A BILEVEL OPTIMIZATION APPROACH TO MACHINE LEARNING

By

Gautam Kunapuli

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

_____
Joseph G. Ecker, Member

_____
John E. Mitchell, Member

_____
Jong-Shi Pang, Member

_____
Bülent Yener, Member

Rensselaer Polytechnic Institute
Troy, New York

February 2008
(For Graduation May 2008)

© Copyright 2008
by
Gautam Kunapuli
All Rights Reserved

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENT

This work would not have been possible without the inestimable guidance of Kristin Bennett, my mentor and advisor. I would like to thank her, first and foremost, for her constant support and encouragement. I like to thank Jong-Shi Pang, with whom I collaborated extensively on these projects, for his guidance and support. I would also like to thank John Mitchell, Joseph Ecker and Bülent Yener, the other members of my Committee for their contributions.

I would also like to thank my friends for everything; they were a great source of support during the good and not so good times, but always there... to the very end. Needless to say, they were also a source of the occasional, albeit warranted distraction. In particular, I would like to acknowledge Guilherme de Oliveira, Ayala Cnaan, Juan Latorre, Maral Erol, Selma Sabanovic, Matt Francisco, Lee Deville, Zack Galbreath, Tara Gallaway, Laurel Reilly-Raska, Jon Collis, Elizabeth Kusel, Charles Beckman, Daryn Ramsden, Nadia Sidarous, Josh Attenberg, John Morris, Marijana and Boris Pokorni, Rachel and Gary Dale, Siddharth Devarajan and Jairam Kalyanasundharam.

To all my friends I say: thank you for all the good times. It is an open secret among graduate students that the only part of a dissertation that is ever of interest is the acknowledgements. So, if you are reading this and realize that I have not adequately thanked you, you should probably call me up to remonstrate with me. If nothing else, it will give us a chance to catch up.

I would also like to acknowledge the encouragement of several members of my family here and back home. I would especially like to thank my parents; their love and support made it all worthwhile. And finally, I dedicate this thesis to my brother, who has always believed in me and has been my greatest well-wisher.

ABSTRACT

A key step in many statistical learning methods used in machine learning involves solving a convex optimization problem containing one or more hyper-parameters that must be selected by the users. While cross validation is a commonly employed and widely accepted method for selecting these parameters, its implementation by a grid-search procedure in the parameter space effectively limits the desirable number of hyper-parameters in a model, due to the combinatorial explosion of grid points in high dimensions. A novel paradigm based on bilevel optimization approach is proposed and gives rise to a unifying framework within which issues such as model selection can be addressed.

The machine learning problem is formulated as a bilevel program–a mathematical program that has constraints which are functions of optimal solutions of another mathematical program called the inner-level program. The bilevel program is transformed to an equivalent mathematical program with equilibrium constraints (MPEC). Two alternative bilevel optimization algorithms are developed to optimize the MPEC and provide a systematic search of the hyper-parameters.

In the first approach, the equilibrium constraints of the MPEC are relaxed to form a nonlinear program with linear objective and non-convex quadratic inequality constraints, which is then solved using a general purpose nonlinear programming solver. In the second approach, the equilibrium constraints are treated as penalty terms in the objective, and the resulting non-convex quadratic program with linear constraints is solved using a successive linearization algorithm.

The flexibility of the bilevel approach to deal with multiple hyper-parameters, makes it powerful approach to problems such as parameter and feature selection (model selection). In this thesis, three problems are studied: model selection for support vector (SV) classification, model selection for SV regression and missing value-imputation for SV regression. Extensive computational results establish that both algorithmic approaches find solutions that generalize as well or better than conventional approaches and are much more computationally efficient.

CHAPTER 1
Introduction

Machine Learning, is a truly multi-disciplinary field and draws concepts from such diverse disciplines as artificial intelligence, computational complexity, control theory, cognitive science, philosophy, neuroscience and statistics. Machine learning deals with automatically building a model of a process or system through analysis of existing patterns or data such that the new model is able to predict the responses of the system on patterns that have yet to be observed. This ability of learner to correctly predict outputs for unseen inputs is called *generalization* and all machine learning methods seek to improve this performance.

Machine learning problems are known to be ill-posed in general [71] because data set sizes are finite and sometimes small, non-uniformly sampled and high dimensional. In addition, there may be problems such as measurement errors, noise and missing values in the data. This situation is further exacerbated by limited parameter selection techniques and inaccurate estimates of generalization—two model selection strategies most learning algorithms inevitably entail. The late 80s and early 90s saw the emergence of an elegant and powerful machine learning paradigm, one that addressed many of the issues above, called *Structural Risk Minimization* (SRM). The principles of SRM give rise to powerful learners called *Support Vector Machines* (SVMs).

SVMs are universal feed-forward networks, or more specifically, are linear learning machines that are able to perform different machine learning tasks. The first SVMs were introduced for binary pattern classification [11, 19, 87]. Subsequent increased attention from the machine learning community lead to several developments: the capability to handle nonlinear data sets via the "kernel trick" [81] and other machine learning tasks such as regression [25, 83, 84], clustering [10], novelty detection [78] semi-supervised learning [6] and SVM variants such as 1-norm and linear SVMs [64, 69, 90] and ν-SVMs [16]. These progresses were accompanied by the development of fast algorithms such as sequential minimal optimization [70] (SMO)—in the late 90s—or more recently, interior-point methods for massive data sets [29], and for linear SVMs: cutting plane algorithms [49] and decomposition based approaches [65]. These advances mean that the current state-of-the-art SVMs are capable of efficiently handling data sets of several thousands to millions of data points.

SVMs have been successfully applied to a wide variety of applications such as

medicine, computational biology, finance, robotics, computer vision, image, object and handwriting recognition and text processing, to name just a few, and have emerged as one of the pre-eminent machine learning technique of our times. However, despite this success and popularity, several open questions, such as optimal model selection, still remain. The need for new methodologies to address these issues forms the primary motivation for this thesis.

Section 1.1 provides a brief background on the principles of structural risk minimization, SVMs for classification and regression and their extension into nonlinear domains via kernels Section 1.2 introduces the concepts of bilevel optimization and some common approaches to solving such problems. A reader familiar with these concepts could skip the discussion in Sections 1.1 and 1.2. Section 1.3 discusses several open questions and issues that motivate this thesis. Section 1.4 introduces a novel paradigm—bilevel programming for machine learning—that forms the framework for the various models discussed in the thesis, the layout of which is detailed in Section 1.5.

1.1 Background: Support Vector Machines

Consider that we are interested in modeling a regressive system of the form $y = f(\mathbf{x}) + \epsilon$, where the data point, $\mathbf{x}$, and its label, y are drawn from some unknown distribution $P(\mathbf{x}, y)$. In general, the hypothesis function, $f(\mathbf{x})$ may depend on the labels y and may be parameterized by $\boldsymbol{\alpha} \in \Lambda$ and is variously written as $f(\mathbf{x}, y)$, $f(\mathbf{x}; \boldsymbol{\alpha})$ or $f(\mathbf{x}, y; \boldsymbol{\alpha})$ as appropriate.

1.1.1 The Statistical Nature of Learning

The target function, f, which belongs to some *target space*, is deterministic, and the intrinsic system error, ϵ, is a random expectational error that represents our ignorance of the dependence of y on $\mathbf{x}$. Thus, $\mathbb{E}[\epsilon] = 0$ and $\mathbb{E}[y] = \mathbb{E}[f(\mathbf{x})] = f(\mathbf{x})$. The goal is to choose a function, $\widehat{f}$, in a *hypothesis space* of candidate functions that is as close to f as possible with respect to some error measure.

Suppose that we have ℓ realizations of labelled data, $D = \{(\mathbf{x}^i, y_i)\}_{i=1}^{\ell}$, that constitute the training sample. Using D, we train a function, $\widehat{y} = \widehat{f}(\mathbf{x})$ that minimizes the squared error. Thus, the expected predicted error or *risk* is

$$R[f] = \mathbb{E}_D\left[(y - \widehat{y})^2\right] = \mathbb{E}_D\left[(y - \widehat{f}(x))^2\right].$$

It is easy to show that

$$R[f] = \epsilon^2 + \left(f(\mathbf{x}) - \mathbb{E}_D\left[\widehat{f}(\mathbf{x})\right]\right)^2 + \mathbb{E}_D\left[(\widehat{y} - \mathbb{E}_D[\widehat{y}])^2\right],$$

where the expected predicted error decomposes into three terms (see Figure 1.2):

- the *noise variance*, ϵ^2, which is the noise that is inherent in the system. This term cannot be minimized by the learner as it is independent of the learning process. It is also referred to as the *intrinsic error*.
- the *squared bias*, $(f(\mathbf{x}) - \mathbb{E}_D[\widehat{f}(\mathbf{x})])^2$, measures the difference between the true function and the average of the measured values at $\mathbf{x}$ and represents the inability of the learner to accurately approximate the true function. This indicates that the choice of model was poor because the hypothesis space is too small to contain the true function. The bias is also called the *approximation error*.
- the *estimation variance*, $\mathbb{E}_D[(\widehat{y} - \mathbb{E}_D[\widehat{y}])^2]$, measures the error in estimating the true labels and how sensitive the model is to random fluctuations in the data set. This is also called the *estimation error*.

To reduce training error on the given data, the learner must minimize the estimation variance over the training set. To reduce generalization error on the future, unseen data, the learner must minimize the bias over the training set. Unfortunately, it is not possible to decrease one without increasing the other as there is a natural trade-off between the bias and the variance. This leads to, what is commonly referred to as, *the bias-variance dilemma*, where the learner must minimize some combination of variance and bias.

To see this, consider Figure 1.1.1, where a data set is fit by a linear function and some high-degree polynomial function. If we restricted the choice of target functions to just linear functions, we would obtain linear fits for every data set and we have, consequently, introduced a *bias* into the learning process. However, if we expanded the set of target functions to include higher degree polynomials, we would obtain highly nonlinear fits that are susceptible to small fluctuations in the data and consequently, have high *variance*. Related to this notion of complexity is the *capacity* of a class of functions that constitute the hypothesis space. A hypothesis space with higher capacity yields overly complex models, which *overfit* the data, whereas smaller capacity gives overly simple models which

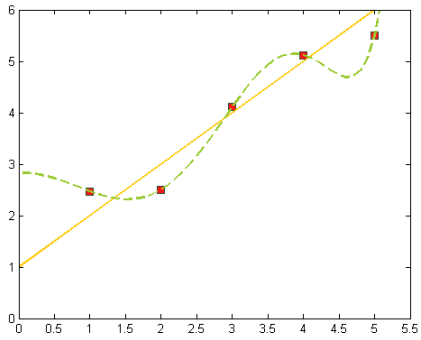


Figure 1.1: The effect of complexity on generalization.

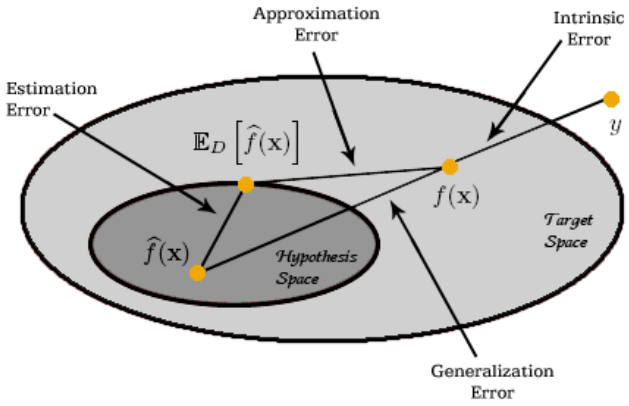


Figure 1.2: Different errors that arise during the modeling of a typical Machine Learning task.

underfit the data.

1.1.2 Empirical Risk Minimization

This section and the next present a brief introduction to empirical and structural risk minimization since most of the machine learning methods considered hereafter are based on these principles. A more detailed discussion of this theory may be found in [87].

We assume that the training data, $\mathbf{x} \in \mathbb{R}^n$, are drawn identically and independently distributed (i.i.d.) from a distribution, $P(\mathbf{x})$. The corresponding labels, y, are assumed to be drawn from a conditional distribution, $P(y|\mathbf{x})$. The machine learning aim is to find a function, $f(\mathbf{x}, \boldsymbol{\alpha})$, parameterized by $\boldsymbol{\alpha} \in \Lambda$, that best approximates (predicts) y. This is achieved by minimizing the loss between the given and predicted labels, i.e., minimizing the *expected risk functional*,

$$R[\boldsymbol{\alpha}] = \int L\left(y, f(\mathbf{x}, \boldsymbol{\alpha})\right) dP(\mathbf{x}, y), \tag{1.1}$$

with $F(\mathbf{x}, y) = F(\mathbf{x})F(y|\mathbf{x})$. The loss function, $L\left(y, f(\mathbf{x}, \boldsymbol{\alpha})\right)$ is an error measure between the expected and predicted labels (typically the number of misclassifications for classification tasks and mean average deviation or mean squared error for regression tasks— there are many others). However, computing the expected risk is not easy since the distribution, $F(\mathbf{x}, y)$, is typically unknown.

Instead, using inductive principles, we define the *empirical risk*, which is based

only on the finite data set, X, that we have for training the learner,

$$R_{emp}[\boldsymbol{\alpha}] = \frac{1}{\ell}\sum_{i=1}^{\ell} L\left(y_i, f(\mathbf{x}^i, \boldsymbol{\alpha})\right). \tag{1.2}$$

and ℓ is the number of data points. On the surface, ERM minimizes the empirical risk, using it as an estimate of the risk. The law of large numbers, which forms the theoretical basis for the application of several estimation approaches, ensures that the empirical risk R_{emp} converges to the expected risk R as the number of data points approaches infinity:

$$\lim_{\ell\to\infty} R_{emp}[\boldsymbol{\alpha}] = R[\boldsymbol{\alpha}]. \tag{1.3}$$

However, (1.3) does not guarantee that the function f_{emp} that minimizes the empirical risk R_{emp} converges towards the function f that minimizes R. This is the notion of *asymptotic consistency* and applies equally to the parameters $\boldsymbol{\alpha}_{emp}$ and $\boldsymbol{\alpha}$ that parameterize f_{emp} and f respectively.

Definition 1.1.1 (Consistency). *We say that ERM is consistent if there exists a sequence of models, $\boldsymbol{\alpha}_\ell$, or functions, $L\left(y, f(\mathbf{x}, \boldsymbol{\alpha}_\ell)\right)$, $\ell = 1, 2, \ldots$, such that the expected and empirical risks converge to the minimal value of the risk, i.e, if the following two sequences converge in probability*[1]

$$\begin{aligned} \lim_{l\to\infty} Prob\left\{R[\boldsymbol{\alpha}_\ell] - \inf_{\boldsymbol{\alpha}\in\Lambda} R[\boldsymbol{\alpha}] > \epsilon\right\} = 0, \quad \forall\epsilon > 0, \\ \lim_{l\to\infty} Prob\left\{\inf_{\boldsymbol{\alpha}\in\Lambda} R[\boldsymbol{\alpha}] - R_{emp}[\boldsymbol{\alpha}_\ell] > \epsilon\right\} = 0, \quad \forall\epsilon > 0. \end{aligned} \tag{1.4}$$

Consistency is essential so as to guarantee that the model $\boldsymbol{\alpha}_\ell$ converges to $\boldsymbol{\alpha}$. Thus, the concept of consistency of the ERM principle becomes central to learning theory.

Theorem 1.1.2 (Key Theorem of Learning Theory, [87]). *Let $L\left(y, f(\mathbf{x}, \boldsymbol{\alpha})\right)$, $\boldsymbol{\alpha} \in \Lambda$, be a set of functions that satisfy the condition*

$$A \leq \int L\left(y, f(\mathbf{x}, \boldsymbol{\alpha})\right) dP(\mathbf{x}, y) \leq B \quad (A \leq R[\boldsymbol{\alpha}] \leq B).$$

[1] The equations (1.4) are examples of probably approximately correct (PAC) bounds where the bound is meant to hold in a probabilistic sense. Thus, Prob $\left\{\inf_{\boldsymbol{\alpha}\in\Lambda} R[\boldsymbol{\alpha}] - R_{emp}[\boldsymbol{\alpha}_\ell] > \epsilon\right\}$ refers to the probability of getting a model $\boldsymbol{\alpha}_l$ with the property $\inf_{\boldsymbol{\alpha}\in\Lambda} R[\boldsymbol{\alpha}] - R_{emp}[\boldsymbol{\alpha}_\ell] > \epsilon$.

Then, for the ERM principle to be consistent, it is necessary and sufficient that the empirical risk $R_{emp}[\boldsymbol{\alpha}]$ converge uniformly to the actual risk $R[\boldsymbol{\alpha}]$ over the set $L\left(y_i, f(\mathbf{x}^i, \boldsymbol{\alpha})\right)$, $\boldsymbol{\alpha} \in \Lambda$, in the following sense:

$$\lim_{l \to \infty} Prob\left\{\sup_{\boldsymbol{\alpha} \in \Lambda}(R[\boldsymbol{\alpha}] - R_{emp}[\boldsymbol{\alpha}]) > \epsilon\right\} = 0, \quad \forall \epsilon > 0.$$

The theorem states that consistency of the ERM principle is equivalent to the existence of one-sided uniform convergence as the number of samples grows infinite. It should be noted that the two-sided convergence is too strong for ERM purposes since, in most machine learning problems, we are only interested in minimizing and not maximizing the empirical risk. It is clear that consistency depends upon the worst function from the hypothesis space that provides the largest error between the empirical and expected risks. thus, ERM is a worst-case analysis.

The theory of uniform convergence that gives the necessary and sufficient conditions for consistency also provides distribution-independent bounds on the rate of convergence. This bound typically depends on a measure of the *capacity* or expressive power of the class of functions. Different bounds can be derived using different capacity measures but the most common bound is provided by the *Vapnik-Chervonenkis dimension* or simply the VC dimension. We will consider the classification task performed upon a data set with ℓ points and a hypothesis space of indicator functions, $\mathcal{F} = \{sign(f(\boldsymbol{\alpha})) \,|\, \boldsymbol{\alpha} \in \Lambda\}$. These ℓ points can be labelled in 2^ℓ possible ways as 0 or 1, i.e., a data set with ℓ points gives 2^ℓ distinct learning problems.

Definition 1.1.3 (VC Dimension)**.** *The maximum number of points (in some configuration) that can be shattered by $\mathcal{F}$ is called the VC dimension (h) of $\mathcal{F}$. If there is no such maximum, the VC dimension is said to be infinite.*

The VC dimension is has nothing to do with geometric dimension; it is a combinatorial concept that tries to represent the capacity (or complexity) of $\mathcal{F}$ with regard to data set size rather than the number of hypotheses, $|\mathcal{F}|$, because the latter could be infinite. Note that if the VC dimension is h, then there exists *at least* one set with h points that can be shattered, but this is not necessarily true for every set with h points except for very simple cases. Figure 1.3, shows that the VC dimension of the set of two dimensional linear discriminants in $\mathbb{R}^2$ is 3.

Intuitively, a hypothesis space with a very large or infinite VC dimension represents

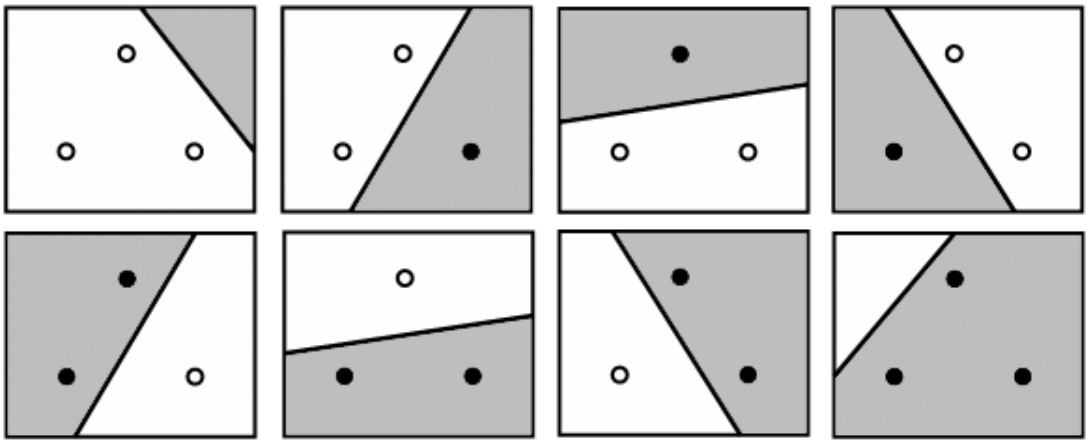


Figure 1.3: All possible placements of three points shattered by the hypothesis space containing all two dimensional linear functions, $\mathcal{F} = \{(\mathbf{w}, b) \in \mathbb{R}^{n+1} \,|\, \mathbf{w}'\mathbf{x} - b = 0\}$. The VC dimension of $\mathcal{F}$ is 3.

a very rich class of functions which will be able to fit every labeling of the data and there will either be overfitting or perhaps, even no learning and this class is not very useful. Thus, for learning to be effective, and ultimately, for good generalization, the VC dimension must be finite. The following theorem summarizes the above conclusions succinctly.

Theorem 1.1.4 (Fundamental Theorem of Learning, [87])**.** *Let $\mathcal{F}$ be a hypothesis space having VC dimension h. Then with probability $1 - \delta$, the following bound holds*

$$R[\boldsymbol{\alpha}] \leq R_{emp}[\boldsymbol{\alpha}] + \sqrt{\frac{h}{\ell}\ln\left(\frac{2\ell}{h} + 1\right) - \frac{1}{\ell} ln\left(\frac{\delta}{4}\right)}. \tag{1.5}$$

The second term in the right hand side is known as the *VC confidence*.

1.1.3 Structural Risk Minimization

Theorem 1.1.4 provides a distribution-independent bound on the true risk; it is a combination of the empirical risk and a confidence interval term that is able to control the capacity of a class of functions measured through the VC dimension, h. Thus, the best model can be obtained by minimizing the left-hand side of the inequality (1.5), and SRM aims to do precisely this. SRM is an inductive principle like ERM which is used to learn models from finite data sets. However, unlike ERM, which simply minimizes the empirical risk, SRM provides a mechanism to control the capacity through a trade-off between the empirical risk and the hypothesis space complexity. This is very similar to

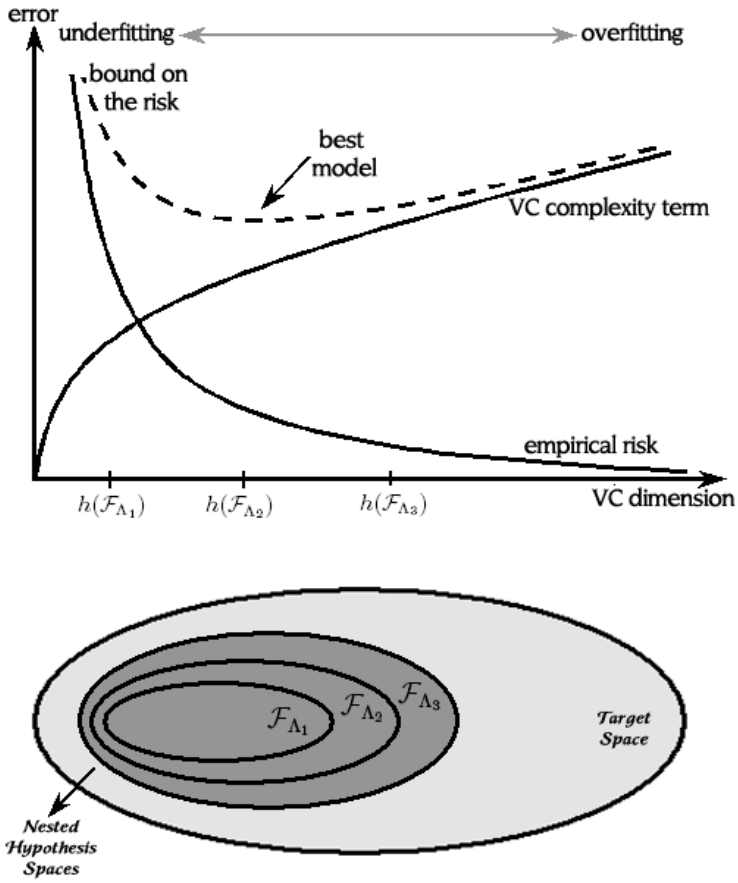


Figure 1.4: SRM creates a structure in the hypothesis space: nested subsets ordered by VC dimension.

the bias-variance trade-off discussed in Section 1.1.1.

SRM (see Figure 1.4) orders function classes according to their complexity such that they form a *nested structure*, with more complex function classes being supersets of the less complex classes such that it is possible to either compute the VC dimension for a subset or at least obtain a bound on it. SRM then consists of finding the subset that minimizes the bound on the actual risk in (1.5). In other words, SRM seeks to create a structure of nested hypothesis spaces, $\mathcal{F}_1 \subset \mathcal{F}_2 \subset \mathcal{F}_3 \ldots$—with $\mathcal{F}_h$ being a hypothesis space of VC dimension h—and solve the problems

$$\min_{\substack{f[\boldsymbol{\alpha}]\in\mathcal{F}_h \\ \boldsymbol{\alpha}\in\Lambda}} R_{emp}[\boldsymbol{\alpha}] + \sqrt{\frac{h}{\ell}\ln\left(\frac{2\ell}{h}+1\right) - \frac{1}{\ell}ln\left(\frac{\delta}{4}\right)}. \tag{1.6}$$

While SRM principles may seem fairly straightforward, implementation is actually difficult because of several reasons. Firstly, it is difficult to find the VC-dimension for a given

hypothesis space since it is not apparent exactly how to calculate this for all machines. Secondly, even if it were possible to calculate h for a specific machine, it is may not be trivial to solve the subsequent optimization problem (1.6). Finally, SRM does not explicitly specify any particular structure or indicate how the nested hypothesis spaces can be constructed. Support Vector Machines, introduced by Boser, Guyon and Vapnik, achieve this by relating the complexity of the function classes to the *margin*.

