x}_j'\mathbf{w} - b) & \geq & 1 - \boldsymbol{\xi}_j & \text{for } j = 1..\ell \\ & \boldsymbol{\xi}_j & \geq & 0 & \text{for } j = 1..\ell. \end{array} \tag{2.7}$$

This smoothed primal formulation has $\ell+m+1$ variables and 2ℓ constraints. General QP algorithms to solve this problem have $O(m\ell(\ell+m+1)^3)$ complexities [20]. Thus this primal problem is not linearly scalable in the dataset size. Doubling the dataset size does not result in doubling the computational time.

The dual to the SVM primal Problem 2.7 is:

$$\begin{array}{ll} \max\limits_{\boldsymbol{\alpha}} & -\frac{1}{2}\sum\limits_{i,j} \boldsymbol{\alpha}_i\boldsymbol{\alpha}_j y_i y_j \mathbf{x}_i'\mathbf{x}_j + \sum\limits_{j} \boldsymbol{\alpha}_j \\ \text{s.t.} & \sum\limits_{j} y_i\boldsymbol{\alpha}_j = 0 \\ & 0 \leq \boldsymbol{\alpha}_j \leq C \quad \forall\, j. \end{array} \tag{2.8}$$

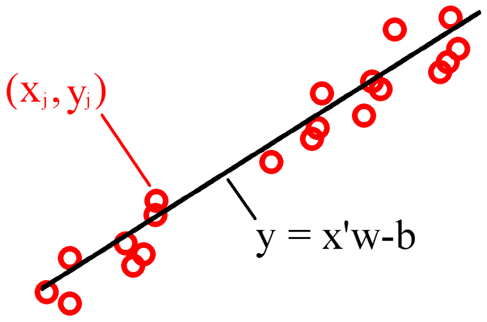

Figure 2.4: Ridge regression fits a linear function (black line) to data points (red circles) by trading off model complexity ($\frac{1}{q}\|\mathbf{w}\|_q^q$) and summed data point errors ($\frac{1}{2}\|y - (\mathbf{x}'\mathbf{w} - b)\|_2^2$).

This formulation has the advantage that it has ℓ variables, one linear constraint and 2ℓ bounds constraints. Using the liner kernel that this thesis addresses, the complexity of computing the kernel is O(ℓm^2) and needs only be computed once and stored. This dual formulation is beneficial for problems with many features compared to the dataset size ($m >> \ell$). The complexity of this dual problem is O($(m\ell)\ell^3$) [20].

Choosing which of the three formulations to use depends on the available optimization solvers (generalized bundle, QP or LP algorithms), the given problem size, and user's end goal. As we will see in later chapters, in addition to computation complexity there are advantages and disadvantages to solving each of the three formulations for use within model selection. None of the three problems gives any insight into the choice of hyperparameter C. This value must be selected by the user, and since varying C yields different models, this choice is referred to as *model selection* and is the focus of this thesis.

2.1.3 Ridge Regression

The single unified SVM framework (see Equation 2.2) can solve many different learning tasks, such as regression, which is called support vector regression (SVR). As seen in Figure 2.4, the goal of regression is to find a hyperplane that best fits a given set of data. The framework is consistent with classification, but instead of minimizing the distance from the hyperplane to misclassified points, regression minimizes regression loss such as the least-squared error. Unlike classification, $y \in \mathbb{R}$ is now a continuous label.

In addition to changing the metric, a 2-norm square penalty is used on errors, rather than the 1-norm penalty previously used. The advantage of the 2-norm squared over the 1-norm is that the function is smooth. This allows for more general algorithms to solve the training problem, and with a smooth objective, the unconstrained primal problem has a linear optimality condition. This latter advantage will be widely used throughout this thesis.

The ridge regression training loss function is

$$\mathcal{L}_{trn}(\mathbf{w}, b, \boldsymbol{\gamma}) = \frac{1}{2}\sum_{j=1}^{\ell}\left(\mathbf{x}_j'\mathbf{w} - b - y_j\right), \tag{2.9}$$

which gives the distance from the hyperplane to the data points. The structural risk term, $\mathcal{S}(\mathbf{w}, b, \boldsymbol{\gamma}) = \frac{1}{2}\|\mathbf{w}\|_q^q$, and tradeoff parameter C are still present to trade off training error with model complexity.

The ridge regression problem can be formulated as the following optimization problem:

$$\min_{\mathbf{w},b} \; \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{2}\sum_{j=1}^{\ell}\left(\mathbf{x}_j'\mathbf{w} - b - y_j\right)^2. \tag{2.10}$$

This formulation is smooth in both $\mathbf{w}$ and b. The number of variables is $m+1$ and the problem is unconstrained. This problem has a closed form optimality condition:

$$\begin{bmatrix} I + C\displaystyle\sum_{(i,j)=1}^{\ell}\mathbf{x}_i\mathbf{x}_j' & C\displaystyle\sum_{j=1}^{\ell}\mathbf{x}_j \\ \displaystyle\sum_{j=1}^{\ell}\mathbf{x}_j' & -\ell \end{bmatrix}\begin{bmatrix}\mathbf{w} \\ b\end{bmatrix} = \begin{bmatrix} C\displaystyle\sum_{(j)=1}^{\ell}\mathbf{x}_j y_j \\ \displaystyle\sum_{j=1}^{\ell} y_j \end{bmatrix}. \tag{2.11}$$

This means that solving Problem 2.10 is equivalent to solving the linear system 2.11. Solving the linear system takes $\mathrm{O}(m^3)$ work.

The choice of hyperparameter C must be defined, and this model selection