1.1.4 SVMs for Linear Classification

We consider the problem of separating the set of labeled training vectors belonging to two separate classes, $\{\mathbf{x}^i, \mathbf{y}_i\}_{i=1}^{\ell} \in \mathbb{R}^{n+1}$, with $y_i = \pm 1$ indicating class membership using a hyperplane, $f(\mathbf{x}) = \mathbf{w}'\mathbf{x} - b$. The function, $f(\mathbf{x})$, is a candidate hypothesis from a set of all possible hyperplanes, $\mathcal{F} = \{f : \mathbb{R}^n \to \mathbb{R} \,|\, f(\mathbf{x}) = \mathbf{w}'\mathbf{x} - b\}$. It is easy to show that the VC dimension of $\mathcal{F}$ is $h(\mathcal{F}) = n + 1$, which indicates that in a n-dimensional input space, the maximal number of arbitrarily labeled points that can be linearly separated is $n + 1$.

1.1.4.1 From SRM to SVM

SRM principles require that the VC dimension of the hypothesis space be finite and the hypothesis space should allow for the construction of nested structure of function classes. The latter is achieved by setting

$$\mathcal{F}_\Lambda = \left\{ f : \mathbb{R}^n \to \mathbb{R} \,|\, f(\mathbf{x}) = \mathbf{w}'\mathbf{x} - b, \|\mathbf{w}\|_2 \leq \Lambda \right\}. \tag{1.7}$$

Before describing exactly how the definition above creates a nested function space, the concept of canonical hyperplanes is introduced because the current hyperplane representation, $(\mathbf{w}, b)$, is not unique. It is apparent that if a given data set is separable by a hyperplane $(\mathbf{w}, b)$, it is also separable by all hyperplanes of the form $(t\mathbf{w}, tb)$, $\forall t \geq 0$. This is a problem since there are infinitely many such hyperplanes and every function class $\mathcal{F}_\Lambda$ would contain the same functions in different representations. Consequently, all the function classes, $\mathcal{F}_{\Lambda_1}, \mathcal{F}_{\Lambda_2}, \ldots$ would have the same VC dimension rather than the desired "nested structure" property that $\mathcal{F}_{\Lambda_1} \subset \mathcal{F}_{\Lambda_2} \subset \ldots$ for $\Lambda_1 \leq \Lambda_2 \leq \ldots$ with increasing VC dimensions for increasing Λ_i (see Figure 1.4).

To ensure that (1.7) actually works, we need to define a unique representation for each hyperplane. So, without loss of generality we can define the canonical hyperplane for

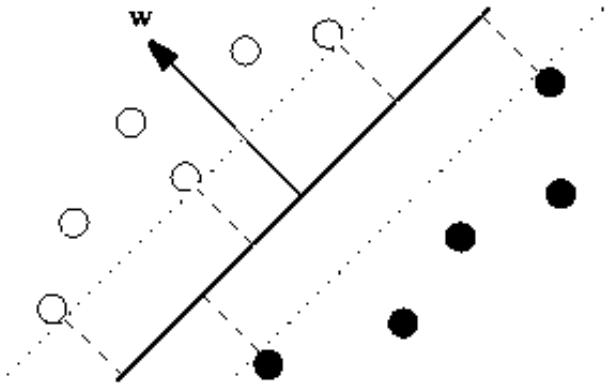


Figure 1.5: Canonical Hyperplanes and Margins: None of the training examples produces an absolute output that is smaller than 1 and the examples closest the hyperplane have exactly an output of one, i.e $\mathbf{w}'\mathbf{x} - b = \pm 1$

a data set as the function, $f(\mathbf{x}) = \mathbf{w}'\mathbf{x} - b$, with $(\mathbf{w}, b)$ constrained by

$$\min_i |\mathbf{w}'\mathbf{x}^i - b| = 1. \tag{1.8}$$

Simply put, it is desired that the norm of the weight vector should be equal to the inverse distance of the nearest point in the set to the hyperplane (see Figure 1.5). Since the data was assumed to be linearly separable, any hyperplane can be set in canonical form by suitably normalizing $\mathbf{w}$ and adjusting b.

The *margin* is defined to be the minimal Euclidean distance between a training example, $\mathbf{x}^i$, and the separating hyperplane. Intuitively, the margin measures how good the separation between the two classes by a hyperplane is. This distance is computed as

$$\begin{aligned} \gamma(\mathbf{w}, b) &= \min_{\mathbf{x}^i : y_i = 1} \frac{|\mathbf{w}'\mathbf{x}^i - b|}{\|\mathbf{w}\|} + \min_{\mathbf{x}^i : y_i = -1} \frac{|\mathbf{w}'\mathbf{x}^i - b|}{\|\mathbf{w}\|} \\ &= \frac{1}{\|\mathbf{w}\|} \left(\min_{\mathbf{x}^i : y_i = 1} |\mathbf{w}'\mathbf{x}^i - b| + \min_{\mathbf{x}^i : y_i = -1} |\mathbf{w}'\mathbf{x}^i - b| \right) \\ &= \frac{2}{\|\mathbf{w}\|} \end{aligned} \tag{1.9}$$

Thus, smaller the norm of the weight vector, larger the margin and larger the margin, smaller the complexity of the function class (see Figure). More generally, it was shown in [87] that if the hyperplanes are constrained by $\|\mathbf{w}\| \leq \Lambda$, then the VC dimension of the class, $\mathcal{F}_\Lambda$, is bounded by $h(\mathcal{F}_\Lambda) \leq \min(\lceil \Lambda^2 R^2 \rceil + 1, n + 1)$, where R is the radius of the smallest sphere enclosing the data. We can construct nested function classes by bounding the margins of each function class by $\frac{2}{\Lambda}$. Thus, for $\Lambda_1 \leq \Lambda_2 \leq \ldots$, we have

$\mathcal{F}_{\Lambda_1} \subset \mathcal{F}_{\Lambda_2} \subset \ldots$ with $h(\mathcal{F}_{\Lambda_1}) \leq h(\mathcal{F}_{\Lambda_2}) \leq \ldots$, realizing the structure necessary to implement the SRM principle. Thus, the typical support vector based classifier can be trained by solving the following problem:

$$\begin{aligned} \underset{\mathbf{w}, b, \boldsymbol{\xi}}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{\ell}\xi_i \\ \text{subject to} \quad & \forall i = 1, \ldots, \ell, \\ & y_i(\mathbf{w}'\mathbf{x}^i - b) \geq 1 - \xi_i, \\ & \xi_i \geq 0. \end{aligned} \tag{1.10}$$

The variables ξ_i are called *slack variables*. The problem above defines a *soft-margin* support vector machine, i.e., the assumption that the data are linearly separable without misclassification error (by some function in the hypothesis space) is dropped. The slack variables are introduced to account for points of one class that are misclassified, by a hypothesis function, as points of the other class; they measure the distance of the misclassified points from the hyperplane. So, if a point $\mathbf{x}^i$ is correctly classified, its corresponding slack, $\xi_i = 0$ otherwise we will have $\xi_i > 0$. The final classifier will be $f(\mathbf{x}) = sign(\mathbf{w}'\mathbf{x} - b)$.

The *regularization constant*, C, gives the trade-off between bias and variance, or in terms of the SRM principle, the confidence interval (capacity) and empirical risk. The parameter C has to be set *a priori* to choose between different models with trade-offs between training error and the margin.

1.1.4.2 First Order Conditions and Dual Program

It is evident that solving (1.10) yields the optimal, canonical, maximal-margin hyperplane, which is the ultimate goal of statistical learning theory implemented through SVMs. Problem (1.10) is an instance of a quadratic program (QP) with inequality constraints. From classical Lagrangian theory, we know that the solution to (1.10) is the saddle point of the Lagrangian. The latter may be constructed by introducing Lagrange multipliers α_i for the hyperplane constraints and η_i for the slack non-negativity constraints. The Lagrangian is

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\eta}) = \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{\ell}\xi_i - \sum_{i=1}^{\ell}\alpha_i\left(y_i(\mathbf{w}'\mathbf{x}^i - b) - 1 + \xi_i\right) - \sum_{i=1}^{\ell}\eta_i\xi_i \tag{1.11}$$

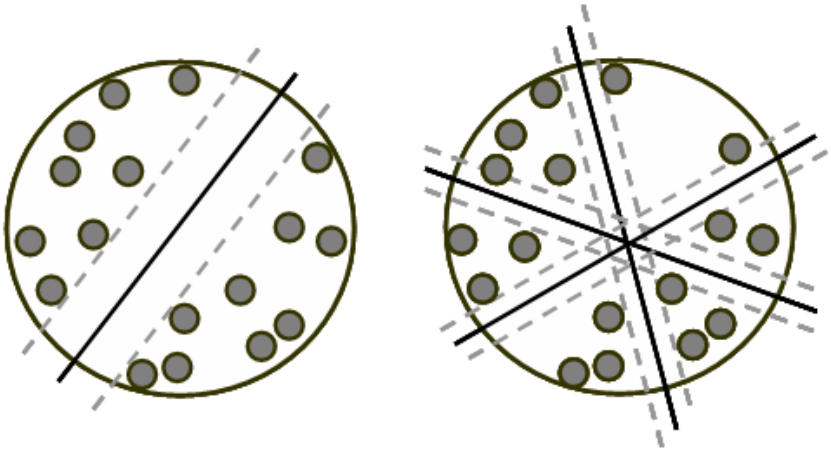


Figure 1.6: Illustration of why a large margin reduces the complexity of a linear hyperplane classifier.

The first-order *Karush-Kunh-Tucker* conditions for (1.10) consist of the following system of equalities,

$$\begin{aligned} \nabla_{\mathbf{w}} \mathcal{L} = 0 &\Longrightarrow \mathbf{w} - \sum_{i=1}^{\ell} \mathbf{y}_i \alpha_i \mathbf{x}^i = 0, \\ \nabla_b \mathcal{L} = 0 &\Longrightarrow \sum_{i=1}^{\ell} y_i \alpha_i = 0, \\ \nabla_{\xi_i} \mathcal{L} = 0 &\Longrightarrow C - \alpha_i - \eta_i = 0, \forall i = 1, \ldots, \ell, \end{aligned} \tag{1.12}$$

and the complementarity conditions,

$$\left.\begin{array}{ll} 0 \leq \alpha_i \perp y_i(\mathbf{w}'\mathbf{x}^i - b) - 1 + \xi_i, \geq 0 & \\ 0 \leq \xi_i \perp & C - \alpha_i \geq 0 \end{array}\right\} \forall i = 1, \ldots, \ell, \tag{1.13}$$

where the variables η_i have been eliminated in constructing the second set of complementarity conditions. The notation $a \perp b$ implies the condition $a'b = 0$. The (Wolfe) dual to (1.10) can be derived by substituting (1.12) into the Lagrangian. This yields:

$$\begin{aligned} \underset{\boldsymbol{\alpha}}{\text{maximize}} \quad & -\frac{1}{2} \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} y_i y_j \alpha_i \alpha_j (\mathbf{x}^i)' \mathbf{x}^j + \sum_{i=1}^{\ell} \alpha_i \\ \text{subject to} \quad & \sum_{i=1}^{\ell} y_i \alpha_i = 0, \\ & 0 \leq \alpha_i \leq C, \quad \forall i = 1, \ldots, \ell. \end{aligned} \tag{1.14}$$

The dual problem, (1.14), is easier to solve than the primal as it has much fewer constraints. It is interesting to note that, in the dual solution, only those training data that are misclassified by the canonical hyperplanes $y_i(\mathbf{w}'\mathbf{x} - b) \geq 1$ have non-zero α_i. The training data for which $\alpha_i > 0$ are called *support vectors*.

Some very interesting points emerge from the study of the KKT conditions (1.12–1.13) and the dual (1.14). Firstly, that the hyperplane $\mathbf{w}$ can be expressed as a linear combination of the training data $\mathbf{x}^i$. Furthermore, it is apparent that only those points that lie on or below the margin for a given class have $\alpha_i > 0$. (see Figure). This suggests that the hyperplane is a sparse linear combination of the training data. Finally, the dual problem (which, as we have already noted, is easier to solve) does not depend on the training data, but rather on the *inner products*. The importance of this last fact will become fully apparent when SVMs are extended to handle nonlinear data sets by applying the "kernel trick" (see Section 1.1.6).

1.1.5 Linear Regression

Though SV techniques were originally introduced for solving classification problems, they have also be extended and successfully applied to solve regression problems [25, 83, 84]. We modify the notation introduced so far slightly. The labels are real-valued, $y_i \in \mathbb{R}$, and are no longer restricted. As before, we wish to train a function $f(x) = \mathbf{w}'\mathbf{x} - b$ to perform regression on the given data.

1.1.5.1 Loss Functions

Recall that SRM attempts to minimize some tradeoff between the VC confidence and the empirical risk. The latter depends on the loss function. There exist several well-known loss functions for regression, most notably, the L_2 loss, $L(y_i, f(\mathbf{x}^i)) = (y_i - f(\mathbf{x}^i))^2$, which is used for least squares and ridge regression, and the L_1 loss, $L(y_i, f(\mathbf{x}^i)) = |y_i - f(\mathbf{x}^i)|$. Yet another loss function that arises from robust regression applications is Huber's loss, which is a smooth combination of L_1 and L_2 losses.

Vapnik introduced a new loss function called the ε-insensitive loss in order to capture the notion of sparsity that arose from the use of the margin in support vector classification. The loss is defined as

$$L(y_i, f(\mathbf{x}^i)) = |y_i - f(\mathbf{x}^i)|_\varepsilon = \begin{cases} 0, & \text{if } |y_i - f(\mathbf{x}^i)| \leq \varepsilon, \\ |y_i - f(\mathbf{x}^i)| - \varepsilon, & \text{if } |y_i - f(\mathbf{x}^i)| > \varepsilon. \end{cases} \tag{1.15}$$

This loss function has the effect of creating an ε-insensitivity tube around the hypothesis function $f(\mathbf{x})$ and only penalizing those points whose labels lie outside the tube. If the predicted values lie inside the tube, the corresponding loss is zero. The empirical risk functional associated with the ε-insensitive loss is

$$R_{emp} = \frac{1}{\ell} \sum_{i=1}^{\ell} \max(|\mathbf{w}'\mathbf{x}^i - b - y_i| - \varepsilon, 0). \tag{1.16}$$

Introducing slack variables ξ_i as before to measure the distance of the outlier-points above and below to the tube, we can write down a formulation for support vector regression using the ε-insensitive loss:

$$\begin{array}{ll} \underset{\mathbf{w}, b, \boldsymbol{\xi}}{\text{minimize}} & \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{\ell} \xi_i \\ \text{subject to} & \forall\, i = 1, \ldots, \ell, \\ & \mathbf{w}'\mathbf{x}^i - b - y_i + \varepsilon + \xi_i \geq 0, \\ & y_i - \mathbf{w}'\mathbf{x}^i + b + \varepsilon + \xi_i \geq 0, \\ & \xi_i \geq 0. \end{array} \tag{1.17}$$

1.1.5.2 First Order Conditions and Dual Program

Lagrange multipliers $\alpha_i^{\pm}$ are introduced for the upper and lower hyperplane constraints and η_i for the ξ-nonnegativity constraints. As for the classification case, the KKT first order conditions can be written down, which consist of the equality conditions,

$$\begin{array}{l} \mathbf{w} + \sum_{i=1}^{\ell} (\alpha_i^+ - \alpha_i^-)\mathbf{x}^i = 0, \\ \sum_{i=1}^{\ell} (\alpha_i^+ - \alpha_i^-) = 0, \end{array} \tag{1.18}$$

and the complementarity conditions,

$$\left.\begin{array}{ll} 0 \leq \alpha_i^+ \perp y_i - \mathbf{w}'\mathbf{x}^i + b + \varepsilon + \xi_i \geq 0 & \\ 0 \leq \alpha_i^- \perp \mathbf{w}'\mathbf{x}^i - b - y_i + \varepsilon + \xi_i \geq 0 & \\ 0 \leq \xi_i \perp \qquad C - \alpha_i^+ - \alpha_i^- \geq 0 & \end{array}\right\} \forall i = 1, \ldots, \ell, \tag{1.19}$$

where the variables η_i have been eliminated in constructing the second set of complementarity conditions. The (Wolfe) dual to the support vector regression problem is

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & -\frac{1}{2}\sum_{i=1}^{\ell}\sum_{j=1}^{\ell}(\alpha_i^+ - \alpha_i^-)(\alpha_j^+ - \alpha_j^-)(\mathbf{x}^i)'\mathbf{x}^j - \varepsilon\sum_{i=1}^{\ell}(\alpha_i^+ + \alpha_i^-) + \sum_{i=1}^{\ell} y_i(\alpha_i^+ - \alpha_i^-) \\ \text{s.t.} \quad & \sum_{i=1}^{\ell}(\alpha_i^+ - \alpha_i^-) = 0, \\ & 0 \le \alpha_i^{\pm} \le C, \quad \forall i = 1, \ldots, \ell. \end{aligned} \tag{1.20}$$

As in the classification case, the dual problem for SV regression is similarly easier to solve compared to the primal. The desirable properties of SV classification dual, such as sparsity, get carried over to the regression problem. In addition, the dual variables $\boldsymbol{\alpha}^{\pm}$ are intrinsically complementary i.e., $\alpha_i^+ \perp \alpha_i^-$. This is because no point can simultaneously be on both sides of the ε-tube causing at least one of the two hyperplane constraints to be inactive always.

1.1.6 Kernel Methods

The SVM methods presented in the previous subsections can be used to learn *linear* functional relationships between the data and their target labels. This learned predictive function can then be used to generalize and predict labels for new data. The methods are limited, however, as they are only able to construct linear relationships while in many cases the relationship is nonlinear. SVMs can be extended to handle nonlinear data sets via a procedure called the "kernel trick". For the purposes of this discussion, we will work with the classification model (1.10) and its dual program (1.14), though the results could be easily extended to the regression models as well.

The most straightforward approach would be to map the input space (denoted $\mathcal{X}$) to a new feature space such that linear relationships can be sought in the new space. Thus, we consider an embedding map $\phi : \mathbb{R}^n \to \mathbb{R}^N$ which transforms the input data $\{(\mathbf{x}^1, y_1), ..., (\mathbf{x}^\ell, y_\ell)\}$ to $\{(\phi(\mathbf{x}^1), y_1), \ldots, (\phi(\mathbf{x}^\ell), y_\ell)\}$ and nonlinear relations to linear ones. The new space is called the feature space. The effect of this transformation is that the new function to be learned is of the form $\mathbf{w}'\phi(\mathbf{x}) - b$.

Direct transformation in the primal (1.10), however, is not a viable approach as the mapping would be very expensive, especially for large N. In addition, all future data would have to be mapped to the feature space in order to use the learned function. These

issues can be circumvented if we considered transformation within the dual (1.14). A glance at (1.14) shows it makes use of the inner products $(\mathbf{x}^i)'\mathbf{x}^j$ in the input space, or after the transformation, $\phi(\mathbf{x}^i)'\phi(\mathbf{x}^j)$, in the feature space.

Usually, the complexity of computing the inner products is proportional to the dimension of the feature space, N. However, for certain appropriate choices of ϕ, the inner products $\phi(\mathbf{x}^i)'\phi(\mathbf{x}^j)$ can be computed far more efficiently as a *direct function of the input features* and *without explicitly computing the mapping* ϕ. A function that performs this is called a *kernel function*.

Definition 1.1.5 (Kernel function)**.** *A kernel, κ, is a function that satisfies for all $\mathbf{x}$, $\mathbf{z} \in \mathcal{X}$, $\kappa(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})'\phi(\mathbf{z})$, where ϕ is a mapping from the input space $\mathcal{X}$ to the feature space $\mathcal{F}_\kappa$.*

Consider a two-dimensional input space $\mathcal{X} \subseteq \mathbb{R}^2$ together with a feature map $\phi : \mathbf{x} = (x_1, x_2) \rightarrow \phi(\mathbf{x}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2) \in \mathcal{F}_\kappa = \mathbb{R}^3$. The hypothesis space of linear functions in $\mathcal{F}_\kappa$ would be

$$g(\mathbf{x}) = w_{11}x_1^2 + w_{22}x_2^2 + w_{12}\sqrt{2}x_1x_2.$$

The feature map takes data from a two dimensional space to a three-dimensional space such that quadratic functional relationships in $\mathcal{X}$ correspond to linear functional relationships in $\mathcal{F}_\kappa$. We have

$$\begin{aligned}\langle\phi(\mathbf{x})'\phi(\mathbf{z})\rangle =& \langle(x_1^2, x_2^2, \sqrt{2}x_1x_2),\ (z_1^2, z_2^2, \sqrt{2}z_1z_2)\rangle \\ =& x_1^2z_1^2 + x_2^2z_2^2 + 2x_1x_2z_1z_2) \\ =& (x_1z_1 + x_2z_2)^2 \\ =& \langle\mathbf{x},\ \mathbf{z}\rangle^2.\end{aligned}$$

Hence the function $\kappa(\mathbf{x}, \mathbf{z}) = \langle\mathbf{x},\ \mathbf{z}\rangle^2$ is a kernel function with $\mathcal{F}_\kappa$ as its corresponding feature space (see Figure 1.7). It is important to note that the feature space is not uniquely determined by the kernel function. For example, the same kernel above computes the inner product corresponding to the four-dimensional mapping $\phi : \mathbf{x} = (x_1, x_2) \rightarrow \phi(\mathbf{x}) = (x_1^2, x_2^2, x_1x_2, x_2x_1) \in \mathcal{F}_\kappa = \mathbb{R}^4$.

Now, any linear model which depends solely on the inner products information of the data (such as the SVM dual) can be extended to handle nonlinear data sets by performing the *kernel trick*, i.e., replacing $\mathbf{x}'\mathbf{z}$ with a suitable kernel function, $\kappa(\mathbf{x}, \mathbf{z})$. Care must

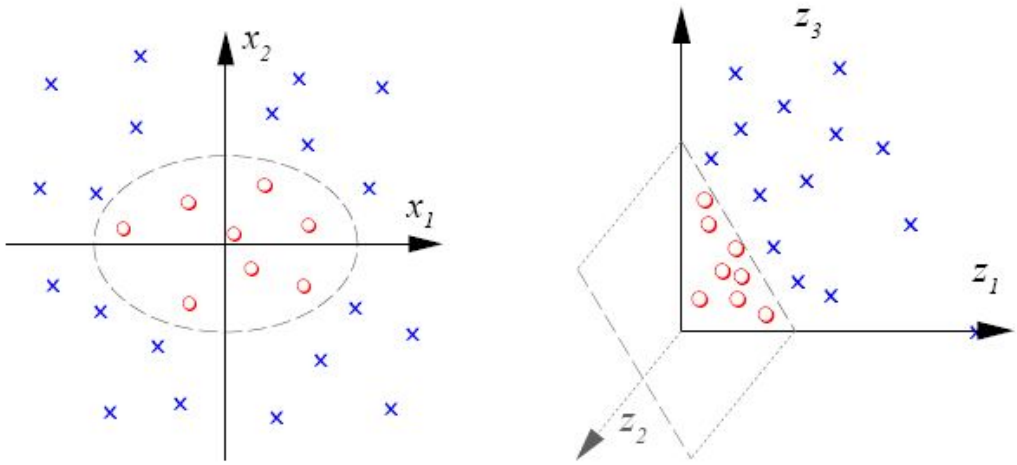


Figure 1.7: Transformation $\phi : \mathbf{x} = (x_1, x_2) \rightarrow \phi(\mathbf{x}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2) \in \mathcal{F}_\kappa$ mapping a quadratic relationship in $\mathbb{R}^2$ to a linear relationship in $\mathbb{R}^3$

be taken in picking kernel functions: they are required to be finite, symmetric and positive semi-definite i.e., satisfy Mercer's Theorem which is fundamental to interpreting kernels as inner products in a feature space [1]. For more on the theory and characterization of kernels see [81]. Some commonly used and well-known kernels are

- Linear kernel: $\kappa(\mathbf{x}, \mathbf{z}) = \langle \mathbf{x}, \mathbf{z} \rangle$
- Polynomial kernel: $\kappa(\mathbf{x}, \mathbf{z}) = (\langle \mathbf{x}, \mathbf{z} \rangle + c)^d, c, d \geq 0$
- Gaussian kernel: $\kappa(\mathbf{x}, \mathbf{z}) = e^{-\frac{\|\mathbf{x}-\mathbf{z}\|^2}{\sigma}}, \sigma > 0$
- Sigmoid kernel: $\kappa(\mathbf{x}, \mathbf{z}) = \tanh^{-1} \eta \langle \mathbf{x}, \mathbf{z} \rangle + \theta$

Kernels can also be constructed from other kernels:

- Conical combinations, $\kappa(\mathbf{x}, \mathbf{z}) = a_1\kappa_1(\mathbf{x}, \mathbf{z}) + a_2\kappa_2(\mathbf{x}, \mathbf{z})$, $a_1, a_2 \geq 0$
- Products of kernels, $\kappa(\mathbf{x}, \mathbf{z}) = \kappa_1(\mathbf{x}, \mathbf{z})\kappa_2(\mathbf{x}, \mathbf{z})$
- Products of functions, $\kappa(\mathbf{x}, \mathbf{z}) = f_1(\mathbf{x})f_2(\mathbf{z})$, f_1, f_2 are real valued functions.

Given training data $\{\mathbf{x}^1, \cdots, \mathbf{x}^\ell\}$ and a kernel function $\kappa(\cdot, \cdot)$, we can construct a symmetric positive semi-definite *kernel matrix*, K, with entries

$$K_{ij} = \kappa(\mathbf{x}^i, \mathbf{x}^j), \forall\, i, j = 1, \cdots, \ell.$$

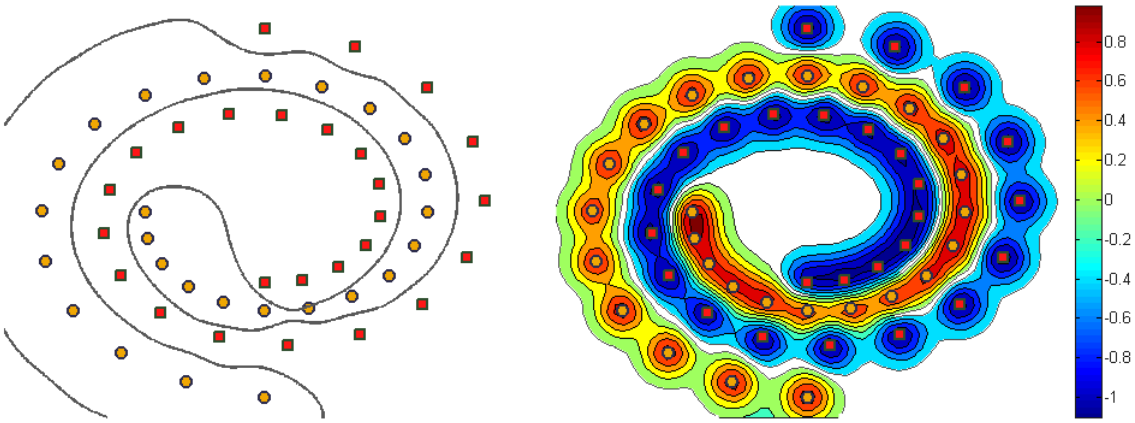


Figure 1.8: A Gaussian kernel used to train a classifier on a highly nonlinear data set (left). The effects of over-fitting are clearly visible (right)

The kernel matrix contains all the information available in order to perform the learning step, with the exception of the training labels. One other way of looking at a kernel matrix is as a pairwise similarity measure between the inputs. These are all key concerns when training in order to improve generalization performance. More specifically, since most kernels are parameterized, the choice of kernel parameter becomes very important with a poor choice leading to either under- or over-fitting and consequently poor generalization (see Figure 1.8).

1.2 Background: Bilevel Optimization

In this subsection, we introduce the bilevel methodology by means of a brief historical perspective. Succinctly, bilevel programs are a class of hierarchical optimization problems in variables x and y, with the optimal x being chosen by solving a constrained optimization problem whose constraints themselves are optimization problems in y, or possibly both x and y. In operations research literature, the class of bilevel optimization problems was introduced by Bracken and McGill [12], and applied to defense problems like minimum-cost weapon mix and economic problems like optimal production and marketing decision making models. Their work is closely related to the extensively studied economic problem of the Stackelberg game [85], whose origin predates the work of Bracken and McGill.

1.2.1 Stackelberg Games and Bilevel Programming

Stackelberg used a hierarchical model to describe the market situation where different decision makers try to optimize their decisions based on individually different objectives according to some hierarchy. The Stackelberg game can be considered an extension of the well-known Nash game. In the Nash game, there are T players, each of whom has a strategy set, Y_t, and the objective of player t is chose a strategy, $y_t \in Y_t$, given that the other players have already chosen theirs, to minimize some utility function. Thus, each player chooses a strategy based on the choices of the other players and there is no hierarchy.

In contrast, in the Stackelberg game, there is a hierarchy where a distinctive player, called the *leader* is aware of the choices of the other players, called the *followers*. Thus, the leader, being in a superior position with regard to everyone else can achieve the best objective while forcing the followers to respond to this choice of strategy by solving the Stackelberg game. Consider the case of a single leader and follower. Let X and Y denote the strategy sets for the leader and follower; let $F(x, y)$ and $f(x, y)$ be their utility functions respectively. Based on the selection, x, of the leader, the follower can select the best strategy $y(x) \in Y$ such that $f(x, y)$ is maximized i.e.,

$$y(x) \in \Psi(x) = \arg\max_{y \in Y} f(x, y). \tag{1.21}$$

The leader then computes the best strategy $x \in X$ as (see Figure 1),

$$x \equiv \max_{x \in X} \{F(x, y) \mid y \in \Psi(x)\}. \tag{1.22}$$

Equations (1.21) and (1.22) can be combined to express the Stackelberg game compactly as

$$\begin{aligned} &\max_{x \in X,\, y} && F(x, y) \\ &\text{s.t.} && y \in \arg\max_{\eta \in Y} f(x, \eta). \end{aligned} \tag{1.23}$$

Bilevel programs are more general than Stackelberg games in the sense that the strategy sets, also known as admissible sets, can depend on both x and y. This leads us to the

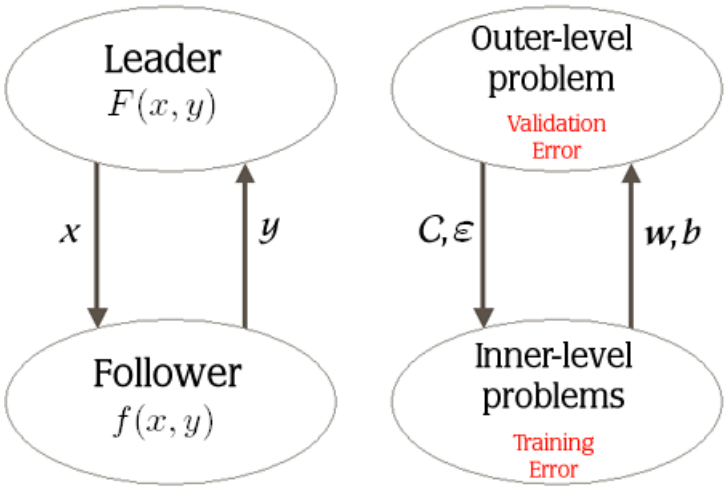


Figure 1.9: The Stackelberg game (left), showing the hierarchy between the leader and the follower; Cross validation modelled as a bilevel program (right), showing the interaction between the parameters, which are optimized in the outer level and the models which are trained in the inner level.

general bilevel program formulated by Bracken and McGill:

$$\begin{aligned} &\max_{x \in X,\, y} && F(x,y) && && \text{outerlevel} \\ &\text{s.t.} && G(x,y) \leq 0, \\ &y \in && \left\{ \begin{array}{ll} \arg\max_{y \in Y} & f(x,y) \\ \text{s.t.} & g(x,y) \leq 0 \end{array} \right\}. && && \text{innerlevel} \end{aligned} \tag{1.24}$$

The bilevel program, (1.24), is a generalization of several well-known optimization problems as noted in [22]. If $F(x,y) = -f(x,y)$, we have the classical minimax problem; if $F(x,y) = f(x,y)$, we have a realization of the decomposition approach to optimization problems; if the dependence of both problems on y is dropped, we have bicriteria optimization.

1.2.2 Bilevel Programs and MPECs

We consider bilevel programs of the type shown below, which is slightly different from the Bracken and McGill formulation, (1.24),

$$\begin{array}{ll} \min\limits_{x,y} & F(x,y) \\ \text{s. t.} & G(x,y) \geq 0, \\ & y \in \left\{ \begin{array}{ll} \arg\min\limits_{y} & f(x,y) \\ \text{s.t} & g_i(x,y) \geq 0, \quad \forall i = 1 \ldots m \end{array} \right\}. \end{array} \tag{1.25}$$

Introducing Lagrange multipliers, $\lambda_i \geq 0$, for the inner-level constraints, (1.25) can be rewritten using either the first-order KKT conditions or a variational inequality as follows:

$$\begin{array}{ll} \min\limits_{x,y} & F(x,y) \\ \text{s. t.} & G(x,y) \geq 0, \\ & \nabla f(x,y) - \sum\limits_{i=1}^{m} \lambda_i \nabla g_i(x,y) = 0, \\ & 0 \leq \lambda_i \perp g_i(x,y) \geq 0, \ \forall i = 1 \ldots m. \\ & \qquad \min\limits_{x,y} \quad F(x,y) \\ & \qquad \text{s. t.} \quad G(x,y) \geq 0, \\ & \Longleftrightarrow \\ & \qquad\qquad (u-y)' \nabla f(x,y) \geq 0, \quad \text{for some } y, \\ & \qquad\qquad u \in \{ y \mid g_i(x,y) \geq 0, \ \forall i = 1 \ldots m \}. \end{array} \tag{1.26}$$

The two formulations above are equivalent nonlinear programs; we shall use the one with the inner-level KKT conditions. The constraints of the form $0 \leq \lambda_i \perp g_i(x,y) \geq 0$ are called *equilibrium constraints* with $\lambda_i \perp g_i(x,y)$ meaning that $\lambda_i' g_i(x,y) = 0$. The presence of these equilibrium constraints makes the program an instance of a *Mathematical Program with Equilibrium Constraints* (MPEC). If the objective, $F(x,y)$, and the constraints, $g_i(x,y)$, are linear then the program becomes an instance of a *Linear Program with Equilibrium Constraints* (LPEC).

LPECs (or MPECs) are difficult to solve since they contain linear (or nonlinear) complementarity constraints; it is known that linear complementarity problems belong to the class of NP-complete problems [18]. Furthermore, the complementarity constraints

cause the feasible region of a bilevel program to lack closedness and convexity or, even possibly, be disjoint [60]. Aside from these obvious sources of intractability, stationary points for MPECs *always* fail to satisfy linear independence constraint qualification (LICQ) or Mangasarian-Fromovitz constraint qualification (MFCQ) in the nonlinear programming sense. There is yet another consideration, that of local optimal points, which is particularly important in the machine learning context. Machine learning problems lead to well-posed complementarity problems, in general, that have multiple local minima [61] which can be useful, especially if it is hard to construct globally optimal solutions

1.2.3 Stationarity Concepts for MPECs/LPECs

As noted above, LICQ or MFCQ, which are necessary to guarantee the existence of the multipliers, λ_i, at stationarity, fail to hold for (1.26) because the gradients of the complementarity constraints, $\lambda_i g_i(x, y) = 0$, are never linearly independent. Denoting the feasible region of the LPEC/MPEC (including the complementarities) is S_0, and the set of multipliers that satisfies the first-order KKT conditions of the inner-level problem is $\Lambda(x, y)$, we can define a key regularity assumption called the sequentially bounded constraint qualification (SBCQ).

Definition 1.2.1 (SBCQ). *For any convergent subsequence $\{(x^k, y^k)\} \subseteq S_0$, there exists, for each k, a multiplier vector, $\lambda^k \in \Lambda(x^k, y^k)$, and $\{\lambda^k\}_{k=1}^{\infty}$ is bounded.*

If SBCQ is satisfied, then it guarantees the non-emptiness of the set of multipliers, $\Lambda(x, y)$, and the existence of bounds on the multipliers on bounded sets. More importantly, it also guarantees the equivalence of (1.25) and (1.26) with regard to global optima; equivalence with regard to local optima can also be guaranteed if the functions $g_i(x, y)$ are convex in y. The SBCQ condition is weak and is easily satisfied under (implied by) other stronger constraint qualifications for the inner-level problem such as MFCQ.

In order to derive stationarity conditions for the MPEC, (1.26), we can relate it to the tightened and relaxed non-linear programs, where the first-order equality constraints

have been collected into $H(x, y, \lambda)$,

$$\begin{array}{lll} \min\limits_{x,y} & F(x,y) & \text{(tightened)} \\ \text{s. t.} & G(x,y) \geq 0,\ H(x,y,\lambda) = 0, & \\ & \lambda_i = 0, & \forall i \in \mathcal{I}_\alpha, \\ & g_i(x,y) = 0, & \forall i \in \mathcal{I}_\gamma, \\ & \lambda_i = 0,\ g_i(x,y) = 0, & \forall i \in \mathcal{I}_\beta. \end{array} \qquad \begin{array}{lll} \min\limits_{x,y} & F(x,y) & \text{(relaxed)} \\ \text{s. t.} & G(x,y) \geq 0, H(x,y,\lambda) = 0, & \\ & \lambda_i = 0, & \forall i \in \mathcal{I}_\alpha, \\ & g_i(x,y) = 0, & \forall i \in \mathcal{I}_\gamma, \\ & \lambda_i \geq 0,\ g_i(x,y) \geq 0, & \forall i \in \mathcal{I}_\beta. \end{array} \tag{1.27}$$

and with the Lagrangian function,

$$\begin{aligned} &\mathcal{L}(x, y, \lambda_i, \mu, \nu, u, v) = \\ &\qquad F(x,y) - \mu G(x,y) - \nu H(x,y,\lambda) - \sum_{i=1}^{m} u_i \lambda_i - \sum_{i=1}^{m} v_i g_i(x,y), \end{aligned} \tag{1.28}$$

where

$$\begin{aligned} \mathcal{I}_\alpha &:= \{i \mid \lambda_i = 0,\ g_i(x,y) > 0\}, \\ \mathcal{I}_\beta &:= \{i \mid \lambda_i = 0,\ g_i(x,y) = 0\}, \\ \mathcal{I}_\gamma &:= \{i \mid \lambda_i > 0,\ g_i(x,y) = 0\}. \end{aligned} \tag{1.29}$$

If the index set, $\mathcal{I}_\beta$, is empty, then *strict complementarity* is said to hold and if not, the complementarity constraints in $\mathcal{I}_\beta$ are said to be *degenerate*. We can now define some stationarity concepts.

Definition 1.2.2 (B-stationarity)**.** *A feasible point* (x^*, y^*, λ^*) *is said to be* Bouligand or B-stationary *if it is a local minimizer of an LPEC obtained by linearizing all the MPEC functions about the point* (x^*, y^*, λ^*) *i.e.,* $\nabla F(x,y)'z \geq 0$, $\forall z \in \mathcal{T}_{\text{lin}}(x^*, y^*, \lambda^*)$*, where* $\mathcal{T}_{\text{lin}}$ *denotes the tangent cone.*

This is a primal stationarity condition and is very general. However, as a certificate, it is not very useful as verifying it is combinatorially expensive due to the difficulty in characterizing the tangent cone. Alternately, we can look at various dual stationarity conditions.

Definition 1.2.3 (W-stationarity)**.** *A feasible point* (x^*, y^*, λ^*) *is said to be* weakly or

W-stationary *if there exist multipliers* μ, ν, u *and* $v \geq 0$ *such that*

$$\begin{aligned} &\nabla \mathcal{L}(x, y, \lambda_i, \mu, \nu, u, v) = 0, \\ &\mu \geq 0, \quad u_i = 0, \, \forall i \in \mathcal{I}_\gamma, \quad v_i = 0, \, \forall i \in \mathcal{I}_\alpha. \end{aligned} \tag{1.30}$$

The conditions above are simply the non-trivial first-order KKT conditions of the tightened nonlinear program. W-stationarity is a very important concept for computational purposes as it can help identify points that are feasible but not stationary[2].

Definition 1.2.4 (S-stationarity)**.** *A feasible point* (x^*, y^*, λ^*) *is said to be* strongly or S-stationary *if the W-stationarity conditions, (1.30), and the condition:* $\forall i \in \mathcal{I}_\beta$, $u_i, v_i \geq 0$, *hold.*

As in the weak case, the conditions for S-stationarity are simply the first-order KKT conditions for the relaxed nonlinear program. Finally, it can be shown that if "LICQ for MPECs" holds, then B-stationarity is equivalent to S-stationarity [77]. This discussion can be easily extended to the case where the outer-level problem may have equality constraints.

1.2.4 Optimization Methods for Bilevel Models

The bilevel and mutlilevel model selection models proposed here require the solutions of LPECs/MPECs. There exist several approaches that can deal with the complementarity constraints that arise in and LPECs/MPECs. Some of these are: penalty methods, which allow for the violation of the complementarity constraints, but penalize them through a penalty term in the outer-level objective; smoothing methods, that construct smooth approximations of the complementarity constraints; and relaxation methods, that relax the complementarity constraints while retaining the relaxations in the constraints. We now discuss some approaches to solving MPECs.

[2] W-stationarity concepts can be strengthened by enforcing additional constraints on the multipliers in (1.28). For example, replacing $\lambda_i g_i(x, y) = 0$ with $\min(\lambda_i, \, g_i(x, y)) = 0$ in (1.26) yields a non-smooth nonlinear program. The first-order KKT conditions for the latter can be written using the Clarke generalized gradient, and are precisely the conditions for Clarke or C-stationarity. See [89] for more details.

1.2.4.1 Nonlinear Programming Approaches

In machine learning, since the inner level problems are typically linear or quadratic, the reformulated bilevel program, yields an LPEC of the following general form

$$\begin{aligned} \min_{x,y} \quad & c'x + d'y \\ \text{s. t.} \quad & 0 \le y \perp w = Nx + My + q \ge 0, \\ & Ax + By + p \ge 0, \\ & Gx + Hy + f = 0. \end{aligned} \tag{1.31}$$

where some subset of variables of y are the multipliers λ_i. The complementarity condition can also be expressed using $\min(y, w) = 0$. This equality condition is equivalent to $y - (y - w)_+ = 0$. Here, $\mathbf{r}_+ = \max(\mathbf{r}, 0)$, the componentwise plus function applied to some vector $\mathbf{r} \ge 0$.

1.2.4.2 Inexact Solutions

This solution approach can be thought of as similar to the well-known machine learning technique of *early stopping*. As mentioned before, inexact and approximate solutions as well as local minima yield fairly good optimal points in the machine learning context. We take advantage of this fact and use the relaxation approach to solve MPECs. This method simply involves replacing all instances of "hard" complementarity constraints of the form

$$0 \le y \perp w \ge 0 \quad \equiv \quad y \ge 0,\ w \ge 0,\ y'w = 0$$

with relaxed, "soft" complementarity constraints of the form

$$0 \le y \perp_{\mathbf{tol}} w \ge 0 \quad \equiv \quad y \ge 0,\ w \ge 0,\ y'w \le \mathbf{tol}$$

where $\mathbf{tol} > 0$ is some prescribed tolerance of the complementarity conditions. If the machine learning problem yields an LPEC, the resulting inexact formulation will be a quadratically constrained quadratic program. For general MPECs, the relaxation will be a nonlinearly constrained optimization problem which can be solved using off-the-shelf NLP solvers such as FILTER [31] or SNOPT [39], which are freely available on the NEOS server [20]. Both these solvers implement the sequential quadratic programming (SQP) method; FILTER uses trust-region based SQP while SNOPT uses line search based SQP.

Inexact cross validation is one of the methods used to solve bilevel models presented here (see Chapters 2 and 3, and also [7, 54]). In spite of the fact that FILTER provides no guarantee of global optimality and generally converges to locally optimal solutions, this method performed well with regard to generalization error, indicating that local optimal solutions can be practically satisfactory. The reported results also compared favorably with grid search techniques with regard to parameter and feature selection and objective values. However, they were more efficient than grid search, especially with regard to feature selection.

1.2.4.3 Smooth Approximations

The condition, $\min(y,\, Nx + My + q) = 0$, can be replaced by a function $\phi(y, w)$, possibly non-smooth, such that $\phi(y, w) = 0 \equiv 0 \leq y \perp w \geq 0$. The Fischer-Burmeister function [30], $\phi(y, w) = y + w - \sqrt{y^2 + w^2}$, is a non-smooth example of such a function. This function is smoothed using a parameter ϵ to give the smoothed Fischer-Burmeister function, $\phi(y, w) = y + w - \sqrt{y^2 + w^2 + \epsilon^2}$. The smoothed function is everywhere differentiable and yields the following approximation of (1.31):

$$\begin{aligned} \min_{x,y} \quad & c'x + d'y \\ \text{s. t.} \quad & w = Nx + My + q \geq 0,\ y \geq 0, \\ & Ax + By + p \geq 0, \\ & Gx + Hy + f = 0 \\ & y_i + w_i - \sqrt{y_i^2 + w_i^2 + \epsilon_k^2} = 0,\ \forall i = 1 \ldots m. \end{aligned} \tag{1.32}$$

Pang and Fukushima [36] showed that for decreasing values of ϵ_k, the sequence of stationary points to the nonlinear program (1.32), (x^k, y^k, w^k), converges to a B-stationary point, (x^*, y^*, w^*), if weak second order necessary conditions hold at each (x^k, y^k, w^k), and LICQ for MPECs holds at (x^*, y^*, w^*). Various methods can be used to solve the sequence of problems (1.32); for example, the sequential quadratic programming (SQP) algorithm [48].

Another approach that was proposed for nonlinear and mixed complementarity problems involves solving the non-smooth equation, $y = (y - w)_+$; the right hand side of the equation, $\max(y - w, 0)$, is not differentiable at zero, and can be replaced by an everywhere differentiable smooth approximation. Chen and Mangasarian [15] propose

several different smooth approximations to the max function generated from different parameterized probability density functions that satisfy certain consistency properties. One approximation generated from the smoothed Dirac delta function that is commonly used in neural network literature is

$$p(z,\alpha) = z + \frac{1}{\alpha}\log\left(1+e^{-\alpha z}\right), \quad \alpha > 0, \tag{1.33}$$

where α is some smoothing parameter. Now, the smoothed non-linear equation representing the complementarity system is $\phi(y,w) = y - p(y-w,\alpha) = 0$.

1.2.4.4 Exact Penalty Methods

Penalty and augmented Lagrangian methods have been widely applied to solving LPECs and MPECs [47]. These methods typically require solving an unconstrained optimization problem. In contrast, exact penalty methods penalize only the complementarity constraints in the objective:

$$\begin{aligned} \min_{x,y} \quad & c'x + d'y + \mu\,\phi(y,w) \\ \text{s. t.} \quad & w = Nx + My + q \geq 0,\ y \geq 0, \\ & Ax + By + p \geq 0, \\ & Gx + Hy + f = 0. \end{aligned} \tag{1.34}$$

One approach to solving exact penalty formulations like (1.34) is the successive linearization algorithm, where a sequence of problems with a linearized objective,

$$c'(x-x^k) + d'(y-y^k) + \mu\left(\partial_x\phi(y^k,w^k)(x-x^k) + \partial_y\phi(y^k,w^k)(y-y^k)\right) \tag{1.35}$$

is solved to generate the next iterate. The algorithm requires concavity of the objective (to guarantee the existence of vertex solutions at each iteration) and lower-boundedness of the objective. An example of a differentiable penalty function is $\phi(y,w) = y'w$. The resulting quadratic program can be solved using the Frank-Wolfe method [61].

Alternately, the concave penalty function, $\phi = (y,w) = \min(y,w)$, has also been proposed. Various approaches can be used to handle the non-smoothness of the penalized objective function arising from this choice of $\phi(y,w)$. The most straight-forward approach is to use successive linearization with the gradients in the linearized objective

being replaced by the supergradients [63],

$$\begin{aligned}
\partial_x \phi &= \sum_{j=1}^{m} \begin{cases} 0, & \text{if } y_j < w_j, \\ (1-\lambda_j)\, 0 + \lambda_j N_j, & \text{if } y_j = w_j, \\ N_j, & \text{if } y_j > w_j. \end{cases} \\
\partial_y \phi &= \sum_{j=1}^{m} \begin{cases} I_j, & \text{if } y_j < w_j, \\ (1-\lambda_j)\, I_j + \lambda_j M_j, & \text{if } y_j = w_j, \\ M_j, & \text{if } y_j > w_j. \end{cases}
\end{aligned} \tag{1.36}$$

and $0 \leq \lambda \leq 1$. A second approach makes use of the fact that $\min(r, s)$, for any two scalars, r and s, can be computed as

$$\min(r, s) = \arg\min_{\rho,\sigma} \{\rho r + \sigma s \mid \rho, \sigma \geq 0,\ \rho + \sigma = 1\}. \tag{1.37}$$

Incorporating this into (1.34) gives a separable bilinear program [62],

$$\begin{aligned}
\min_{x,y} \quad & c'x + d'y + \rho' r + \sigma' s \\
\text{s. t.} \quad & w = Nx + My + q \geq 0,\ y \geq 0, \\
& Ax + By + p \geq 0, \\
& Gx + Hy + f = 0.
\end{aligned} \tag{1.38}$$

which can be solved using a finite Frank-Wolfe method. A third approach requires replacing the non-smooth min with its smooth approximation, which can be defined analogous to the approximation for the max function shown in the previous subsection,

$$m(z, \alpha) = -\frac{1}{\alpha} \log\left(1 + e^{-\alpha z}\right), \quad \alpha > 0. \tag{1.39}$$

The application of these methods to the bilevel machine learning applications is presently under investigation.

1.2.4.5 Integer Programming Approaches

The connections between bilevel programs, MPECs and mixed integer programs (MIPs) are well known. It was shown in [5] that there exists a polynomial time reformu-

lation to convert a mixed integer program to a bilevel program. Also demonstrated in [5] was an implicit reformulation of a bilevel program as a mixed integer program via MPECs. Specifically, a program with equilibrium constraints, such as (1.31), can be converted to a MIP by splitting the complementarity constraints through the introduction of integer variables, z, and a large finite constant θ.

$$\begin{aligned} \min_{x,y} \quad & c'x + d'y \\ \text{s. t.} \quad & 0 \leq Nx + My + q \leq \theta(1 - z), \\ & 0 \leq y \leq \theta z, \quad z \in \{0,1\}^m, \\ & Ax + By + p \geq 0, \\ & Gx + Hy + f = 0. \end{aligned} \tag{1.40}$$

Care must be taken to compute the value of θ large enough so as not to cut off parts of the feasible region. This is done by solving several LPs to obtain bounds on all the variables and constraints of (1.40) and setting θ to be equal to the largest bound. Once θ is fixed, the MIP can now be solved by using standard techniques such as branch and bound.

The biggest drawback of this approach is that the computation of the bound, θ, requires solving a very large number of LPs. Other drawbacks are that the approach can only be applied to LPECs with bounded feasible regions (thus ensuring that the feasible region of the MIP is also bounded) and does not necessarily always converge to a global optimum. These latter limitations tend to be less of a concern for bilevel programs arising from machine learning applications. However, all of the drawbacks mentioned here are all satisfactorily dealt with in the method of [46], wherein a parameter-free dual program of (1.40) is derived, reformulated as a minimax problem, and solved using Bender's approach. The application of this method to the bilevel machine learning applications is presently under investigation.

1.2.4.6 Other Approaches

The discussion of the solution approaches above is not meant to be exhaustive. There are several other approaches to solving MPECs and LPECs such as active set identification methods [56], interior point methods [59, 60], implicit programming [17, 60] and non-smooth methods [67].

1.3 The Need for New Methodologies

We are now ready to discuss the factors that motivated the research presented here for the various machine learning models and the paradigm, based on bilevel optimization, that was used to approach them.

Two issues that have been around as long as SVMs are parameter and kernel selection, collectively called *model selection*. Support vector machines and kernels are parameterized; these parameters have to be set *a priori* and the choice of parameters (and consequently the model) dramatically affects generalization behavior. There have been many interesting attempts to pick these parameters; notable among these are approaches are approaches that use bounds [14], or trace the complete regularization path for either one parameter—as in the case of SV classification [44]—or two parameters—as in the case of SV regression [88]. However, the most systematic and commonly used method for selecting these hyper-parameters is T-fold cross validation [52] (CV). T-fold cross validation attempts to estimate generalization error by simulating out-of-sample testing on various subsets of the training data. CV leaves out subsets of the training data, trains models on the reduced sets of data, and then tests the resulting models on the left-out data. Cross validation can be applied to arbitrary machine learning problems and gives a good estimate of generalization error (even for small data sets) which shows a strong correlation with the test error [26].

Current model selection methods have several limitations. The main disadvantage of bound- or regularization-path-based methods is that they are restricted to being able to perform model selection for problems with only one, or at most two parameters. This limitation severely restricts the applicability of these methods to more difficult machine learning tasks that yield multi-parametric models such as feature and kernel selection, tasks that are very germane to effective many applications. CV itself suffers from a significant drawback as it involves implementing a *grid search*: training T models (for T-fold CV) at each point of a discretized parameter space in order to determine the best model. This process becomes prohibitively expensive as the training set size, dimensionality of the data set or the number of parameters grows, severely limiting the applicability of CV to smaller data sets. In addition, the discretization of the parameter space effectively limits the search for optimal parameters to a finite set of points and the overall quality of the "solution" cannot be guaranteed. Specifically for high-dimensional problems, other heuristic methods exist, such as stepwise regression, backward elimination, filter methods

and genetic algorithms, to name a few. However, these methods suffer the same limitations of grid search: practical inefficiency with regard to implementation and inability to guarantee the quality of the solution.

One of the key motivations for this work is to address the urgent need to develop improved methodologies that can combine model selection methods for different machine learning problems—such as the widely-accepted CV method—with the the sound theoretical foundations and robust efficiency of extant and emerging mathematical programming approaches.

In addition to model selection, some of the other issues that have been pervasive in machine learning or emerged recently include newer regularization techniques, feature selection for dimensionality reduction [9, 43], kernel construction [55, 66], complexity minimization, semi-supervised learning, predicting missing values in data, incorporating prior knowledge and multi-task learning [13, 28].

Another key motivation is to develop a general unified framework that facilitates easy incorporation of the various tasks mentioned above into model selection approaches such as CV.

The work presented here proposes and explores one such methodology that attempts to address the deficiencies of the current state-of-the-art: *bilevel programming.* Bilevel model selection offers several advantages over prior approaches. The most obvious advantage is the ability to deal with multi-parametric models and deal with them in continuous rather than discrete space. This is possible because of recent advances in bilevel programming in the optimization community, which permit the systematic treatment of models based on different loss and regularization functions and kernels. In addition to being able to incorporate existing methods, the bilevel approach offers a broad framework in which novel regularization methods and generalization measures can be developed. Most significantly, the bilevel approach provides a framework in which to combine model selection with feature selection, multi-task learning etc. As we will see, these advantages allow for the formulation of different machine learning tasks such as model, feature and kernel selection for classification and regression as well as for semi-supervised learning.

Yet another key motivation is to construct algorithms that can solve mathematical programs that arise from the model selection problems efficiently such that they are capable of handling large-scale data sets.

As mentioned above, the other main motivation was the development of efficient algorithms that do not suffer the combinatorial drawbacks of heuristic methods such as grid-search-based CV. From the optimization point of view, bilevel programs resulting from these applications belong to the general class of mathematical programs with equilibrium constraints (MPECs), [60], for which there are extensive advances in theory, algorithms, and software in recent years [3, 4, 22, 23, 33, 34, 37, 38, 48, 77, 79, 80]. These advances include efficient techniques such as filter-based sequential quadratic programming (SQP) methods and successive linearization algorithms which, as will been shown, can be successfully applied to solve the MPECs arising from the bilevel machine learning formulations efficiently and provide "good solutions".

1.4 A New Paradigm

The general predictive learning task is to construct a function using present data that performs well on future data. A loss function specific to the learning tasks is used to measure how well the function is performing. Cross validation (CV) is a method of estimating the out-of-sample generalization error of the model for given hyper-parameters. Cross validation leaves out subsets of the training data, trains models on the reduced sets of data, and then tests the resulting models on the left-out data. Cross validation can be applied to arbitrary machine learning problems, gives a very good estimate of generalization error (even for small data sets) which shows a strong correlation with the test error [26]. The CV step is typically followed by a post processing step in which the final model is trained on all the available data, using the "optimal" hyper-parameters given by CV, to build the final model (see Figure 1.10). The efficacy is this model may further be examined by observing its performance on a hold-out test set.

To perform model selection, CV must be embedded within an optimization algorithm. In the most common approach, Grid Search, CV is performed over a grid that discretizes the hyper-parameter space of interest and involves, for T folds, training T models at each grid point. As the number of hyper-parameters grows, so does the number of

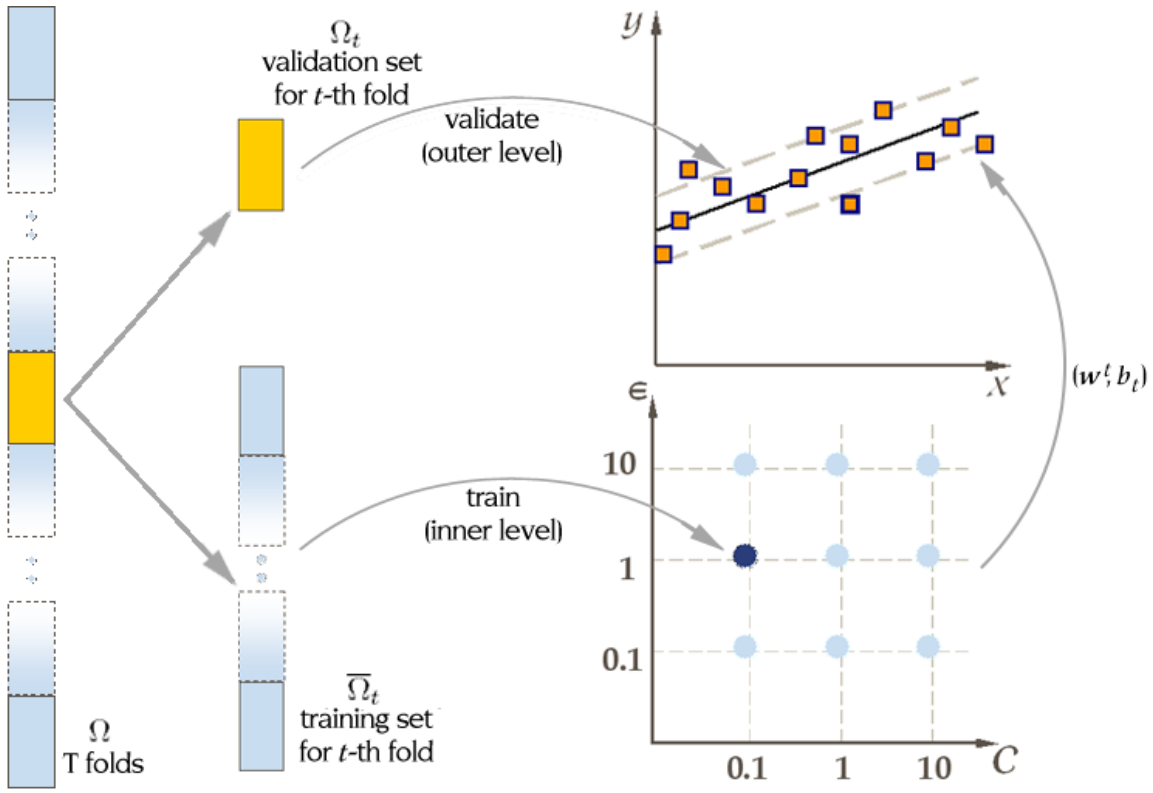


Figure 1.10: T**-fold cross validation at a particular grid point,** (C_k, ε_k)**, on a base-**10 **logarithmic grid.**

problems to be solved and cross validation becomes prohibitively expensive. Efficiency can only be achieved at the expense of grid refinement and coarser grids inevitably yield poor models. In fact, even for a small number of parameters, cross validation can still be expensive for high-dimensional data sets. For example, feature selection for high-dimensional data sets leads to a combinatorial explosion of grid points. Such problems are ubiquitous in machine learning e.g., in feature selection [9, 43], kernel construction [55, 66], and multi-task learning [13, 28].

Another drawback in grid search is that the discretization is restricted to examining only a finite set of points. Recent work on determining the full regularization path of support vector machines underscores the fact that regularization parameter is continuous. In particular, the paper [44] argues that the choice of the single regularization parameter, C, is critical and shows that it is quite tractable to compute the SVM solution for all possible values of the regularization parameter C. But as it is well known in optimization, this parametric programming approach for a single parameter is not extendable to models

with multiple parameters and certainly is not possible for models with a large number of parameters. Bayesian methods can treat model parameters as random variables but then the challenge becomes the choice of appropriate priors. In the end, out-of-sample testing is still the gold standard for selecting parameters values. The *bilevel optimization* methodology attempts to address this issues by formulating CV for SVMs as a bilevel program.

T-fold cross validation for some general machine learning problem depends on partitions of the data—the training sets, $\overline{\Omega}_t$, and validation sets, Ω_t, within the t-th fold—and can be written as

$$
\begin{aligned}
&\underset{f^t,\,\boldsymbol{\lambda}}{\text{minimize}} && \Theta(f^1|_{\Omega_1}, \ldots, f^T|_{\Omega_T}; \boldsymbol{\lambda}) && \text{(outer level)}\\
&\text{subject to} && \boldsymbol{\lambda} \in \Lambda, \\
& && \text{and for } t = 1, \ldots, T, \\
& && f^t \in \underset{f \in \mathcal{F}}{\arg\min} \left\{ \mathcal{P}(f, \boldsymbol{\lambda}) + \sum_{(\mathbf{x}^i, y_i) \in \overline{\Omega}_t} \mathcal{L}\left(y_i, f(\mathbf{x}^i), \boldsymbol{\lambda}\right) \right\}. && \text{(inner level)}
\end{aligned}
\tag{1.41}
$$

Here, $f^t : (\mathbb{R}^n \times \mathbb{R}) \cap \mathcal{F} \to \mathbb{R}$ is the learning function trained within the t-th fold, $\boldsymbol{\lambda} \in \Lambda$ is the set of model selection parameters for the machine learning problem, $\mathcal{P}$ is the regularization operator and $\mathcal{L}$ is the inner-level loss function. This definition admits many variations and well-known machine learning problems. In fact, the number of machine learning tasks that can be cast into the bilevel cross-validation framework is virtually limitless. These learning tasks determine the objective and constraints used in the inner-level problems and the outer-level objective i.e., the overall testing/generalization objective is minimized in the "outer" (or upper) level subject to the learning functions which are optimized in the "inner" (or lower) level. The resulting cross-validation problem can be re-formed as a bilevel optimization problem as long as the inner-level problems can be replaced by their corresponding KKT conditions, and the outer-level objective and constraints can be replaced by differentiable counterparts.

Equation (1.41) is the generalized template model that defines the underlying paradigm of this thesis i.e., cross-validation based model selection formulated as a bilevel program. Various problems such as model selection for support vector classification and regression, semi-supervised learning, missing-value imputation, kernel selection, multi-task learning and complexity minimization are described and explored in the following sec-

tions. Each of these problems can be cast in the bilevel framework (1.41) by appropriately choosing the regularization and loss functions. Then, the appropriate "parameters", $\boldsymbol{\lambda}$, can be found by optimizing them in the outer level to minimize some estimate of generalization error, Θ while simultaneously learning f^t on the training sets in the inner level. This interaction is exactly cross validation (see Figure 1.10) except that the parameters $\boldsymbol{\lambda}$ now vary smoothly and continuously rather than being restricted to (possibly coarse) discretization.

Bilevel model selection offers several advantages over prior approaches. The most obvious advantage is the ability to deal with multi-parametric model selection and deal with them in continuous rather than discrete space. This is possible because of recent advances in bilevel programming in the optimization community, which permit the systematic treatment of models based on different loss and regularization functions and kernels. In addition to being able to incorporate existing methods, the bilevel approach offers a broad framework in which novel regularization methods and generalization measures can be developed. Most significantly, these advantages allow for improved model selection which ultimately leads to better generalization.

1.5 Organization of the Thesis

The thesis is organized into the following sections. Chapter 2 introduces bilevel model selection by applying it to cross-validation-based simultaneous model and feature selection for support vector classification. The resulting LPECs are solved using FILTER, an freely available, off-the-shelf, NLP solver. This essentially serves as a proof of concept. Chapter 3 extends this approach to support vector regression. In addition to solving the resultant LPECs using FILTER, this chapter also introduces SLAMS, a successive linearization approach to solving LPECs that is particularly effective for solving machine learning problems to local optimality. Chapter 4 presents a novel approach to learning with missing data (features) through bilevel programming. Finally, Chapter 5 presents more machine learning models that can be solved using the bilevel approach as challenges that merit attention in the future and the concludes the thesis.

CHAPTER 2
SV Classification and Inexact Cross Validation

2.1 Introduction

Support Vector Machines (SVM) [19, 87] are one of the most widely used methods for classification. The underlying quadratic programming problem is convex (thus, is generally not difficult to deal with, both theoretically and computationally), but typically it contains *hyper-parameters* that must be selected by the users.

The models presented in this chapter nontrivially extend previous work [7]. In essence, unlike many traditional grid search methods used in machine learning that are severely restricted by the number of hyper-parameters to be searched, the bilevel approach enables the identification of many such parameters all at once by way of the state-of-the-art optimization methods and their softwares (such as those publicly available on the NEOS servers). Another important advantage of the bilevel approach is its modeling versatility in handling multiple machine learning goals simultaneously and efficiently; these include optimal choice of model parameters [14, 41], feature selection for dimension reduction [9], inexact cross validation, kernelization to handle nonlinear data sets [81], and variance control for fold consistency through multi-tasking.

The focus is on the bilevel binary classification problem where the main task is to classify data into two groups according to a linear model using a classical support-vector (SV) classifier [19]. The hyper-parameters are selected to minimize the T-fold cross validated estimate of the out-of-sample misclassification error. Each fold of training defines an inner-level convex quadratic program (QP) with parameters constrained by some bounds that are part of the overall variables to be optimized; such bounds provide a mechanism for feature selection whereby those features corresponding to small bounds in the solution of the bilevel problem will be deemed insignificant. The outer-level problem minimizes the T-fold average classification error based on the optimal solutions of the inner-level QPs for all possible hyper-parameters. Using the approach in [61], we add inner-level linear programs to compute the number of misclassified test points for each fold. In principle, the objective functions in the inner-level classification optimization problems could be rather general; the only restriction we impose is their convexity so that the only non-convexity generated by the inner-level problems in the MPEC is essentially

the complementarity slackness in the optimality conditions of the inner-level problems.

The organization of the chapter is as follows. The mathematical formulation of the bilevel cross validation classification model and its transformation to an instance of an MPEC is described in Section 2.2. Additional variations of the classification problems are described in Section 2.3. Section 2.4 contrasts grid search and with the bilevel approach, or more specifically, a relaxed nonlinear programming reformulation of the MPEC called inexact cross validation. Section 2.5, describes the experimental setup, the data sets used and computational results comparing the grid search and bilevel cross validation methods. Finally, conclusions are presented in Section 2.6.

2.2 Model Formulation

Let Ω denote a given finite set of $\ell = |\Omega|$ labeled data points, $\{(\mathbf{x}_i, y_i)\}_{i=1}^{\ell} \subset \mathbb{R}^{n+1}$. Since we are interested in the binary classification case, the labels y_i are ± 1. Let the set of indices for the points in Ω be $\mathcal{N} = \{1, \ldots, \ell\}$. For T-fold cross validation, Ω is partitioned into T pairwise disjoint subsets, Ω_t, called the *validation sets*. The sets $\overline{\Omega}_t = \Omega \setminus \Omega_t$ are the *training sets* within each fold. The corresponding index sets for the validation and training sets are $\mathcal{N}_t$ and $\overline{\mathcal{N}}_t$ respectively. The hyperplane trained within the t-th fold using the training set $\overline{\Omega}_t$ is identified by the pair $(\mathbf{w}^t, b_t) \in \mathbb{R}^{n+1}$. For compactness of notation, the vectors $\mathbf{w}^t$ are collected, column-wise, into the matrix $\mathbf{W} \in \mathbb{R}^{n \times T}$, and the scalars b_t into the vector $\mathbf{b} \in \mathbb{R}^T$. A vector of ones of arbitrary dimension is denoted by $\mathbb{1}$. Given two vectors, $\mathbf{r}$ and $\mathbf{s} \in \mathbb{R}^n$, the complementarity condition $\mathbf{r} \perp \mathbf{s}$ means $\mathbf{r}'\mathbf{s} = 0$, where the prime $'$ denotes the transpose; and $\mathbf{r}_\star$ denotes the step function applied to each component of the vector $\mathbf{r}$ as defined in (2.7).

The well-known SV classification problem depends on a nonnegative regularization parameter, λ, which is selected through cross validation based on the average misclassification error measured on the out-of-sample data, i.e, the validation sets. Specifically, the basic bilevel model for support vector classification, is formulated as follows:

$$\begin{aligned} \underset{\mathbf{W},\mathbf{b},\lambda,\overline{\mathbf{w}}}{\text{minimize}} \quad & \Theta(\mathbf{W}, \mathbf{b}) \\ \text{subject to} \quad & \lambda_{\mathbf{lb}} \le \lambda \le \lambda_{\mathbf{ub}}, \quad \overline{\mathbf{w}}_{\mathbf{lb}} \le \overline{\mathbf{w}} \le \overline{\mathbf{w}}_{\mathbf{ub}}, \qquad (2.1) \\ & \text{and for } t = 1, \ldots, T, \\ & (\mathbf{w}^t, b_t) \in \underset{\substack{-\overline{\mathbf{w}} \le \mathbf{w} \le \overline{\mathbf{w}} \\ b \in \mathbb{R}}}{\arg\min} \left\{ \frac{\lambda}{2} \| \mathbf{w} \|_2^2 + \sum_{j \in \overline{\mathcal{N}}_t} \max\left(1 - y_j(\mathbf{x}_j'\mathbf{w} - b), 0\right) \right\}. \qquad (2.2) \end{aligned}$$

We can see that the program above is a specific case of the template (1.41), with L_2 norm regularization and the hinge-loss function. It is clear that, with regard to the general model, the parameters λ and $\overline{\mathbf{w}}$ are outer level hyper-parameters in this model. The outer-level objective, $\Theta(\mathbf{W}, \mathbf{b})$, is some measure of validation accuracy over all the folds, typically the average number of misclassifications. There are T inner-level subproblems, one for each fold in T-fold cross validation. The arg min is the last constraint in (2.1) and denotes the set of all optimal solutions to the T convex optimization problems (2.2). Each t-th subproblem is simply a classical support vector classification problem applied to the corresponding training set, $\overline{\Omega}_t$, along with the additional box constraint of the form $-\overline{\mathbf{w}} \leq \mathbf{w} \leq \overline{\mathbf{w}}$, where $\overline{\mathbf{w}}$ is a variable in the overall bilevel optimization; in turn, $\overline{\mathbf{w}}$ is restricted to given bounds $\overline{\mathbf{w}}_{\mathbf{ub}} \geq \overline{\mathbf{w}}_{\mathbf{lb}} \geq 0$.

The symmetric box constraint is included for the purposes of wrapper-type feature selection and regularization. In addition, the box constraint was selected to illustrate that the bilevel approach can successfully optimize many hyper-parameters. Note that there is one box constraint parameter for every descriptor. Consider a particular feature that is expected to be redundant or irrelevant i.e., that feature does not contribute much to the final classifier. Then, the corresponding weight in $\overline{\mathbf{w}}$ would be small or zero, which, in turn, constrains the corresponding weights in each $\mathbf{w}^t$, thereby effectively controlling their capacity and potentially increasing generalization performance. The bilevel program will effectively be a wrapper feature selection method. Wrapper methods search for subsets of feature that optimize estimates of the testing generalization error for models trained with those features [53]. Sparse 1-norm regularization can also be used for feature selection but the subset features in each of the CV folds illustrates great variability [9]. The box constraints will ensure that a consistent subset of the features will be used across all the folds. If $\overline{\mathbf{w}}$ can be picked effectively, the box constrained SVM, (2.2), could represent a fundamentally new way to perform feature selection. Thus, the box constraints are embedded in the bilevel cross validation scheme and $\overline{\mathbf{w}}$ becomes a vector of hyper-parameters in the problem.

Note that we use $\frac{\lambda}{2}\|\mathbf{w}\|_2^2$ for regularization rather than the typical term $C\sum_{j\in\overline{\mathcal{N}}_t} \max(1 - y_j(\mathbf{x}_j'\mathbf{w} - b), 0)$, where C is the parameter to be chosen. This is due to our empirical observation that the former is more numerically robust than the latter within the bilevel setting. Clearly, the two modes of regularization are equivalent for each inner-level problem with $C = \frac{1}{\lambda}$, provided that both parameters are positive. Note that the

bilevel program selects the hyperparameters. The final classifier can be constructed by optimizing a single instance of the lower-level problem using the optimal hyper-parameters and all of the training data. In this case, the final λ should be scaled by $\frac{T}{T-1}$ to account for the larger training set size.

Similar to $\overline{\mathbf{w}}$, the parameter λ is subject to given bounds $\lambda_{\mathsf{ub}} \geq \lambda_{\mathsf{lb}} > 0$. This is done for three reasons: first, to facilitate direct comparison with grid search based cross validation (see Section 2.4.2); second, to improve the stability and speed up convergence of a general purpose NLP solver; and third, to ensure the positivity of the parameters λ and $\overline{\mathbf{w}}$: so that the bilevel approach yields, in case of the former, a nonzero regularization parameter, in case of the latter, a nontrivial box constraint for feature selection.

2.2.1 The inner-level problems

As mentioned above, there are T inner-level problems that model the training of classifiers within each fold. Consider the inner-level problem corresponding to the t-th fold i.e., the t-th training set, $\overline{\Omega}_t$, indexed by $\overline{\mathcal{N}}_t$, is used. With λ and $\overline{\mathbf{w}}$ fixed in this subproblem, we introduce slack variables, $\boldsymbol{\xi}^t$, in (2.2) to reformulate the max function using standard linear programming techniques. This gives the the box-constrained SV classifier (BoxSVC) which is nearly identical to the classical SVM for classification, except that it has the additional box constraint for regularization and feature selection:

$$\begin{aligned} \underset{\mathbf{w}^t,\, b_t,\, \boldsymbol{\xi}^t}{\text{minimize}} \quad & \frac{\lambda}{2}\left\|\mathbf{w}^t\right\|_2^2 + \sum_{j \in \overline{\mathcal{N}}_t} \xi_j^t \\ \text{subject to} \quad & -\overline{\mathbf{w}} \leq \mathbf{w}^t \leq \overline{\mathbf{w}}, \\ & \left.\begin{aligned} & y_j(\mathbf{x}_j'\mathbf{w}^t - b_t) \geq 1 - \xi_j^t \\ & \xi_j^t \geq 0 \end{aligned}\right\} \forall\, j \in \overline{\mathcal{N}}_t. \end{aligned} \tag{2.3}$$

The BoxSVC is a convex quadratic program in the variables $\mathbf{w}^t$, b_t and $\{\xi_j^t\}_{j \in \overline{\mathcal{N}}_t}$. Let $\boldsymbol{\gamma}^{t,-}$ and $\boldsymbol{\gamma}^{t,+}$ be the multipliers of the lower and upper bound constraints $-\overline{\mathbf{w}} \leq \mathbf{w}^t \leq \overline{\mathbf{w}}$ respectively and α_j^t be the multiplier for the hyperplane constraint, $y_j(\mathbf{x}_j'\mathbf{w}^t - b_t) \geq 1 - \xi_j^t$. Using these multipliers, we can write down the primal and dual feasibility and

complementarity slackness conditions of (2.3) compactly as follows:

$$\left.\begin{array}{l} 0 \le \alpha_j^t \quad \perp y_j(\mathbf{x}_j'\mathbf{w}^t - b_t) - 1 + \xi_j^t \ge 0 \\ 0 \le \xi_j^t \quad \perp 1 - \alpha_j^t \ge 0 \end{array}\right\} \forall\, j \in \overline{\mathcal{N}}_t, \qquad (2.4)$$

$$0 \le \boldsymbol{\gamma}^{t,+} \perp \overline{\mathbf{w}} - \mathbf{w}^t \ge 0,$$

$$0 \le \boldsymbol{\gamma}^{t,-} \perp \overline{\mathbf{w}} + \mathbf{w}^t \ge 0,$$

which together with the following first-order conditions,

$$\begin{array}{l} \lambda\, \mathbf{w}^t - \displaystyle\sum_{j \in \overline{\mathcal{N}}_t} y_j \alpha_j^t \mathbf{x}_j + \boldsymbol{\gamma}^{t,+} - \boldsymbol{\gamma}^{t,-} = 0, \\ \displaystyle\sum_{j \in \overline{\mathcal{N}}_t} y_j \alpha_j^t = 0, \end{array} \qquad (2.5)$$

constitute the Karush-Kuhn-Tucker (KKT) optimality conditions to (2.3). The KKT conditions are necessary and sufficient conditions for the optimal solution of (2.3). Thus the inner-level optimization problems (2.2) can be replaced with the system of equations (2.4) and (2.5).

2.2.2 The outer-level optimization

The inner-level problems solve T box-constrained SV classification problems on the training sets to yield T hyperplanes, $(\mathbf{w}^t, b_t)$, one for each fold. The outer-level objective function is a measure of generalization error based on the T out-of-sample validation sets, which we minimize. The measure used here is the classical cross-validation error for classification, the average number of points misclassified. The outer-level objective that achieves this can be written using the step function, $()_\star$, as

$$\Theta(\mathbf{W}, \mathbf{b}) = \frac{1}{T} \sum_{t=1}^{T} \frac{1}{|\mathcal{N}_t|} \sum_{i \in \mathcal{N}_t} \left[-y_i(\mathbf{x}_i'\mathbf{w}^t - b_t)\right]_\star. \qquad (2.6)$$

Note that in the inner summation, Ω_t, the t-th validation set, indexed by $\mathcal{N}_t$, is used. The inner summation averages the number of misclassifications within each fold while the outer summation averages the averaged misclassification error over the folds. The step

function used in (2.6) can be defined, componentwise, for a vector, $\mathbf{r}$, as

$$(\mathbf{r}_\star)_i = \begin{cases} 1, & \text{if } r_i > 0, \\ 0, & \text{if } r_i \leq 0. \end{cases} \tag{2.7}$$

It is clear that $()_\star$ is discontinuous and that (2.6) cannot be used directly in the bilevel setting. The step function, however, can be characterized as the solution to a linear program as demonstrated in [61], i.e.,

$$\mathbf{r}_\star = \arg\min_{\boldsymbol{\zeta}} \{-\boldsymbol{\zeta}'\mathbf{r} : 0 \leq \boldsymbol{\zeta} \leq \mathbb{1}\}. \tag{2.8}$$

Thus, we have to solve T linear programs of the form (2.9) to determine which validation points, $\mathbf{x}_i \in \mathcal{N}_t$, are misclassified within the t-th fold, i.e., when the sign of $y_i(\mathbf{x}_i'\mathbf{w}^t - b_t)$ is *negative*. These LPs are inserted as inner-level problems into the bilevel setting in order to recast the discontinuous outer-level objective into a continuous one. They yield $\boldsymbol{\zeta}^t = \left[-y_i(\mathbf{x}_i'\mathbf{w}^t - b_t)\right]_\star$, with $\zeta_i^t = 1$ if the point $\mathbf{x}_i$ is misclassified and 0 otherwise. Finally, it should be noted that if $\mathbf{x}_i$ lies on the hyperplane, $(\mathbf{w}^t, b_t)$, then we will have $0 < \zeta_i^t < 1$.

$$\boldsymbol{\zeta}^t \in \arg\min_{0 \leq \boldsymbol{\zeta} \leq \mathbb{1}} \left\{ \sum_{i \in \mathcal{N}_t} \zeta_i y_i \left(\mathbf{x}_i'\mathbf{w}^t - b_t\right) \right\}. \tag{2.9}$$

Returning to the general case, we introduce additional multipliers, $\mathbf{z}$, for the constraint $\boldsymbol{\zeta} \leq \mathbb{1}$. Consequently, any solution to (2.8) should satisfy the following linear complementarity conditions:

$$\begin{aligned} 0 \leq \boldsymbol{\zeta} \perp -\mathbf{r} + \mathbf{z} \geq 0, \\ 0 \leq \mathbf{z} \perp \ 1 - \boldsymbol{\zeta} \geq 0. \end{aligned} \tag{2.10}$$

We noted in Section 2.2.1 that the inner-level problems, (2.2), can be replaced with the first-order KKT conditions, (2.4–2.5). Furthermore, the inner-level step function LPs, (2.9), can be rewritten using the linear complementarity conditions, (2.10). The overall

two-level classification problem becomes

$$
\begin{aligned}
\min \quad & \frac{1}{T}\sum_{t=1}^{T}\frac{1}{|\overline{\mathcal{N}}_t|}\sum_{i\in\mathcal{N}_t}\zeta_i^t \\
\text{s. t.} \quad & \lambda_{\mathbf{lb}} \le \lambda \le \lambda_{\mathbf{ub}},\ \overline{\mathbf{w}}_{\mathbf{lb}} \le \overline{\mathbf{w}} \le \overline{\mathbf{w}}_{\mathbf{ub}}, \\
& \text{and for } t = 1 \dots T, \\
& \left.\begin{aligned}
& 0 \le \zeta_i^t \quad \perp y_i(\mathbf{x}_i'\mathbf{w}^t - b_t) + z_i^t \ge 0 \\
& 0 \le z_i^t \quad \perp 1 - \zeta_i^t \ge 0
\end{aligned}\right\} \forall\, i \in \mathcal{N}_t, \\
& \left.\begin{aligned}
& 0 \le \alpha_j^t \quad \perp y_j(\mathbf{x}_j'\mathbf{w}^t - b_t) - 1 + \xi_j^t \ge 0 \\
& 0 \le \xi_j^t \quad \perp 1 - \alpha_j^t \ge 0
\end{aligned}\right\} \forall\, j \in \overline{\mathcal{N}}_t, \\
& 0 \le \boldsymbol{\gamma}^{t,+} \perp \overline{\mathbf{w}} - \mathbf{w}^t \ge 0, \\
& 0 \le \boldsymbol{\gamma}^{t,-} \perp \overline{\mathbf{w}} + \mathbf{w}^t \ge 0, \\
& \lambda\,\mathbf{w}^t - \sum_{j\in\overline{\mathcal{N}}_t} y_j\alpha_j^t\mathbf{x}_j + \boldsymbol{\gamma}^{t,+} - \boldsymbol{\gamma}^{t,-} = 0, \\
& \sum_{j\in\overline{\mathcal{N}}_t} y_j\alpha_j^t = 0,
\end{aligned}
\tag{2.11}
$$

which is an instance of an MPEC. It is a nonconvex optimization problem because of the complementarity constraints. We refer to this problem as the Bilevel Misclassification Minimization (BilevelMM) problem.

2.3 Bilevel Classification Variations

There are many possible variations of the bilevel classification problem. To illustrate the versatility of the bilevel approach, we discuss the outer-level objective and feature selection strategies.

2.3.1 Outer-level objective

The outer-level objective, (2.6), is not the only criterion that can be used to estimate generalization error within the cross validation scheme. An intuitively appealing alternative is to use the same misclassification measure for both the outer- and inner-level problems. Thus, we can also use the *hinge loss*, which minimizes the distance of each misclassified validation point from the classifier margin trained within each fold. The hinge

loss is an upper bound on the misclassification objective:

$$\Theta(\mathbf{W},\mathbf{b}) = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{|\mathcal{N}_t|}\sum_{i\in\mathcal{N}_t}\max\left(1-y_i(\mathbf{x}_i'\mathbf{w}^t-b_t),\,0\right). \tag{2.12}$$

The resulting MPEC is simpler because the lower level problems, (2.9), introduced to calculate the average number of points misclassified are not required. One might expect that this would lead to faster solutions by the FILTER solver on NEOS. But as we see later, this is not the case. When the hinge-loss is used in the outer-level, the MPEC becomes

$$\begin{aligned}
\min\quad & \frac{1}{T}\sum_{t=1}^{T}\frac{1}{|\mathcal{N}_t|}\sum_{i\in\mathcal{N}_t} z_i^t \\
\text{s. t.}\quad & \lambda_{\mathbf{lb}} \le \lambda \le \lambda_{\mathbf{ub}},\ \overline{\mathbf{w}}_{\mathbf{lb}} \le \overline{\mathbf{w}} \le \overline{\mathbf{w}}_{\mathbf{ub}}, \\
& \text{and for } t = 1\ldots T, \\
& \left.\begin{array}{l} z_i^t \ge 1 - y_i\left(\mathbf{x}_i'\mathbf{w}^t - b_t\right) \\ z_i^t \ge 0 \end{array}\right\} \forall\, i\in\mathcal{N}_t, \\
& \left.\begin{array}{l} 0 \le \alpha_j^t \quad \perp\, y_j(\mathbf{x}_j'\mathbf{w}^t - b_t) - 1 + \xi_j^t \ge 0 \\ 0 \le \xi_j^t \quad \perp\, 1 - \alpha_j^t \ge 0 \end{array}\right\} \forall\, j\in\overline{\mathcal{N}}_t, \\
& 0 \le \boldsymbol{\gamma}^{t,+} \perp \overline{\mathbf{w}} - \mathbf{w}^t \ge 0, \\
& 0 \le \boldsymbol{\gamma}^{t,-} \perp \overline{\mathbf{w}} + \mathbf{w}^t \ge 0, \\
& \lambda\,\mathbf{w}^t - \sum_{j\in\overline{\mathcal{N}}_t} y_j\alpha_j^t\mathbf{x}_j + \boldsymbol{\gamma}^{t,+} - \boldsymbol{\gamma}^{t,-} = 0, \\
& \sum_{j\in\overline{\mathcal{N}}_t} y_j\alpha_j^t = 0.
\end{aligned} \tag{2.13}$$

We refer to this problem as Bilevel Hinge Loss (BilevelHL) problem. AMPL models (constraints and objectives) for both approaches are given in Appendix A.1.

2.3.2 Enhanced feature selection

The introduction of $\overline{\mathbf{w}}$ into the SVM represents a novel and powerful way to perform feature selection and to force $\mathbf{w}$ to be sparse. A simple way to enhance this would be to use either an L_1-norm regularization or a combination of L_1 and L_2 norms (elastic nets [91]) in the inner level. These variations would only require straightforward modifications

to the model. However, we will focus on yet another variation, one that attempts to incorporate prior knowledge into feature selection.

Suppose, for n-dimensional data, it was known *a priori* that at most $n_{\max}$ features are sufficient. This can be incorporated into the model by introducing the constraint $\|\overline{\mathbf{w}}\|_0 \leq n_{\max}$ into the outer-level problem, where $\|\cdot\|_0$ is called the zero-norm or the cardinality of a vector, i.e., it counts the number of non-zero elements in its argument. This constraint forces the number of allowable features to be bounded above by some user-defined maximum and causes the features with the smallest weights to be dropped from the model. The constraint can be rewritten using the $()_\star$ function, since we have $\|\overline{\mathbf{w}}\|_0 = \mathbb{1}'\overline{\mathbf{w}}_\star$. If the conditions (2.10) are used to rewrite the constraint, the following inequality and complementarity constraints are added to the outer-level of (2.11):

$$\begin{aligned} &\sum_{m=1}^{n} \delta_m \leq n_{\max}, \\ &0 \leq \boldsymbol{\delta} \quad \perp -\overline{\mathbf{w}} + \mathbf{d} \geq 0, \\ &0 \leq \mathbf{d} \quad \perp \quad \mathbb{1} - \boldsymbol{\delta} \geq 0. \end{aligned} \tag{2.14}$$

In the constraints above, $\boldsymbol{\delta}$ counts the selected features of $\overline{\mathbf{w}}$, and $\mathbf{d}$ is the multiplier to the constraint $\mathbb{1} - \boldsymbol{\delta} \geq 0$.

2.4 Inexact and Discretized Cross Validation

The bilevel formulations described in the previous section perform model selection by searching a continuous parameter space. In contrast, classical cross validation approximately solves the bilevel problem by searching a discretized version of the same parameter space. In the bilevel approach also performs inexact cross validation, by solving a relaxed version of the bilevel MPEC. Pertinent details of both these methods are described below.

2.4.1 Inexact cross validation

There exist several approaches that can deal with the complementarity constraints in MPECs such as (2.11). Some of these are: penalty methods, which allow for the violation of the complementarity constraints, but penalize them through a penalty term in the outer-level objective; smoothing methods, that construct smooth approximations of the complementarity constraints; and relaxation methods, that relax the complementarity constraints while retaining the convex constraints. We use the relaxation approach to

solve (2.11).

This method of solving an MPEC simply involves replacing all instances of the "hard" complementarity constraints of the form

$$0 \leq \mathbf{c} \perp \mathbf{d} \geq 0 \quad \equiv \quad \mathbf{c} \geq 0,\ \mathbf{d} \geq 0,\ \mathbf{c}'\mathbf{d} = 0,$$

with relaxed, "soft" complementarity constraints of the form

$$0 \leq \mathbf{c} \perp_{\mathbf{tol}} \mathbf{d} \geq 0 \quad \equiv \quad \mathbf{c} \geq 0,\ \mathbf{d} \geq 0,\ \mathbf{c}'\mathbf{d} \leq \mathbf{tol},$$

where $\mathbf{tol} > 0$ is some prescribed tolerance of the complementarity conditions. This leads us to the bilevel SVC problem with *inexact cross validation*, which is the same as (2.11) except that all the $\perp$ conditions are replaced by $\perp_{\mathbf{tol}}$. Even though this is still a non-convex optimization problem, it represents a novel approach in the context of machine learning. The tolerance parameter, **tol**, which is set *a priori*, determines the accuracy of the relaxation and performs inexact cross validation. That means: an appropriately chosen **tol** can enlarge the search region of the model at the expense of a tolerable decrease in model accuracy. This is similar to the well-known machine-learning concept of "early stopping" in that the quality of the out-of-sample errors—measured in the outer-level objective of the bilevel program—is not affected significantly by small perturbations to a computed solution, in turn facilitating an early termination of cross validation. This approach also has the advantage of easing the difficulty of dealing with the disjunctive nature of the complementarity constraints. The exact same approach can also be applied to the hinge-loss MPEC, (2.13).

2.4.2 Grid search

Classical cross validation is performed by discretizing the parameter space into a grid and searching for the combination of parameters that minimizes the out-of-sample error, also referred to as validation error. This corresponds to the outer-level objective of the bilevel program (2.11). Typically, coarse logarithmic parameter grids of base 2 or 10 are used. Once a locally optimal grid point with the smallest validation error has been found, it may be refined or fine-tuned by a local search.

In the case of SV classification, the only hyper-parameter is the regularization constant, λ. However, the bilevel model (2.11) uses the box-constrained SVM for feature

selection; Grid Search has to determine $\overline{\mathbf{w}}$ as well. It is this search in the $\overline{\mathbf{w}}$-space that causes a serious combinatorial difficulty for the grid approach. To see this, consider the case of T-fold cross validation using grid search, where λ and $\overline{\mathbf{w}}$ are each allowed to take on d discrete values. Assuming the data is n-dimensional, grid search would have to solve roughly $O(Td^{n+1})$ problems. The resulting combinatorial explosion makes grid search intractable for all but the smallest n. In this paper, to counter this difficulty, we implement the following heuristic scheme:

- To determine λ: Perform a one-dimensional grid search using the classical SVC problem (without the box constraint). The range $[\lambda_{\mathbf{lb}}, \lambda_{\mathbf{ub}}]$ is discretized into a coarse, base-10, logarithmic grid. These grid points constitute the search space for this step, which we will call *unconstrained grid search*. At each grid point, T SVC problems are solved on the training sets and the error is measure on the validation sets. The grid point with the smallest average validation error, $\overline{\lambda}$, is "optimal".

- To determine $\overline{\mathbf{w}}$: Perform a n-dimensional grid search to determine the relevant features of $\overline{\mathbf{w}}$ using the box-constrained SVC problem (BoxSVC) and $\overline{\lambda}$ obtained from the previous step. Only two distinct choices for each feature are considered: 0, to test feature redundancy, and some large value that would not affect the choice of an appropriate feature weight. In this setting, 3-fold cross validation would involve solving about $O(3*2^n)$ BoxSVC problems. We call this step *constrained grid search*.

- The number of problems in constrained grid search is already impractical necessitating a further restriction of the relevant features to a maximum of $n = 10$. If a data set has more features, they are ranked using *recursive feature elimination* [43], and the 10 best features are chosen.

2.5 Numerical Tests

We compare unconstrained and constrained grid search approaches to the bilevel approaches (2.11) and (2.13) relaxed through inexact cross validation. The bilevel programs were implemented in AMPL and solved using FILTER [31, 32, 35], which is a general-purpose nonlinear programming solver available on the NEOS server (www-neos.mcs.anl.gov). Unconstrained and constrained grid were solved using MOSEK's quadratic program solver accessed through a MATLAB interface.

Data set	ℓ_{train}	ℓ_{test}	n	T
Pima Indians Diabetes Database	240	528	8	3
Wisconsin Breast Cancer Database	240	442	9	3
Cleveland Heart Disease Database	216	81	14	3
Johns Hopkins University Ionosphere Database	240	111	33	3
Star/Galaxy Bright Database	600	1862	14	3
Star/Galaxy Dim Database	900	3292	14	3

Table 2.1: Descriptions of data sets used for classification.

2.5.1 Experimental design

We used 6 real-world classification data sets, four of which are available via anonymous ftp from the UCI Repository for Machine Learning and two from the Star/Galaxy database at the University of Minnesota. The data sets were all standardized to zero norm and unit standard deviation. Twenty instances of each data set were randomly generated and each instance was split into a training set with ℓ_{train} points, which is used for cross validation and a hold-out test set, with ℓ_{test} points. The data descriptions are shown in Table 2.1. The hyper-parameters in the bilevel program were restricted as follows: $\lambda \in [10^{-4}, 10^{4}]$ and $\overline{\mathbf{w}} \in [0, 1.5]$. Grid search used the exact same bounds but was further restricted to $\lambda \in \{10^{-4}, 10^{-3}, \ldots, 10^{3}, 10^{4}\}$ and $\overline{\mathbf{w}} \in \{0, 1.5\}$. The complementarity tolerance was set to be $\mathbf{tol} = 10^{-6}$ in all runs except in the BilevelHL problem on the `dim` data set, where the value of $\mathbf{tol} = 10^{-4}$ was used. These settings were used to perform 3-fold cross validation on each instance.

Using the cross-validated hyper-parameters $\widehat{\lambda}$ and $\widehat{\overline{\mathbf{w}}}$ obtained from the bilevel and the grid search approaches, we implement a post-processing procedure to calculate the generalization error on the hold-out data for each instance. Specifically, a constrained SVC problem is solved on *all* the training data using $\frac{3}{2}\widehat{\lambda}$ and $\widehat{\overline{\mathbf{w}}}$ giving the final classifier $(\widehat{\mathbf{w}}, \widehat{b})$ which is used to compute the test (hold-out) error rate:

$$ERROR_{test} = \frac{1}{\ell_{test}} \sum_{(\mathbf{x},y)\,\text{test}} \frac{1}{2} |\,\mathbf{sign}(\widehat{\mathbf{w}}'\mathbf{x} - \widehat{b}) - y\,|.$$

Recall that the bilevel model uses 3-fold cross-validation and that each training fold consists of two-thirds of the total training data. Consequently, the final regularization parameter, $\widehat{\lambda}$, is rescaled by a factor of $\frac{3}{2}$ because the final model, which is constructed in the post-processing phase, uses all of the training data. For general T-fold cross validation, as mentioned before, this factor will be $\frac{T}{T-1}$, $T > 1$, assuming that each fold contains the

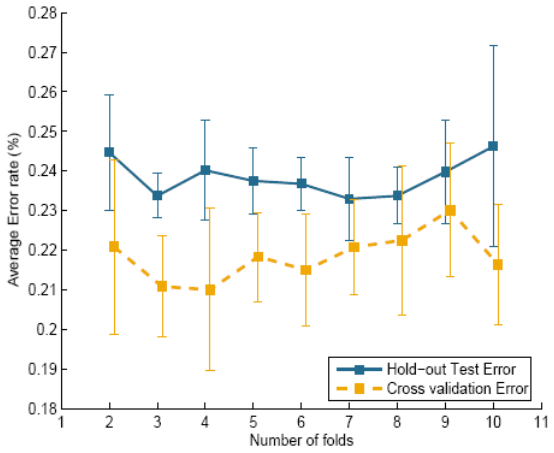


Figure 2.1: Effect of increasing the number of folds on learning rate of classifiers for the Pima Indians data set.

same fraction of data.

In addition, we also compute the cardinality of the final $\overline{\mathbf{w}}$ returned by the different approaches to determine the effectiveness of feature selection. For the bilevel approaches, the features in $\overline{\mathbf{w}}$ with weights less than $\sqrt{\mathbf{tol}}$ were considered irrelevant and set to zero, *after* which the test error was computed. Various criteria are used to compare the bilevel approach to the grid search approach: cross-validation error, test error, feature selection and execution time. The results, averaged over 20 instances for each data set, are presented in Table 2.2. Results which are significantly different (using a paired t-test at 10% confidence) with respect to unconstrained grid are shown in bold. The computational results in Table 2.2 all used $T = 3$ cross validation folds.

To study the effect of increasing the number of folds on cross validation error and test error, we report, in Figure 2.1, the results averaged over 5 instances of the `pima` data set. The results clearly demonstrate that larger number of folds can be successfully solved, but computation time does grow with the number of folds. The range of generalization values observed for different numbers of folds is not large, so $T = 3$ represents a reasonable choice. The best choice of the number of folds for a particular data set remains an open question.

Data set	Method	CV Error	Test Error	$\|\overline{\mathbf{w}}\|_0$	Time (sec.)
pima	UNC. GRID	23.10 ± 2.12	23.75 ± 0.94	8.0	12.3± 1.1
	CON. GRID	$\mathbf{21.04 \pm 1.63}$	24.13 ± 1.13	4.5	434.9± 35.1
	BILEVELMM	$\mathbf{21.87 \pm 2.25}$	23.90 ± 0.95	6.4	51.8± 23.0
	BILEVELHL	$\mathbf{44.04 \pm 3.19}$	23.80 ± 1.14	5.4	156.5± 57.5
cancer	UNC. GRID	3.54 ± 1.14	3.61 ± 0.64	9.0	11.7± 0.4
	CON. GRID	$\mathbf{2.73 \pm 0.88}$	$\mathbf{4.42 \pm 0.85}$	5.8	815.5± 29.7
	BILEVELMM	$\mathbf{3.13 \pm 1.05}$	3.59 ± 0.79	8.2	19.9± 8.3
	BILEVELHL	$\mathbf{6.13 \pm 2.22}$	3.76 ± 0.74	6.8	58.3± 33.6
heart	UNC. GRID	15.93 ± 2.02	16.05 ± 3.65	13.0	10.6± 0.8
	CON. GRID	$\mathbf{13.94 \pm 1.69}$	16.85 ± 4.15	7.1	1388.7± 37.6
	BILEVELMM	$\mathbf{14.49 \pm 1.47}$	16.73 ± 3.89	11.2	64.0± 20.5
	BILEVELHL	$\mathbf{28.89 \pm 3.20}$	16.30 ± 3.29	8.8	217.1± 82.5
ionosphere	UNC. GRID	22.27 ± 2.45	23.06 ± 2.45	33.0	7.5± 0.6
	CON. GRID	$\mathbf{19.25 \pm 2.07}$	22.34 ± 2.02	6.9	751.1± 3.0
	BILEVELMM	$\mathbf{19.16 \pm 2.44}$	23.65 ± 2.99	20.2	423.0±159.5
	BILEVELHL	$\mathbf{33.79 \pm 2.79}$	22.79 ± 2.03	14.2	1248.8±618.5
bright	UNC. GRID	0.78 ± 0.34	0.74 ± 0.13	14.0	22.7± 0.2
	CON. GRID	$\mathbf{0.51 \pm 0.24}$	$\mathbf{0.97 \pm 0.33}$	6.7	3163.7± 11.5
	BILEVELMM	$\mathbf{0.62 \pm 0.31}$	0.79 ± 0.14	11.2	110.9± 61.2
	BILEVELHL	$\mathbf{1.12 \pm 0.58}$	0.75 ± 0.14	8.9	564.2±335.7
dim	UNC. GRID	4.71 ± 0.55	4.96 ± 0.29	14.0	55.0± 5.1
	CON. GRID	$\mathbf{4.36 \pm 0.51}$	$\mathbf{5.21 \pm 0.37}$	7.2	7643.5± 74.5
	BILEVELMM	4.77 ± 0.64	$\mathbf{5.51 \pm 0.33}$	7.7	641.5±344.1
	BILEVELHL	$\mathbf{9.54 \pm 1.00}$	$\mathbf{5.28 \pm 0.36}$	5.7	1465.2±552.9

Table 2.2: Computational Results comparing Grid Search and Bilevel Approaches.

2.5.2 Discussion

We first examine the performance of the bilevel misclassification minimization (BilevelMM) programming approach with respect to the grid search methods. The first conclusion that can be drawn, from computational efficiency perspective, is that BilevelMM vastly outperforms the constrained grid search approach; the execution times for the former are several orders of magnitude smaller than the latter. It should be noted that the reported computation times for FILTER include transmission times as well as solve times, and that the reported computation times for grid search are enhanced by the use of smart restarting heuristics. However, despite the latter, it is clear that constrained grid search quickly becomes impractical as the problem size grows. This effect is clearly noticeable in the computation times for the Star/Galaxy data sets, where the execution time is affected, not only by the number of features, but also by the data set sizes, which contain hundreds

of training points. BilevelMM, on the other hand, is capable of cross-validating the `dim` data set, with 900 training points, in around 10-11 minutes on average. This suggests that the scalability of the bilevel approach could be improved significantly by exploiting the structure and sparsity inherent in SVMs. Research is currently underway in this direction and findings will be reported elsewhere.

With regard to generalization error, two interesting points emerge. First, the BilevelMM approach consistently produces results that are comparable to, if not slightly better than the grid search approaches, in spite of the fact that the cross-validation error is typically higher. This can be attributed to the fact that general-purpose NLP solvers tend to converge to acceptable solutions, with no guarantee of global optimality. Second, the only exception is the `dim` data set where the slight degradation in generalization performance can be imputed to numerical difficulties experienced by the NLP solvers because of large dimensionality and large data set size. Again, a specialized algorithm that could guarantee global optimality could produce better generalization performance.

With regard to feature selection, it is clear that unconstrained grid performs none at all, while, interestingly, constrained grid search uses less features than BilevelMM, albeit at the expense of excessive computational times and poorer generalization. This can be attributed to the fact that constrained grid is greedy, i.e., it analyzes *every* combination of features to find an optimal set and performs feature selection aggressively on data sets with more than 10 features as it drops the remaining features using RFE. BilevelMM has no such heuristic or mechanism to drive the number of selected features down. Despite this, it is clear that it does succeed in performing a better trade-off between feature selection and generalization. See Section 2.3.2 for ideas that might improve feature selection in the bilevel setting.

Next, we discuss the performance of the bilevel hinge-loss (BilevelHL) approach and compare it to BilevelMM. The most striking difference is in the computation times of the two approaches, with BilevelHL, quite unexpectedly, taking two to three times longer. We theorize that this is because BilevelMM has many more stationary points (for an intuitive explanation of this curious property that is endemic to misclassification minimization problems, see [61]) than BilevelHL and consequently, a general-purpose NLP solver tends to converge to stationarity faster. However, BilevelHL is still considerably faster than grid search; again, the only exception being the `ionosphere` data set. It should be noted that constrained grid search used only 10 features—after recursive feature elimination was

used to drop 23 of the 33 features—while BilevelHL solved the full problem using all the features. If constrained grid were to use all 33 features, it would have to solve around $O(10^{11})$ BoxSVC problems.

In terms of generalization error, BilevelHL performs as well or better than BilevelMM and never significantly worse than unconstrained grid (except for the `dim` data set), despite the fact that the CV errors of BilevelHL are uniformly higher. Recall that a complementarity tolerance of 10^{-4} was used for the `dim` data set. This was to relax the problem further for the numerical stability of the NLP solver. This relaxation, however, leads to a slight degradation in the quality of the solution.

Finally, BilevelHL tends to pick fewer features than BilevelMM, but still more than constrained grid. This comparison between BilevelMM and BilevelHL indicates that the best choice of outer-level objective is still an open question in need of further research.

2.6 Chapter Conclusions

It was shown that T-fold cross-validation can be cast as a continuous bilevel program: inner-level problems are introduced for each of the T-folds to compute classifiers on the training sets and to calculate the misclassification errors on the training sets within each fold. Furthermore, this chapter introduced the box-constrained SVM which has a hyper-parameter for each feature to perform feature selection. The resulting bilevel program is converted to an MPEC, which is in turn converted to a nonlinear programming problem through inexact cross validation. The advantage of the bilevel approach is that many hyper-parameters can be optimized simultaneously, unlike prior grid search approaches that are practically limited to one or two parameters. Initial computational results using FILTER through NEOS were very promising. High quality solutions were found using few features using much less computation time than grid search approaches over the same hyper-parameters.

This work represents a first proof of concept. It was showed that cross-validation through minimization of different objectives such as averaged misclassification error and hinge loss could be solved efficiently with large numbers of hyper-parameters. The resulting classifiers demonstrated good generalization ability and were dependent on only a few features. The success of these two different bilevel approaches suggests that other changes in the objective and regularization can lead to further enhancement of performance for classification problems. Furthermore, the versatility of the bilevel approach suggests that

further variations could be developed to tackle other challenges in machine learning such as missing data, semi-supervised learning, kernel learning and multi-task learning.

A major outstanding research question is the development of efficient optimization algorithms for the bilevel program. A step in this direction is taken in the next chapter which discusses model selection for SV regression through bilevel cross validation and how the resulting LPEC can be solved quickly and efficiently using the *Successive Linearization Algorithm* (SLA).

CHAPTER 3
SV Regression and Successive Linearization

3.1 Introduction

In this chapter, we apply the bilevel cross validation approach to model selection for support vector regression. Two alternative methods for solving the resulting LPEC are examined. In both methods, the convex lower level problems are replaced by their Karush-Kuhn-Tucker (KKT) optimality conditions, so that the problem becomes a mathematical programming problem with equilibrium constraints (MPEC). The equivalent optimization problem has a linear objective and linear constraints except for the set of equilibrium constraints formed by the complementarity conditions. In the first approach, the equilibrium constraints are relaxed from equalities to inequalities to form a nonlinear program (NLP) that is then solved by a state-of-the-art general-purpose nonlinear programming solver, FILTER [31]. In the second approach, the equilibrium constraints are treated as penalty terms and moved to the objective. The resulting penalty problem is then solved using the successive linearization algorithm for model selection (SLAMS). Further performance enhancements are obtained by stopping SLAMS at the first MPEC feasible solution found, a version we term EZ-SLAMS.

The proposed bilevel programming approaches offer several fundamental advantages over prior approaches. First, recent advances in bilevel programming in the optimization community permit the systematic treatment of models based on popular loss functions used for SVM and kernel methods with many hyper-parameters. In Section 3.3, we examine two algorithms for locally optimizing the models. Computational results in Sections 3.5 and 3.6 illustrate that the bilevel methods, particularly EZ-SLAMS, outperform traditional grid search approaches. In addition to the ability to simultaneously optimize many hyper-parameters, the bilevel programming approach offers a broad framework in which novel regularization methods can be developed, valid bounds on the test set errors can be obtained, and most significantly, improved model selection can be performed.

3.2 A Bilevel Support-Vector Regression Model

The regression data are described by the ℓ points $\Omega := \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell)\}$ in the Euclidean space $\mathbb{R}^{n+1}$ for some positive integers ℓ and n. Consider the regression problem of finding a function $f^* : \mathbb{R}^n \to \mathbb{R}$ among a given class that minimizes the regularized risk

functional

$$R[f] \equiv P[f] + \frac{C}{\ell} \sum_{i=1}^{\ell} L(y_i, f(\mathbf{x}_i)),$$

where L is a loss function of the observed data and model outputs, P is a regularization operator, and C is the regularization parameter. Usually the ε-insensitive loss $L_\varepsilon(y, f(\mathbf{x})) = \max\{|y - f(\mathbf{x})| - \varepsilon, 0\}$ is used in SVR, where $\varepsilon > 0$ is the *tube parameter*, which could be difficult to select as one does not know beforehand how accurately the function will fit the data. For linear functions: $f(\mathbf{x}) = \mathbf{w}'\mathbf{x} = \sum_{i=1}^{n} w_i x_i$, where the bias term is ignored but can easily be accommodated, the regularization operator in classic SVR is the squared ℓ_2-norm of the normal vector $\mathbf{w} \in \mathbb{R}^n$; i.e., $P[f] \equiv \|\mathbf{w}\|_2^2 = \sum_{i=1}^{n} w_i^2$.

The classic SVR approach has two hyper-parameters, the regularization constant C and the tube width ε, that are typically selected by cross validation based on the mean square error (MSE) or mean absolute deviation (MAD) measured on the out-of-sample data. In what follows, we focus on the latter and introduce additional parameters for feature selection and improved regularization and control. We partition the ℓ data points into T disjoint partitions, Ω_t for $t = 1, \ldots, T$, such that $\bigcup_{t=1}^{T} \Omega_t = \Omega$. Let $\overline{\Omega}_t \equiv \Omega \setminus \Omega_t$ be the subset of the data other than those in group Ω_t. The sets $\overline{\Omega}_t$ are called *training sets* while the sets Ω_t are called the *validation sets*. We denote $\overline{\mathcal{N}}_t$ and $\mathcal{N}_t$ to be their index sets respectively. For simplicity, we will ignore the bias term, b, but the method can easily be generalized to accommodate it. By specifying, for the general formulation (1.41), L_2 norm regularization and the ε-insensitive loss function we get the program below; the model selection bilevel program is to find the parameters ε, C and $\mathbf{w}^t$ for $t = 1, \cdots, T$, and also the bound $\overline{\mathbf{w}}$ in order to

$$\begin{aligned} \underset{C,\varepsilon,\mathbf{w}^t,\overline{\mathbf{w}}}{\text{minimize}} \quad & \frac{1}{T} \sum_{t=1}^{T} \frac{1}{|\overline{\mathcal{N}}_t|} \sum_{i \in \overline{\mathcal{N}}_t} |\mathbf{x}_i' \mathbf{w}^t - y_i| \\ \text{subject to} \quad & \varepsilon, C, \geq 0, \overline{\mathbf{w}} \geq 0, \\ \text{and for} \quad & t = 1, \ldots, T, \end{aligned} \tag{3.1}$$

$$\mathbf{w}^t \in \underset{-\overline{\mathbf{w}} \leq \mathbf{w} \leq \overline{\mathbf{w}}}{\arg\min} \left\{ C \sum_{j \in \overline{\mathcal{N}}_t} \max(|\mathbf{x}_j' \mathbf{w} - y_j| - \varepsilon, 0) + \frac{1}{2} \| \mathbf{w} \|_2^2 \right\}, \tag{3.2}$$

where the argmin in the last constraint denotes the set of optimal solutions to the convex optimization problem (3.2) in the variable $\mathbf{w}$ for given hyper-parameters ε, C, $\mathbf{w}_0$ and $\overline{\mathbf{w}}$.

The parameter, $\overline{\mathbf{w}}$, is related to feature selection and regularization. The bound constraints $-\overline{\mathbf{w}} \leq \mathbf{w} \leq \overline{\mathbf{w}}$ are introduced as in the classification case. In addition to

constraining the capacity of each of the functions, they also force all the subsets to use the same descriptors, a form of variable selection. This effect can be enhanced by adopting the one-norm, which forces $\mathbf{w}$ to be sparse. The box constraints will ensure that a consistent but not necessarily identical sets will be used across the folds.

Note that the loss functions used in the first level and second level—to measure errors—need not match. For the inner-level optimization, we adopt the ε-insensitive loss function because it produces robust solutions that are sparse in the dual space. But typically, ε-insensitive loss functions are not employed in the outer cross-validation objective; so here we use mean absolute deviation (as an example). Also, to facilitate comparison with grid search, we restrict C and ε to be within prescribed upper bounds.

3.2.1 Bilevel Problems as MPECs

Just as in the last section, we can convert the bilevel program (3.1) into a LPEC. Collecting all the weight vectors across the folds, $\mathbf{w}^t$, column-wise into the matrix W for compactness, the cross-validation error measured as mean average deviation across all the folds is

$$\Theta(W) = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{|\mathcal{N}_t|}\sum_{i\in\mathcal{N}_t}|\,\mathbf{x}_i'\mathbf{w}^t - y_i\,|, \tag{3.3}$$

and is subject to the simple restrictions on these parameters, and most importantly, to the additional inner-level optimality requirement of each $\mathbf{w}^t$ for $t = 1,\ldots,T$. To solve (3.1), we rewrite the inner-level optimization problem (3.2) by introducing additional slack variables, $\boldsymbol{\xi}^t \geq 0$ within the t-th fold as follows: for given ε, C and $\overline{\mathbf{w}}$,

$$\begin{aligned}
\underset{\mathbf{w}^t,\,\boldsymbol{\xi}^t}{\text{minimize}} \quad & C\sum_{j\in\overline{\mathcal{N}}_t}\xi_j^t + \frac{1}{2}\|\,\mathbf{w}^t\,\|_2^2 \\
\text{subject to} \quad & -\overline{\mathbf{w}} \leq \mathbf{w}^t \leq \overline{\mathbf{w}}, \\
& \left.\begin{aligned}
\xi_j^t &\geq \mathbf{x}_j'\mathbf{w}^t - y_j - \varepsilon \\
\xi_j^t &\geq y_j - \mathbf{x}_j'\mathbf{w}^t - \varepsilon \\
\xi_j^t &\geq 0
\end{aligned}\right\} \quad j \in \overline{\mathcal{N}}_t,
\end{aligned} \tag{3.4}$$

which is easily seen to be a convex quadratic program in the variables $\mathbf{w}^t$ and $\boldsymbol{\xi}^t$. By letting $\boldsymbol{\gamma}^{t,\pm}$ be the multipliers of the bound constraints, $-\overline{\mathbf{w}} \leq \mathbf{w} \leq \overline{\mathbf{w}}$, respectively, and $\alpha_j^{t,\pm}$ be the multipliers of the constraints $\xi_j^t \geq \mathbf{x}_j'\mathbf{w}^t - y_j - \varepsilon$ and $\xi_j^t \geq y_j - \mathbf{x}_j'\mathbf{w}^t - \varepsilon$, respectively, we obtain the Karush-Tucker-Tucker optimality conditions of (3.4) as the

following linear complementarity problem in the variables $\mathbf{w}^t$, $\boldsymbol{\gamma}^{t,\pm}$, $\boldsymbol{\alpha}_j^{t,\pm}$, and $\boldsymbol{\xi}_j^t$:

$$\left.\begin{array}{l} 0 \le \boldsymbol{\gamma}^{t,-} \perp \overline{\mathbf{w}} + \mathbf{w}^t \ge 0, \\ 0 \le \boldsymbol{\gamma}^{t,+} \perp \overline{\mathbf{w}} - \mathbf{w}^t \ge 0, \\ \left.\begin{array}{l} 0 \le \alpha_j^{t,-} \perp \mathbf{x}_j'\mathbf{w}^t - y_j + \varepsilon + \xi_j^t \ge 0 \\ 0 \le \alpha_j^{t,+} \perp y_j - \mathbf{x}_i'\mathbf{w}^t + \varepsilon + \xi_j^t \ge 0 \\ 0 \le \xi_j^t \quad \perp C - \alpha_j^{t,+} - \alpha_j^{t,-} \ge 0 \end{array}\right\} \forall j \in \overline{\mathcal{N}}_t, \\ 0 = \mathbf{w}^t + \sum_{j \in \overline{\mathcal{N}}_t} (\alpha_j^{t,+} - \alpha_j^{t,-})\mathbf{x}_j + \boldsymbol{\gamma}^{t,+} - \boldsymbol{\gamma}^{t,-}, \end{array}\right. \tag{3.5}$$

where $a \perp b$ means $a'b = 0$. The orthogonality conditions in (3.5) express the well-known complementary slackness conditions of the inner-level (parametric) quadratic program. All the conditions (3.5) represent the Karush-Kuhn-Tucker conditions. The overall two-level regression problem is therefore

$$\begin{array}{ll} \text{minimize} & \dfrac{1}{T}\displaystyle\sum_{t=1}^{T} \dfrac{1}{|\mathcal{N}_t|} \sum_{i \in \mathcal{N}_t} z_i^t \\ \text{subject to} & \varepsilon, C \ge 0, \quad \overline{\mathbf{w}} \ge 0, \\ \text{and} & \text{for all } t = 1, \ldots, T \\ & \quad -z_i^t \le \mathbf{x}_i'\mathbf{w}^t - y_i \le z_i^t, \qquad \forall\, i \in \mathcal{N}_t, \\ & \quad \left.\begin{array}{l} 0 \le \alpha_j^{t,-} \perp \mathbf{x}_j'\mathbf{w}^t - y_j + \varepsilon + \xi_j^t \ge 0 \\ 0 \le \alpha_j^{t,+} \perp y_j - \mathbf{x}_i'\mathbf{w}^t + \varepsilon + \xi_j^t \ge 0 \\ 0 \le \xi_j^t \quad \perp C - \alpha_j^{t,+} - \alpha_j^{t,-} \ge 0 \end{array}\right\} \forall j \in \overline{\mathcal{N}}_t, \\ & \quad 0 \le \boldsymbol{\gamma}^{t,-} \perp \overline{\mathbf{w}} + \mathbf{w}^t \ge 0, \\ & \quad 0 \le \boldsymbol{\gamma}^{t,+} \perp \overline{\mathbf{w}} - \mathbf{w}^t \ge 0, \\ & \quad 0 = \mathbf{w}^t + \displaystyle\sum_{j \in \overline{\mathcal{N}}_t} (\alpha_j^{t,+} - \alpha_j^{t,-})\mathbf{x}_j + \boldsymbol{\gamma}^{t,+} - \boldsymbol{\gamma}^{t,-}. \end{array} \tag{3.6}$$

The most noteworthy feature of the above optimization problem is the complementarity conditions in the constraints, making the problem an instance of a linear program with linear complementarity constraints (sometimes called an LPEC). The AMPL model (con-

straints and objectives) for this approach is given in Appendix A.2. The discussion in the remainder of this paper focuses on this case.

3.3 Bilevel Optimization Methods

In this section, we describe two alternative methods for solving the model. We also describe the details of the classical grid search approach. The difficulty in solving the LPEC reformulation (3.6) of the bilevel optimization problem (3.1) stems from the linear complementarity constraints formed from the optimality conditions of the inner problem (3.5); all of the other constraints and the objective are linear. It is well recognized that a straightforward solution using the LPEC formulation is not appropriate because of the complementarity constraints, which give rise to both theoretical and computational anomalies that require special attention.

Among various proposals to deal with these constraints, two are particularly effective for finding a local solution: one is to relax the complementarity constraints and retain the relaxations in the constraints. The other proposal is via a penalty approach that allows the violation of these constraints but penalizes the violation by adding a penalty term in the objective function of (3.6). There are extensive studies of both treatments, including detailed convergence analyses and numerical experiments on realistic applications and random problems [8, 60, 61, 67, 72]. In this work, we experiment with both approaches.

3.3.1 A Relaxed NLP Reformulation

Exploiting the LPEC structure, the first solution method that is implemented in our experiments for solving (3.6) employs a relaxation of the complementarity constraint. In the relaxed complementarity formulation, we let $\mathbf{tol} > 0$ be a prescribed tolerance of the complementarity conditions. Just as for the classification case we replace $\perp$ with $\perp_{\mathbf{tol}}$ where $a \perp_{\mathbf{tol}} b$ means $a'b \leq \mathbf{tol}$. The latter formulation constitutes the *relaxed bilevel support-vector regression problem* that we employ to determine the hyper-parameters C, ε and $\overline{\mathbf{w}}$; the computed parameters are then used to define the desired support-vector model for data analysis. The relaxed complementary slackness is a novel feature that aims at enlarging the search region of the desired regression model; the relaxation corresponds to *inexact cross validation* whose accuracy is dictated by the prescribed scalar, **tol**.

The above NLP remains a non-convex optimization problem; thus, finding a global optimal solution is hard, but the state-of-the-art general-purpose NLP solver, FILTER [31]

that is available on the NEOS server is used. We also experimented with SNOPT [39] but as reported in [7], we found FILTER to work better overall. FILTER is a sequential quadratic programming (SQP) based method, which is a Newton-type method for solving problems with nonlinear objectives and nonlinear constraints. The method solves a sequence of approximate convex quadratic programming subproblems. FILTER implements a SQP algorithm using a trust-region approach with a "filter" to enforce global convergence. It terminates either when a Karush-Kuhn-Tucker point is found within a specified tolerance or no further step can be processed (possibly due to the infeasibility of a subproblem).

3.3.2 Penalty Reformulation

Another approach to solving the problem (3.6) is the penalty reformulation. Penalty and augmented Lagrangian methods have been widely applied to solving LPECs and MPECs, for instance, by [47]. These methods typically require solving an unconstrained optimization problem. In contrast, penalty methods penalize only the complementarity constraints in the objective by means of a penalty function.

Consider the LPEC, (3.6), resulting from the reformulation of the bilevel regression problem. Define S_t, for $t = 1, \ldots, T$, to be the constraint set within the t-th fold, without the complementarity constraints:

$$S_t := \left\{ z^t, \boldsymbol{\alpha}^{t,\pm}, \boldsymbol{\xi}^t, \boldsymbol{\gamma}^{t,\pm}, \mathbf{r}^t, s^t \;\middle|\; \begin{array}{ll} -z_i^t \le \mathbf{x}_i'\mathbf{w}^t - y_i \le z_i^t, & \forall\, i \in \mathcal{N}_t, \\ \mathbf{x}_j'\mathbf{w}^t - y_j + \varepsilon + \xi_j^t \ge 0 & \\ y_j - \mathbf{x}_i'\mathbf{w}^t + \varepsilon + \xi_j^t \ge 0 & \forall\, j \in \overline{\mathcal{N}}_t, \\ C - \alpha_j^{t,+} - \alpha_j^{t,-} \ge 0 & \\ -\overline{\mathbf{w}} \le \mathbf{w}^t \le \overline{\mathbf{w}}, & \\ 0 = \mathbf{w}^t + \sum_{j \in \overline{\mathcal{N}}_t} (\alpha_j^{t,+} - \alpha_j^{t,-})\mathbf{x}_j + \boldsymbol{\gamma}^{t,+} - \boldsymbol{\gamma}^{t,-}, & \\ \mathbf{w}^t = \mathbf{r}^t - \mathbb{1}\, s^t, & \\ z^t, \boldsymbol{\alpha}^{t,\pm}, \boldsymbol{\xi}^t, \boldsymbol{\gamma}^{t,\pm}, \mathbf{r}^t, s^t \ge 0. & \end{array} \right\}, \tag{3.7}$$

where we rewrite the weight vector, $\mathbf{w}^t$, within each fold as $\mathbf{w}^t = \mathbf{r}^t - \mathbb{1}\, s^t$, with $\mathbf{r}^t$, $s^t \ge 0$

and $\mathbb{1}$ denotes a vector of ones of appropriate dimension. Also, let S_0 be defined as

$$S_0 := \left\{ C, \varepsilon, \overline{\mathbf{w}} \;\middle|\; \begin{array}{c} C \in [\underline{C}, \overline{C}], \\ \varepsilon \in [\underline{\varepsilon}, \overline{\varepsilon}], \\ \mathbf{w}_{\mathsf{lb}} \leq \overline{\mathbf{w}} \leq \mathbf{w}_{\mathsf{ub}} \end{array} \right\}. \tag{3.8}$$

Then, the overall constraint set for the LPEC (3.6), without the complementarity constraints is defined as $S_{\mathsf{LP}} := \bigcap_{t=0}^{T} S_t$. Let all the variables in (3.7) and (3.8) be collected into the vector $\boldsymbol{\zeta} \geq 0$.

In the penalty reformulation, all the complementarity constraints of the form $a \perp b$ in (3.6) are moved into the objective via the penalty function, $\phi(a, b)$. This effectively converts the LPEC (3.6) into a penalty problem of minimizing some, possibly non-smooth, objective function on a *polyhedral set*. Typical penalty functions include the differentiable quadratic penalty term, $\phi(a, b) = a'b$, and the non-smooth piecewise-linear penalty term, $\phi(a, b) = \min(a, b)$. Here, we consider the quadratic penalty. The penalty term, which is a product of the complementarity terms is

$$\phi(\boldsymbol{\zeta}) = \sum_{t=1}^{T} \Bigg(\overbrace{\frac{1}{2}\|\mathbf{w}^t\|_2^2 + C \sum_{j \in \mathcal{N}_t} \xi_j^t + \frac{1}{2} \sum_{i \in \mathcal{N}_t} \sum_{j \in \mathcal{N}_t} (\alpha_i^{t,+} - \alpha_i^{t,-})(\alpha_j^{t,+} - \alpha_j^{t,-}) \mathbf{x}_i' \mathbf{x}_j}^{\Theta_p^t} + \varepsilon \sum_{j \in \overline{\mathcal{N}}_t} (\alpha_j^{t,+} + \alpha_j^{t,-}) + \sum_{j \in \overline{\mathcal{N}}_t} y_j (\alpha_j^{t,+} - \alpha_j^{t,-}) - \overline{\mathbf{w}}' \boldsymbol{\gamma}^{t,+} + \underline{\mathbf{w}}' \boldsymbol{\gamma}^{t,-} \Bigg)}^{-\Theta_d^t}. \tag{3.9}$$

The first two terms in the quadratic penalty constitute the primal objective, Θ_p^t, while the last five terms constitute the negative of the dual objective, Θ_d^t, for support vector regression in the t-th fold. Consequently, the penalty function is a combination of T differences between the primal and dual objectives of the regression problem in each fold. Thus,

$$\phi(\boldsymbol{\zeta}) = \sum_{t=1}^{T} \left(\Theta_p^t(\boldsymbol{\zeta}_p^t) - \Theta_d^t(\boldsymbol{\zeta}_d^t) \right),$$

where $\boldsymbol{\zeta}_p^t \equiv (\mathbf{w}^t, \boldsymbol{\xi}^t)$, the vector of primal variables in the t-th primal problem and $\boldsymbol{\zeta}_d^t \equiv (\boldsymbol{\alpha}^{t,\pm}, \boldsymbol{\gamma}^{t,\pm}, \boldsymbol{\xi}^t)$, the vector of dual variables in the t-th dual problem. However, the penalty

function also contains the hyper-parameters, C, ε and $\overline{\mathbf{w}}$ as variables, rendering $\phi(\boldsymbol{\zeta})$ non-convex. Recalling that the linear cross-validation objective was denoted by Θ, we define the penalized objective: $P(\boldsymbol{\zeta};\, \mu) = \Theta(\boldsymbol{\zeta}) + \mu\, \phi(\boldsymbol{\zeta})$, and the penalized problem, $PF(\mu)$, is

$$\begin{array}{ll} \min\limits_{\boldsymbol{\zeta}} & P(\boldsymbol{\zeta};\, \mu) \\ \text{subject to} & \boldsymbol{\zeta} \in S_{\mathsf{LP}}. \end{array} \tag{3.10}$$

This penalized problem has very nice properties that have been extensively studied. First, we know that finite values of μ can be used, since local solutions of LPEC, as defined by strong stationarity, correspond to stationarity points of $PF(\mu)$. The point $\boldsymbol{\zeta}^*$ is a stationary point of $PF(\mu)$ if and only if there exists a Lagrangian multiplier vector $\boldsymbol{\rho}^*$, such that $(\boldsymbol{\zeta}^*,\, \boldsymbol{\rho}^*)$ is a KKT point of $PF(\mu)$. In general, KKT points do not exist for LPECs. An alternative local optimality condition, strong stationarity of the LPEC, means that $\boldsymbol{\zeta}^*$ solves an LP formed by fixing the LPEC complementarity conditions appropriately. See [72, Definition 2.2] for precise details on strong stationarity. Finiteness ensures that the penalty parameter can be set to reasonable values, contrasting with other approaches in which the penalty problem only solve the original problem in the limit.

Theorem 3.3.1 (Finite penalty parameter, [72], Theorem 5.2)**.** *Suppose that $\boldsymbol{\zeta}^*$ is a strongly stationary point of (3.6), then for all μ sufficiently large, there exists a Lagrangian multiplier vector $\boldsymbol{\rho}^*$, such that $(\boldsymbol{\zeta}^*,\, \boldsymbol{\rho}^*)$ is a KKT point of $PF(\mu)$ (3.10).*

It is perhaps not surprising to note that the zero penalty corresponds to a point where the primal and dual objectives are equal in (3.3.2). These strongly stationary solutions correspond to solutions of (3.10) with $\phi(\boldsymbol{\zeta}) = 0$, i.e., a zero penalty. The quadratic program, $PF(\mu)$, is non-convex, since the penalty term is not positive definite. Continuous optimization algorithms will not necessarily find a global solution of $PF(\mu)$. But we do know know that local solutions of $PF(\mu)$ that are feasible for the LPEC are also local optimal for the LPEC.

Theorem 3.3.2 (Complementary $PF(\mu)$ solution solves LPEC, [72], Theorem 5.2)**.** *Suppose $\boldsymbol{\zeta}^*$ is a stationary point of $PF(\mu)$ (3.10) and $\phi(\boldsymbol{\zeta}^*) = 0$. Then $\boldsymbol{\zeta}^*$ is a strongly stationary for (3.6).*

One approach to solving exact penalty formulations like (3.10) is the successive lineariza-

Algorithm 3.1 Successive linearization algorithm for model selection

Fix $\mu > 0$.

1. *Initialization*:
   Start with an initial point, $\boldsymbol{\zeta}^0 \in S_{\mathsf{LP}}$.

2. *Solve Linearized Problem*:
   Generate an intermediate iterate, $\bar{\boldsymbol{\zeta}}^k$, from the previous iterate, $\boldsymbol{\zeta}^k$, by solving the linearized penalty problem, $\bar{\boldsymbol{\zeta}}^k \in \arg \underset{\boldsymbol{\zeta} \in S_{\mathsf{LP}}}{\text{vertex min}} \nabla_{\boldsymbol{\zeta}} P(\boldsymbol{\zeta}^k; \mu)' (\boldsymbol{\zeta} - \boldsymbol{\zeta}^k)$.

3. *Termination Condition*:
   Stop if the minimum principle holds, i.e., if $\nabla_{\boldsymbol{\zeta}} P(\boldsymbol{\zeta}^k; \mu)' (\bar{\boldsymbol{\zeta}}^k - \boldsymbol{\zeta}^k) \geq 0$.

4. *Compute Step Size*:
   Compute step length $\lambda \in \arg \min_{0 \leq \lambda \leq 1} P\Big((1-\lambda)\, \boldsymbol{\zeta}^k + \lambda\, \bar{\boldsymbol{\zeta}}^k; \mu \Big)$, and get the next iterate, $\boldsymbol{\zeta}^{k+1} = (1-\lambda)\, \boldsymbol{\zeta}^k + \lambda\, \bar{\boldsymbol{\zeta}}^k$.

tion algorithm, where a sequence of problems with a linearized objective,

$$\Theta(\boldsymbol{\zeta} - \boldsymbol{\zeta}^k) + \mu \nabla \phi(\boldsymbol{\zeta}^k)'(\boldsymbol{\zeta} - \boldsymbol{\zeta}^k), \tag{3.11}$$

is solved to generate the next iterate. We now describe the *Successive Linearization Algorithm for Model Selection* (SLAMS).

3.3.3 Successive Linearization Algorithm for Model Selection

The QP, (3.10), can be solved using the Frank-Wolfe method [61] which simply involves solving a sequence of LPs until either a global minimum or some locally stationary solution of (3.6) is reached. In practice, a sufficiently large value of μ will lead to the penalty term vanishing from the penalized objective, $P(\boldsymbol{\zeta}^*; \mu)$. In such cases, the locally optimal solution to (3.10) will also be feasible and locally optimal to the LPEC (3.6).

Algorithm 1 gives the details of SLAMS. In Step 2, the notation *arg vertex min* indicates that $\boldsymbol{\zeta}^k$ is a vertex solution of the LP in Step 2. The step size in Step 4 has a simple closed form solution since a quadratic objective subject to bounds constraints is minimized . The objective has the form $f(\lambda) = a\lambda^2 + b\lambda$, so the optimal solution is either 0, 1 or $\frac{-b}{2a}$, depending on which value yields the smallest objective. SLAMS converges to a solution of the penalty problem. SLAMS is a special case of the Frank-Wolfe algorithm and a convergence proof of the Frank-Wolfe algorithm with no assumptions on the convexity of $P(\boldsymbol{\zeta}^j, \mu)$ can be found in [8], thus we offer the convergence result without proof.

Theorem 3.3.3 (Convergence of SLAMS [8]). *Algorithm 1 terminates at $\boldsymbol{\zeta}^k$ that satisfies the minimum principle necessary optimality condition of $PF(\mu)$: $\nabla_{\boldsymbol{\zeta}} P(\boldsymbol{\zeta}^k;\mu)'(\boldsymbol{\zeta}-\boldsymbol{\zeta}^k) \geq 0$ for all $\boldsymbol{\zeta} \in S_{LP}$, or each accumulation $\bar{\boldsymbol{\zeta}}$ of the sequence $\{\boldsymbol{\zeta}^k\}$ satisfies the minimum principle.*

Furthermore, for the case where SLAMS generates a complementary solution, SLAMS finds a strongly stationary solution of the LPEC.

Theorem 3.3.4 (SLAMS solves LPEC). *If the sequence $\boldsymbol{\zeta}^k$ generated by* SLAMS *accumulates to $\bar{\boldsymbol{\zeta}}$ such that $\phi(\bar{\boldsymbol{\zeta}}) = 0$, then $\boldsymbol{\zeta}$ is strongly stationary for LPEC (3.6).*

Proof. For notational convenience let the set $S_{\mathsf{LP}} = \{\boldsymbol{\zeta} \,|\, \mathbf{A}\boldsymbol{\zeta} \geq \mathbf{b}\}$, with an appropriate matrix, $\mathbf{A}$, and vector, $\mathbf{b}$. We first show that $\bar{\boldsymbol{\zeta}}$ is a KKT point of the problem

$$\begin{aligned} \min_{\boldsymbol{\zeta}} \quad & \nabla_{\boldsymbol{\zeta}} P(\boldsymbol{\zeta};\,\mu) \\ \text{s.t.} \quad & \mathbf{A}\boldsymbol{\zeta} \geq \mathbf{b}. \end{aligned}$$

We know that $\bar{\boldsymbol{\zeta}}$ satisfies $\mathbf{A}\bar{\boldsymbol{\zeta}} \geq \mathbf{b}$ since $\boldsymbol{\zeta}^k$ is feasible at the k-th iteration. By Theorem 3.3.3 above, $\bar{\boldsymbol{\zeta}}$ satisfies the minimum principle; thus, we know the systems of equations

$$\nabla_{\boldsymbol{\zeta}} P(\bar{\boldsymbol{\zeta}};\,\mu)'(\boldsymbol{\zeta} - \bar{\boldsymbol{\zeta}}^k) < 0, \quad \boldsymbol{\zeta} \in S_{\mathsf{LP}},$$

has no solution for any $\boldsymbol{\zeta} \in S_{\mathsf{LP}}$. Equivalently, if $I = \{i | A_i\bar{\boldsymbol{\zeta}} = \mathbf{b}_i\}$, then

$$P(\bar{\boldsymbol{\zeta}};\,\mu)'(\boldsymbol{\zeta} - \bar{\boldsymbol{\zeta}}) < 0, \quad A_i\boldsymbol{\zeta} \geq 0,\ i \in I,$$

has no solution. By Farkas' Lemma, there exists $\bar{\mathbf{u}}$ such that

$$\nabla_{\boldsymbol{\zeta}} P(\bar{\boldsymbol{\zeta}};\mu) - \sum_{i\in I} \bar{\mathbf{u}}_i A_i = 0, \quad \bar{\mathbf{u}} \geq 0.$$

Thus $(\bar{\boldsymbol{\zeta}}, \bar{\mathbf{u}})$ is a KKT point of $PF(\mu)$ and $\bar{\boldsymbol{\zeta}}$ is a stationary point of $PF(\mu)$. By Theorem 3.3.2, $\bar{\boldsymbol{\zeta}}$ is also a strongly stationary point of LPEC (3.6). □

3.3.4 Early Stopping

Typically, in many machine learning applications, emphasis is placed on generalization and scalability. Consequently, inexact solutions are preferred to globally optimal solutions as they can be obtained cheaply and tend to perform reasonably well. Noting

that, at each iteration, the algorithm is working to minimize the LPEC objective as well as the complementarity penalty, one alternative to speeding up termination at the expense of the objective is to stop as soon as *complementarity is reached.* Thus, as soon as an iterate produces a solution that is feasible to the LPEC, (3.6), the algorithm is terminated. We call this approach *Successive Linearization Algorithm for Model Selection with Early Stopping* (EZ-SLAMS). This is similar to the well-known machine learning concept of early stopping, except that the criterion used for termination is based on the status of the complementarity constraints i.e., feasibility to the LPEC. We adapt the finite termination result in [8] to prove that EZ-SLAMS terminates finitely for the case when complementary solutions exist, which is precisely the case of interest here. Note that the proof relies upon the fact that S_{LP} is polyhedral with no straight lines going to infinity in both directions.

Theorem 3.3.5 (Finite termination of EZ-SLAMS)**.** *If the sequence $\boldsymbol{\zeta}^k$ generated by* SLAMS *accumulates to $\bar{\boldsymbol{\zeta}}$ such that $\phi(\bar{\boldsymbol{\zeta}}) = 0$, then EZ-SLAM terminates at an LPEC (3.6) feasible solution $\boldsymbol{\zeta}^k$ in finitely many iterations.*

Proof. Let $\mathcal{V}$ be the finite subset of vertices of S_{LP} that constitutes the vertices $\{\bar{\mathbf{v}}^k\}$ generated by SLAMS. Then,

$$\{\boldsymbol{\zeta}^k\} \in \text{convex hull}\{\boldsymbol{\zeta}^0 \cup \mathcal{V}\},$$
$$\bar{\boldsymbol{\zeta}} \in \text{convex hull}\{\boldsymbol{\zeta}^0 \cup \mathcal{V}\}.$$

If $\bar{\boldsymbol{\zeta}} \in \mathcal{V}$, we are done. If not, then for some $\boldsymbol{\zeta} \in S_{\mathsf{LP}}$, $\mathbf{v} \in \mathcal{V}$ and $\lambda \in (0, 1)$,

$$\bar{\boldsymbol{\zeta}} = (1 - \lambda)\boldsymbol{\zeta} + \lambda \mathbf{v}.$$

For notational convenience define an appropriate matrix M and vector b such that $0 = \phi(\bar{\boldsymbol{\zeta}}) = \bar{\boldsymbol{\zeta}}'(M\bar{\boldsymbol{\zeta}} + q)$. We know $\bar{\boldsymbol{\zeta}} \geq 0$ and $M\bar{\boldsymbol{\zeta}} + q \geq 0$. Hence,

$$\mathbf{v}_i = 0, \text{ or } M_i\mathbf{v} + q_i = 0.$$

Thus, $\mathbf{v}$ is feasible for LPEC (3.6). □

The results comparing SLAMS to EZ-SLAMS are reported in Sections 3.5 and 3.6. It is interesting to note that there is always a significant decrease in running time with no significant degradation in training or generalization performance when early stopping is employed.

3.3.5 Grid Search

In classical cross-validation, parameter selection is performed by discretizing the parameter space into a grid and searching for the combination of parameters that minimizes the validation error (which corresponds to the upper level objective in the bilevel problem). This is typically followed by a local search for fine-tuning the parameters. Typical discretizations are logarithmic grids of base 2 or 10 on the parameters. In the case of the classic SVR, cross validation is simply a search on a two-dimensional grid of C and ε.

This approach, however, is not directly applicable to the current problem formulation because, in addition to C and ε, we also have to determine $\overline{\mathbf{w}}$, and this poses a significant combinatorial problem. In the case of k-fold cross validation of n-dimensional data, if each parameter takes d discrete values, cross validation would involve solving roughly $O(kd^{n+2})$ problems, a number that grows to intractability very quickly. To counter the combinatorial difficulty, we implement the following heuristic procedures:

- Perform a two-dimensional grid search on the unconstrained (classic) SVR problem to determine C and ε. We call this the *unconstrained grid search* (Unc. Grid). A coarse grid with values of 0.1, 1 and 10 for C, and 0.01, 0.1 and 1 for ε was chosen.

- Perform an n-dimensional grid search to determine the features of $\overline{\mathbf{w}}$ using C and ε obtained from the previous step. Only two distinct choices for each feature of $\overline{\mathbf{w}}$ are considered: 0, to test if the feature is redundant, and some large value that would not impede the choice of an appropriate feature weight, otherwise. Cross validation under these settings would involve solving roughly $O(3.2^N)$ problems; this number is already impractical and necessitates the heuristic. We label this step the *constrained grid search* (Con. Grid).

- For data sets with more than 10 features, *recursive feature elimination* [42] is used to rank the features and the 10 largest features are chosen.

3.4 Experimental Design

Our experiments aim to address several issues. The first series of experiments were designed to compare the successive linearization approaches (with and without early stopping) to the classical grid search method with regard to generalization and running time. The data sets used for these experiments consist of randomly generated synthetic data sets and real world chemoinformatics (QSAR) data. The second experiment was

designed to perform a scalability analysis comparing grid search with the SLA methods to evaluate their performance on large data sets. The third experiment was designed to evaluate the effectiveness of T-fold cross validation itself with different values of T, the number of folds. We now describe the data sets and the design of our experiments.

3.4.1 Synthetic Data

Data sets of different dimensionalities, training sizes and noise models were generated. The dimensionalities i.e., number of features considered were $n = 10$, 15 and 25, among which, only $n_r = 7$, 10 and 16 features respectively, were relevant. We trained on sets of $\ell = 30$, 60, 90, 120 and 150 points using 3-fold cross validation and tested on a hold-out set of a further $1,000$ points. Two different noise models were considered: Laplacian and Gaussian. For each combination of feature size, training set size and noise model, 5 trials were conducted and the test errors were averaged. In this subsection, we assume the following notation: $U(a, b)$ represents the uniform distribution on $[a, b]$, $N(\mu, \sigma)$ represents the normal distribution with probability density function $\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$, and $L(\mu, b)$ represents the Laplacian distribution with the probability density function $\frac{1}{2b} \exp\left(-\frac{|x-\mu|}{b}\right)$.

For each data set, the data, $\mathbf{w}_{\text{REAL}}$ and labels were generated as follows. For each point, 20% of the features were drawn from $U(-1, 1)$, 20% were drawn from $U(-2.5, 2.5)$, another 20% from $U(-5, 5)$, and the last 40% from $U(-3.75, 3.75)$. Each feature of the regression hyperplane $\mathbf{w}_{\text{REAL}}$ was drawn from $U(-1, 1)$ and the smallest $n - n_r$ features were set to 0 and considered irrelevant. Once the training data and $\mathbf{w}_{\text{REAL}}$ were generated, the noise-free regression labels were computed as $y_i = \mathbf{x}_i' \mathbf{w}_{\text{REAL}}$. Note that these labels now depend only on the relevant features. Depending on the chosen noise model, noise drawn from $N(0, 0.4\sigma_{\mathbf{y}})$ or $L(0, \frac{0.4\sigma_{\mathbf{y}}}{\sqrt{2}})$ was added to the labels, where $\sigma_{\mathbf{y}}$ is the standard deviation of the noise-less training labels.

3.4.2 Real-world QSAR Data

We examined four real-world regression chemoinformatics data sets: Aquasol, Blood/Brain Barrier (BBB), Cancer, and Cholecystokinin (CCK), previously studied in [21]. The goal is to create Quantitative Structure Activity Relationship (QSAR) models to predict bioactivites typically using the supplied descriptors as part of a drug design process. The data is scaled and preprocessed to reduce the dimensionality. As was done in [21], we standardize the data at each dimension and eliminate the uninformative variables

Data set	# Obs.	# Train	# Test	# Vars.	# Vars. (stdized)	# Vars. (postPCA)
AQUASOL	197	100	97	640	149	25
B/B BARRIER (BBB)	62	60	2	694	569	25
CANCER	46	40	6	769	362	25
CHOLECYSTOKININ (CCK)	66	60	6	626	350	25

Table 3.1: The Chemoinformatics (QSAR) data sets

that have values outside of ± 4 standard deviations range. Next, we perform principle components analysis (PCA), and use the top 25 principal components as descriptors. The training and hold out set sizes and the dimensionalities of the final data sets are shown in Table 3.1. For each of the training sets, 5-fold cross validation is optimized using bilevel programming. The results are averaged over 20 runs.

The LPs within each iterate in both SLA approaches were solved with CPLEX. The penalty parameter was uniformly set to $\mu = 10^3$ and never resulted in complementarity failure at termination. The hyper-parameters were bounded as $0.1 \leq C \leq 10$ and $0.01 \leq \varepsilon \leq 1$ so as to be consistent with the hyper-parameter ranges used in grid search. All computational times are reported in seconds.

3.4.3 Post-processing

The outputs from the bilevel approach and grid search yield the bound $\overline{\mathbf{w}}$ and the parameters C and ε. With these, we solve a constrained support vector problem on all the data points:

$$\begin{array}{ll} \text{minimize} & C \sum_{i=1}^{\ell} \max(\,|\,\mathbf{x}_i'\mathbf{w} - y_i\,| - \varepsilon, 0\,) + \frac{1}{2}\|\,w\,\|_2^2 \\ \text{subject to} & -\overline{\mathbf{w}} \leq \mathbf{w} \leq \overline{\mathbf{w}} \end{array}$$

to obtain the vector of model weights $\widehat{\mathbf{w}}$, which is used in computing the generalization errors on the hold-out data:

$$\text{MAD} \equiv \frac{1}{1000} \sum_{(\mathbf{x},y)\ \text{hold-out}} |\,\mathbf{x}'\widehat{\mathbf{w}} - y\,|$$

and

$$\text{MSE} \equiv \frac{1}{1000} \sum_{(\mathbf{x},y)\ \text{hold-out}} (\,\mathbf{x}'\widehat{\mathbf{w}} - y\,)^2.$$

The computation times, in seconds, for the different algorithms were also recorded.

3.5 Computational Results: Synthetic Data

In the following sections, constrained (abbreviated con.) methods refer to the bilevel models that have the box constraint $-\overline{\mathbf{w}} \leq \mathbf{w} \leq \overline{\mathbf{w}}$, while unconstrained (abbreviated unc.) methods refer to the bilevel models without the box constraint. In this section, we compare the performance of several different methods on synthetic data sets. The criteria used for comparing the various methods are training error (cross-validation objective), test error (generalization error measured as MAD or MSE on the 1000-point hold-out test set) and computation time (in seconds). For MAD and MSE, the results in bold refer to those that are significantly different than those of the unconstrained grid as measured by a two-sided t-test with significance of 0.1. The results that are significantly better and worse are tagged with a check (✓) or a cross (✗) respectively. The tables are appended at the end of the chapter.

From an optimization perspective, the bilevel programming methods consistently tend to outperform the grid search approaches significantly. The objective values found by the bilevel methods, especially FILTER, are much smaller than those found by their grid-search counterparts. The coarse grid size and feature elimination heuristics used in the grid search cause it to find relatively poor objective values.

The reported times provide a rough idea of the computational effort of each algorithm. As noted above, the computation times for the NEOS solver, FILTER, includes transmission, and waiting times as well as solve times. For grid search methods, smart restart techniques were used to gain a considerable increase in speed. However, for Con. Grid, even these techniques cannot prevent the running time from becoming impractical as the problem size grows. While the computation times of FILTER are both much less than that of Con. Grid, it is the SLA approaches that really dominate. The efficiency of the SLA approaches is vastly superior to both grid search and FILTER, especially for smaller problems. However, as problem size grows, Unc. SLAMS and Con. SLAMS become more expensive as bigger LPs have to be solved at each iteration while the algorithm converges. However, the early stopping methods, Unc. EZ-SLAMS and Con. EZ-SLAMS are very competitive, even as the problem size grows.

The bilevel approach is much more computationally efficient than grid search on the fully parameterized problems. The results, for FILTER, are relatively efficient and very

acceptable; the FILTER approach does have a drawback, in that is that it tends to struggle as the problem size increases. For the synthetic data, FILTER failed to solve 10 problems each from the 25d data sets with 120 and 150 points and these runs have been left out of Table 3.4.

Of course, in machine learning, an important measure of performance is generalization error. Compared to classic SVR optimized with Unc. Grid, FILTER and the SLA approaches yield solutions that are better or comparable to the test problems and never significantly worse. In contrast, the generalization performance of Con. Grid steadily degrades as problem size and dimensionality grow.

Finally, the SLA approaches that employ early stopping tend to generalize identically to the SLA approaches that do not stop early. This is a very important discovery because it suggests that allowing the SLA approaches to iterate to termination is very expensive, and it is without any corresponding improvement in the cross-validation objective or the generalization performance. The early stopping methods, Unc. EZ-SLAMS and Con. EZ-SLAMS are clearly competitive with the classical Unc. Grid approach with respect to training and generalization; their main advantage is their efficiency even when handling several hyper-parameters (which Unc. Grid is unable to do).

Thus, the Unc. and Con. EZ-SLAMS are the preferred methods. We investigate the behavior of Unc. SLAMS and Unc. EZ-SLAMS further in the following subsections.

3.5.1 Scalability Analysis

The purpose of this section is to study the scalability of the Unc. SLA methods to large data sets. The data set used here was a 10d synthetic data set with Laplacian noise. There were 11 different sets of problems solved with each set corresponding to different training set sizes: $\ell = 90,\ 180, \ldots, 990$. Each data set was trained on Unc. Grid, Unc. SLAMS and Unc. EZ-SLAMS. For each method, 10 random instances of the data set were solved using 3-fold cross validation and post-processed as described in Section 3.4.3. As before, the 3 methods were compared on training and test errors and computation times. Unc. SLAMS and Unc. EZ-SLAMS were also compared with regard to the number of LPs solved. The averaged results are shown in Figure 3.1.

It is interesting to note that the training and generalization behavior of all three algorithms are identical, especially as the data set size grows. However, the telling statistic is the computation time. For data set sizes up to 360 points, Unc. Grid performs compet-

itively with Unc. EZ-SLAMS and significantly better than Unc. SLAMS. However, beyond this, its efficiency drops off sharply as its computation time increases almost quadratically. Note that the computational times for all three approaches do not monotonically increase with the size and that the SLA methods are much more robust to increases in the data set size. Unsurprisingly, early stopping allows Unc. EZ-SLAMS to perform significantly better than Unc. SLAMS with no degradation in either the training or testing performance. We see that early stopping allows Unc. EZ-SLAMS to perform significantly better than Unc. SLAMS with no degradation in either the training or testing performance.

This is even more evident when the two SLA algorithms are compared with regard to the average number of LPs solved. Convergence to LPEC feasibility always occurs in less than 10 iterations after which Unc. EZ-SLAMS terminates. Unc. SLAMS, however, continues to solve more LPs and run for longer as it attempts to converge to a local solution. Again, this added computational time does not yield better generalization, suggesting that early stopping is indeed quite beneficial, especially for larger problems.

3.5.2 Folds Analysis

The purpose of this section is to determine the effect of increasing the number of folds in T-fold cross validation. Again, we use 5 different 10d data sets, each corresponding to data set sizes $\ell = 30, 60, \cdots, 150$. Each data set was trained on Unc. SLAMS and Unc. EZ-SLAMS using $T = 3$, 5 and 10 folds. There are 10 instances of each problem. The two methods are compared on generalization error and running time and the results are reported in Figure 3.2.

Rather unsurprisingly, as the number of folds, T, increase, the computation time increases and Unc. EZ-SLAMS continues to perform consistently better than Unc. SLAMS because it stops early. In fact, the former runs almost 4 times faster when solving 10-fold cross validation. Interestingly, there was no significant difference in the generalization performance of either method over the folds. This suggests that the commonly accepted norm of 10-fold cross validation is much more computationally expensive for the same generalization performance that 3-fold or 5-fold cross validation can deliver more quickly. This might be because classical cross-validation solves the training problems sequentially while the bilevel approaches solve them simultaneously which is clearly beneficial as all the information is embedded in the bilevel formulation.

3.6 Computational Results: QSAR Data

Table 3.5 shows the average results for the QSAR data. After the data is preprocessed, we randomly partition the data into 20 different training and testing sets. For each of the training sets, 5-fold cross validation is optimized using bilevel programming. The results are averaged over the 20 runs.

Again, as with the synthetic data, FILTER finds solutions with the smallest training errors. However, computation times for FILTER are not competitive with the SLA methods and not even with Unc. Grid. Unsurprisingly, constrained grid search has the worst computation time. The difficulty of the underlying bilevel optimization problem is underscored by the fact that the greedy Con. Grid search in Section 3.3.5 sometimes fails to find a better solution than the unconstrained grid search. The constrained search drops important variables that cause it to have bad generalization.

In terms of test set error, FILTER and the constrained SLA approaches outperform the unconstrained approaches on the cancer data and do as well on the remaining three data sets. However, on the remaining data sets, the SLA approaches generalize very well and tend to be competitive with Unc. Grid with regard to execution time. The best running times, however, are produced by the early stopping based SLA approaches, which SLAM the door on all other approaches while maintaining good generalization performance.

3.7 Chapter Conclusions

It was shown that the widely used model selection technique of cross validation (for support vector regression) could be formulated as a bilevel programming problem which is then converted to an LPEC. This class of problems is difficult to solve due to the nonconvexity created by the complementarity constraints introduced in the reformulation. Two approaches were proposed to solve the LPEC: a relaxed NLP-based approach which was solved using the off-the-shelf, SQP-based, NLP solver, FILTER and a exact penalty-based approach which was solved using a finite successive linearization algorithm.

Preliminary computational results indicate that general purpose SQP solvers can tractably find high-quality solutions that generalize well and this is consistent with results reported in Chapter 2 for classification. The computation times of the FILTER solver are good and generalization results on random data show that FILTER yields are comparable, if not better results than current methods. The successive linearization algorithms for

model selection (SLAMS) and their early stopping variants (EZ-SLAMS) performed even better than FILTER and demonstrated scalability to high dimensional data sets containing up to 1000 points. In fact, it was shown that the SLAMS algorithms comprehensively outperform classical grid search approaches as the size of the problem grows.

This is despite the fact that neither the NLP- or the SLA-based approaches take advantage of the structure inherent in bilevel problems arising from machine learning applications. Machine learning problems, especially support vector machines, are highly structured, and yield elegant and sparse solutions, a fact that several decomposition algorithms such as sequential minimal optimization target. Despite the non-convexity of the LPECs, bilevel programs for machine learning problems retain the structure inherent in the original machine learning problems. In addition, the variables in these LPECs tend to decouple, for example, in cross validation, the variables may be decoupled along the folds. This suggests that applying decomposition methods to bilevel approaches can make them even more efficient. An avenue for future research is developing a decomposition-based algorithm that can train on data sets containing tens of thousands of points.

The most pressing question, however, arises from a serious limitation of the formulation presented herein: the model can only handle linear data sets. Classical machine learning addresses this problem by means of the kernel trick. It will be shown in Chapter 5 that the kernel trick can be incorporated into a generic bilevel model for cross validation. In the next chapter, however, a problem that is germane to learning from real-world data is studied i.e., the problem of learning with missing data.

Method	Objective	Time	MAD	MSE
30 pts				
Unc. Grid	1.385 ± 0.323	5.2± 0.5	1.376	2.973
Unc. Slams	1.379 ± 0.290	1.4± 0.8	1.311	2.709
Unc. ez-Slams	1.475 ± 0.295	0.4± 0.1	**1.294** ✓	**2.642** ✓
Con. Grid	1.220 ± 0.276	635.3±59.1	1.391	3.044
filter (SQP)	0.984 ± 0.206	36.1±14.8	**1.298** ✓	**2.657** ✓
Con. Slams	1.183 ± 0.217	2.4± 0.9	1.320	2.746
Con. ez-Slams	1.418 ± 0.291	0.6± 0.1	1.308	2.684
60 pts				
Unc. Grid	1.200 ± 0.254	5.9± 0.5	1.208	2.324
Unc. Slams	1.235 ± 0.211	3.4± 2.9	1.181	2.223
Unc. ez-Slams	1.253 ± 0.218	1.2± 0.3	1.183	2.231
Con. Grid	1.143 ± 0.245	709.2±55.5	1.232	2.418
filter (SQP)	1.032 ± 0.197	35.2± 7.0	1.189	2.244
Con. Slams	1.191 ± 0.206	3.7± 2.4	1.186	2.239
Con. ez-Slams	1.232 ± 0.208	1.3± 0.3	1.186	2.238
90 pts				
Unc. Grid	1.151 ± 0.195	7.2± 0.5	1.180	2.215
Unc. Slams	1.206 ± 0.203	5.6± 2.8	**1.154** ✓	**2.124** ✓
Unc. ez-Slams	1.213 ± 0.208	2.6± 0.5	**1.154** ✓	**2.123** ✓
Con. Grid	1.108 ± 0.192	789.8±51.7	**1.163** ✓	**2.154** ✓
filter (SQP)	1.072 ± 0.186	44.2± 6.3	1.166	2.163
Con. Slams	1.188 ± 0.190	5.8± 2.6	**1.158** ✓	**2.140** ✓
Con. ez-Slams	1.206 ± 0.197	2.7± 0.8	**1.159** ✓	**2.139** ✓
120 pts				
Unc. Grid	1.124 ± 0.193	7.0± 0.1	1.144	2.087
Unc. Slams	1.133 ± 0.191	11.2± 8.6	1.141	2.085
Unc. ez-Slams	1.139 ± 0.190	5.0± 1.3	1.140	2.084
Con. Grid	1.095 ± 0.199	704.3±15.6	1.144	2.085
filter (SQP)	1.058 ± 0.188	50.2±10.9	1.150	2.114
Con. Slams	1.116 ± 0.193	15.6±15.3	1.141	2.082
Con. ez-Slams	1.137 ± 0.191	4.2± 1.1	1.143	2.089
150 pts				
Unc. Grid	1.091 ± 0.161	8.2± 0.3	1.147	2.098
Unc. Slams	1.108 ± 0.172	11.5± 3.3	1.134	2.055
Unc. ez-Slams	1.110 ± 0.172	7.8± 0.5	1.133	2.054
Con. Grid	1.068 ± 0.154	725.1± 2.7	1.142	2.081
filter (SQP)	1.045 ± 0.171	72.4±29.7	1.149	2.108
Con. Slams	1.103 ± 0.173	20.1± 5.5	1.136	2.063
Con. ez-Slams	1.110 ± 0.172	7.4± 1.1	1.136	2.062

Table 3.2: 10-d synthetic data with Laplacian and Gaussian noise under 3-fold cross validation. Results that are significantly better or worse are tagged ✓ or ✗ respectively.

Method	Objective	Time	MAD	MSE
30 pts				
Unc. Grid	1.995 ± 0.421	9.1± 9.3	1.726	4.871
Unc. Slams	2.041 ± 0.400	2.0± 0.5	1.697	4.722
Unc. ez-Slams	2.160 ± 0.453	0.7± 0.1	1.709	4.782
Con. Grid	1.659 ± 0.312	735.8±92.5	1.854	5.828
filter (SQP)	1.149 ± 0.216	23.4± 6.3	1.691	4.706
Con. Slams	1.497 ± 0.258	5.0± 1.3	1.675	4.596
Con. ez-Slams	1.991 ± 0.374	0.9± 0.2	1.697	4.716
60 pts				
Unc. Grid	1.613 ± 0.257	7.3± 1.3	1.584	4.147
Unc. Slams	1.641 ± 0.218	3.9± 2.0	1.511	3.863
Unc. ez-Slams	1.661 ± 0.225	1.8± 0.2	1.507	3.845
Con. Grid	1.520 ± 0.265	793.5±83.1	1.589	4.254
filter (SQP)	1.275 ± 0.159	39.7± 7.4	**1.495** ✓	**3.774** ✓
Con. Slams	1.565 ± 0.203	8.3± 3.5	1.504	3.820
Con. ez-Slams	1.673 ± 0.224	2.3± 0.3	1.498	3.807
90 pts				
Unc. Grid	1.553 ± 0.261	8.2± 0.5	1.445	3.553
Unc. Slams	1.528 ± 0.195	7.5± 5.3	1.423	3.455
Unc. ez-Slams	1.539 ± 0.192	3.4± 0.6	1.422	3.453
Con. Grid	1.575 ± 0.421	866.2±67.0	1.551	4.124
filter (SQP)	1.293 ± 0.183	58.4±15.4	1.434	3.511
Con. Slams	1.476 ± 0.182	16.3± 6.3	**1.411** ✓	**3.398** ✓
Con. ez-Slams	1.524 ± 0.197	3.8± 0.9	1.412	**3.404** ✓
120 pts				
Unc. Grid	1.481 ± 0.240	7.5± 0.0	1.396	3.350
Unc. Slams	1.478 ± 0.190	15.5± 8.0	1.374	**3.273** ✓
Unc. ez-Slams	1.488 ± 0.190	6.2± 0.9	1.374	**3.273** ✓
Con. Grid	1.432 ± 0.171	697.9± 2.2	1.395	3.333
filter (SQP)	1.304 ± 0.175	85.9±17.7	1.387	3.320
Con. Slams	1.419 ± 0.166	32.6±18.6	1.375	**3.273** ✓
Con. ez-Slams	1.474 ± 0.181	6.2± 0.8	1.379	3.291
150 pts				
Unc. Grid	1.448 ± 0.264	8.7± 0.1	1.362	3.221
Unc. Slams	1.457 ± 0.215	19.2±10.1	1.357	3.206
Unc. ez-Slams	1.460 ± 0.214	9.4± 1.5	1.357	3.204
Con. Grid	1.408 ± 0.232	723.2± 2.0	1.376	3.268
filter (SQP)	1.318 ± 0.205	114.2±48.6	1.367	3.229
Con. Slams	1.436 ± 0.217	41.8±17.5	1.360	3.214
Con. ez-Slams	1.459 ± 0.216	10.1± 1.8	1.359	3.206

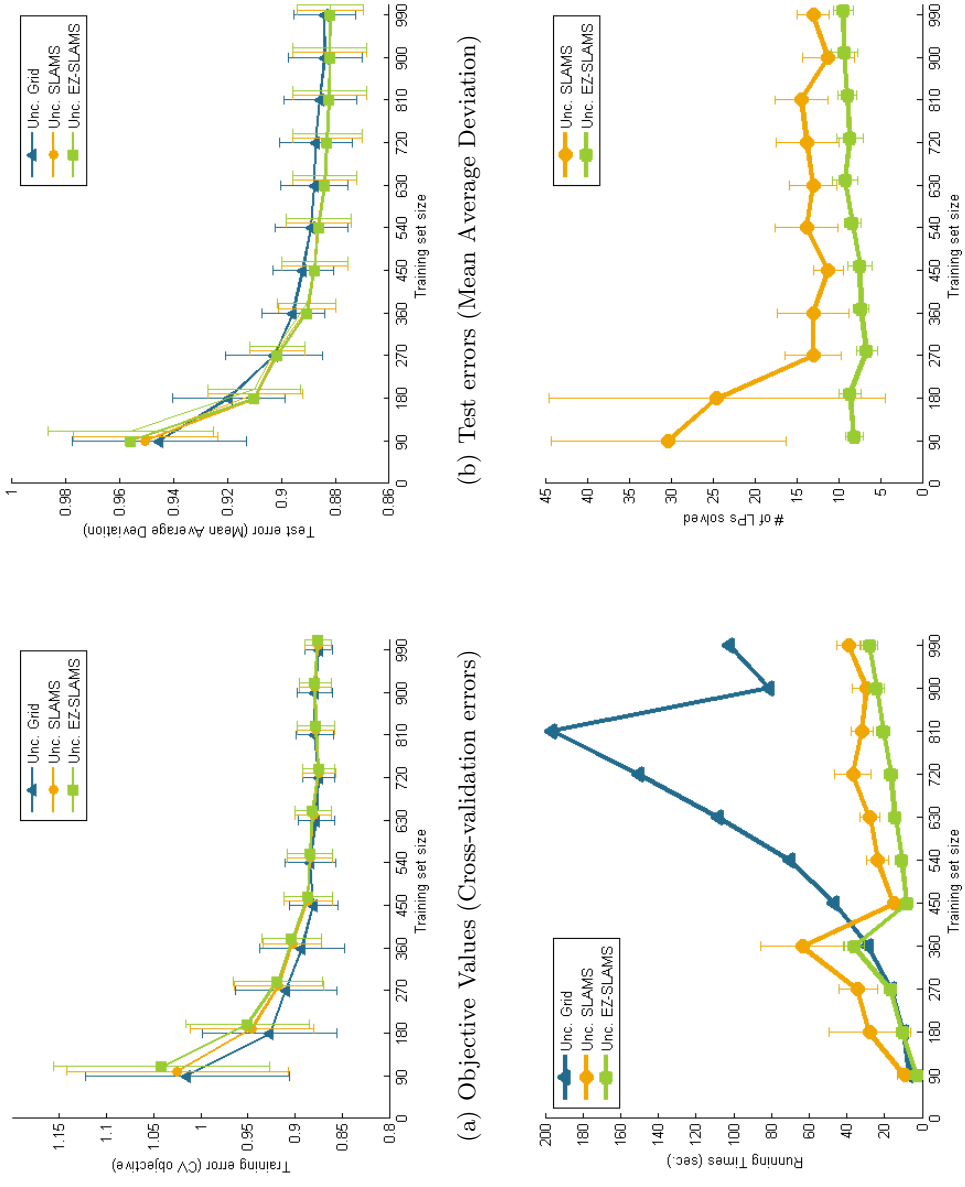
Table 3.3: 15-d synthetic data with Laplacian and Gaussian noise under 3-fold cross validation. Results that are significantly better or worse are tagged ✓ or ✗ respectively.

Method	Objective	Time	MAD	MSE
30 pts				
Unc. Grid	3.413 ± 0.537	5.7± 0.0	2.915	13.968
Unc. Slams	3.426 ± 0.780	6.2± 2.9	2.887	13.699
Unc. ez-Slams	4.503 ± 1.408	0.7± 0.3	3.052	15.269
Con. Grid	2.636 ± 0.566	628.5± 0.5	**3.687 ✗**	**22.065 ✗**
filter (SQP)	0.918 ± 0.242	47.9±42.4	**3.274 ✗**	**17.636 ✗**
Con. Slams	1.684 ± 0.716	7.9± 2.4	2.962	14.607
Con. ez-Slams	3.100 ± 0.818	1.5± 0.2	2.894	13.838
60 pts				
Unc. Grid	2.375 ± 0.535	6.2± 0.0	2.321	8.976
Unc. Slams	2.412 ± 0.462	8.1± 4.2	2.345	9.120
Unc. ez-Slams	2.461 ± 0.503	2.6± 0.5	2.375	9.353
Con. Grid	2.751 ± 0.653	660.9± 1.6	**3.212 ✗**	**16.734 ✗**
filter (SQP)	1.443 ± 0.280	78.5±29.9	2.361	9.261
Con. Slams	2.065 ± 0.469	15.3± 7.1	2.305	8.855
Con. ez-Slams	2.362 ± 0.441	3.1± 0.4	2.312	8.894
90 pts				
Unc. Grid	2.256 ± 0.363	7.0± 0.0	2.161	7.932
Unc. Slams	2.344 ± 0.409	17.4± 7.0	2.133	7.826
Unc. ez-Slams	2.381 ± 0.408	4.8± 0.8	2.134	7.836
Con. Grid	2.927 ± 0.663	674.7± 1.0	**3.117 ✗**	**15.863 ✗**
filter (SQP)	1.612 ± 0.245	146.5±54.6	2.101	7.564
Con. Slams	2.149 ± 0.304	29.3±12.2	2.119	7.711
Con. ez-Slams	2.328 ± 0.400	6.3± 1.2	2.131	7.803
120 pts				
Unc. Grid	2.147 ± 0.343	8.4± 0.0	2.089	7.505
Unc. Slams	2.242 ± 0.474	25.0±13.5	**2.031 ✓**	**7.133 ✓**
Unc. ez-Slams	2.267 ± 0.473	8.4± 0.8	**2.030 ✓**	**7.125 ✓**
Con. Grid	2.910 ± 0.603	696.7± 1.5	**3.124 ✗**	**15.966 ✗**
Con. Slams	2.156 ± 0.433	45.6±16.4	**2.028 ✓**	**7.121 ✓**
Con. ez-Slams	2.226 ± 0.461	10.3± 1.5	**2.034 ✓**	**7.154 ✓**
150 pts				
Unc. Grid	2.186 ± 0.383	9.9± 0.1	1.969	6.717
Unc. Slams	2.129 ± 0.378	34.7±14.6	1.952	6.645
Unc. ez-Slams	2.141 ± 0.378	12.3± 1.4	1.953	6.653
Con. Grid	2.759 ± 0.515	721.1± 1.7	**2.870 ✗**	**13.771 ✗**
Con. Slams	2.069 ± 0.368	63.5±30.5	1.949	6.636
Con. ez-Slams	2.134 ± 0.380	14.2± 2.5	**1.947 ✓**	6.619

Table 3.4: 25-d synthetic data with Laplacian and Gaussian noise under 3-fold cross validation. Results that are significantly better or worse are tagged ✓ or ✗ respectively.

Method	Objective	Time	MAD	MSE
Aquasol				
Unc. Grid	0.719 ± 0.101	17.1± 0.4	0.644	0.912
Unc. Slams	0.743 ± 0.098	89.3± 77.3	0.652	0.922
Unc. ez-Slams	0.755 ± 0.101	13.0± 2.1	0.653	0.933
Con. Grid	0.778 ± 0.094	1395.9± 3.5	**0.849 ✗**	**1.605 ✗**
filter (SQP)	0.551 ± 0.066	678.8±194.9	**0.702 ✗**	0.979
Con. Slams	0.670 ± 0.092	137.8± 52.0	0.647	0.911
Con. ez-Slams	0.710 ± 0.088	19.1± 3.3	0.643	0.907
Blood/Brain Barrier				
Unc. Grid	0.364 ± 0.048	13.4± 1.9	0.314	0.229
Unc. Slams	0.368 ± 0.042	8.8± 2.3	**0.292 ✓**	0.205
Unc. ez-Slams	0.370 ± 0.042	5.1± 0.6	**0.292 ✓**	0.205
Con. Grid	0.463 ± 0.081	1285.7±155.3	**0.733 ✗**	**0.856 ✗**
filter (SQP)	0.176 ± 0.013	121.3± 51.6	0.332	0.161
Con. Slams	0.363 ± 0.042	17.1± 9.8	0.312	0.231
Con. ez-Slams	0.370 ± 0.042	8.0± 1.6	0.315	0.235
Cancer				
Unc. Grid	0.489 ± 0.032	10.3± 0.9	0.502	0.472
Unc. Slams	0.637 ± 0.100	7.1± 2.6	0.518	0.396
Unc. ez-Slams	0.652 ± 0.100	3.1± 0.6	0.525	0.403
Con. Grid	0.477 ± 0.065	1035.3± 1.5	**0.611 ✗**	**0.653 ✗**
filter (SQP)	0.210 ± 0.025	100.2± 40.3	**0.395 ✓**	**0.252 ✓**
Con. Slams	0.476 ± 0.086	25.5± 9.2	0.481	**0.336 ✓**
Con. ez-Slams	0.567 ± 0.096	5.2± 1.1	0.483	**0.341 ✓**
Cholecystokinin				
Unc. Grid	0.798 ± 0.055	12.0± 0.4	1.006	1.625
Unc. Slams	0.988 ± 0.127	18.6± 7.9	**1.223 ✗**	**2.644 ✗**
Unc. ez-Slams	1.009 ± 0.144	6.4± 1.5	**1.225 ✗**	**2.670 ✗**
Con. Grid	0.783 ± 0.071	1157.6± 1.8	**1.280 ✗**	**2.483 ✗**
filter (SQP)	0.499 ± 0.031	189.8± 57.4	1.022	1.607
Con. Slams	0.881 ± 0.108	35.1± 20.4	**1.235 ✗**	**2.584 ✗**
Con. ez-Slams	0.941 ± 0.092	9.1± 1.3	**1.217 ✗**	**2.571 ✗**

Table 3.5: Results for QSAR data under 5-fold cross validation. Results that are significantly better or worse are tagged ✓ or ✗ respectively.



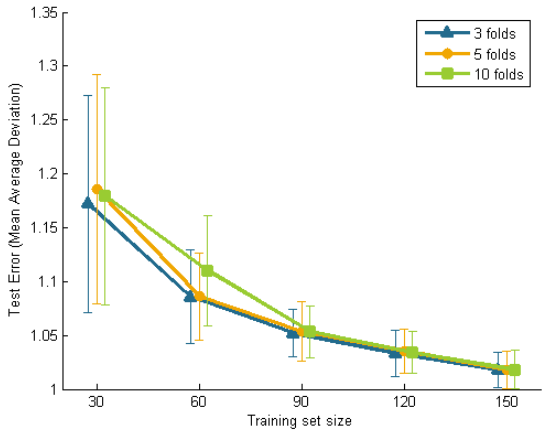
(a) Objective Values (Cross-validation errors)

(b) Test errors (Mean Average Deviation)

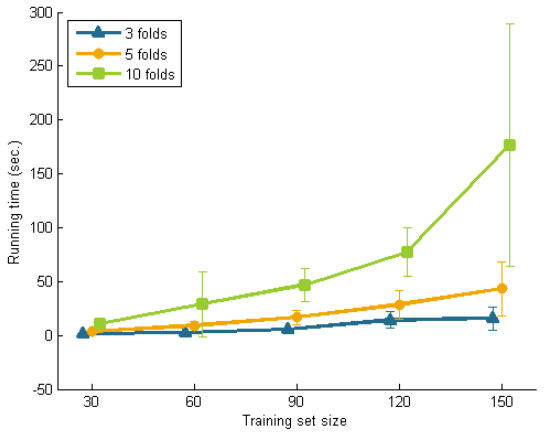
(c) Running Times

(d) Number of LPs solved by SLA

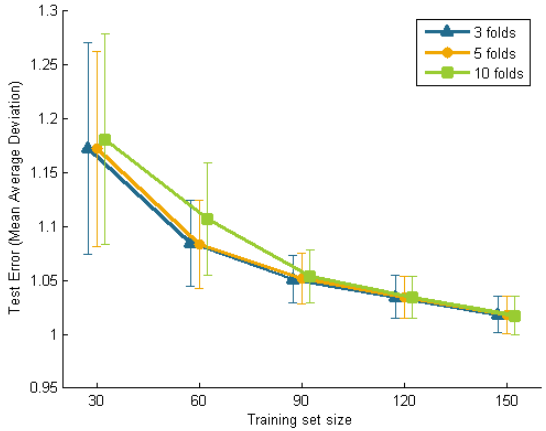
Figure 3.1: Scalability analysis for 10 instances of 10D data with 3-fold cross validation.



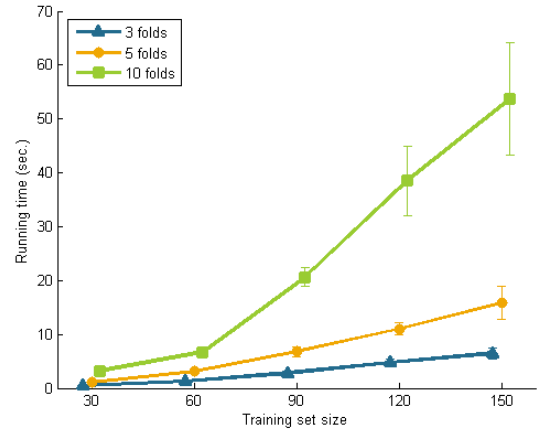
(a) Test Error for SLAMS



(b) Running Time for SLAMS



(c) Test Error for EZ-SLAMS



(d) Running Time for EZ-SLAMS

Figure 3.2: Folds Analysis for 10 instances of 10D data for $T = 3$, 5 and 10 folds.

CHAPTER 4
Missing Value Imputation via Bilevel Cross Validation

In the preceding sections, we have seen how parameter and feature selection via cross validation can be cast into the bilevel framework. Tasks like classification and regression are supervised learning tasks where in addition to the training labels, it is assumed that complete data features are available for learning. There are many real-world applications where this assumption is unrealistic and there might be missing labels (semi-supervised learning) or missing features. In this chapter, we consider the latter—which is still a supervised learning task—and formulate an approach to learning with missing data via bilevel programming.

4.1 Introduction

Missing data frequently occur in machine learning tasks because of various reasons. The most common reasons for learning tasks whose data comes from physical experiments is systemic measurement error or noise, human error or sampling errors. For example, in speech recognition tasks, the system may make errors so that the observed text is corrupted with noise. Proper handling of missing values is very important because, in addition to reduced sample size, it introduces the possibility that the remaining data set is biased.

Missingness in the data is of three types. Data are said to be *missing completely at random* (MCAR) if the probability that a value is missing is independent of any value, observed or missing. This type of missingness typically arises due to measurement errors and noise. Data are said to be *missing at random* (MAR) if the probability that a value is missing is related only to observed values and is conditionally independent of the missing values. Thus, MCAR exists when missing values are randomly distributed across all observations while MAR exists when missing values are not randomly distributed across all observations, rather they are randomly distributed within one or more sub-samples. These two cases of missingness are called *ignorable* which means that we do not have to model the missingness property. Alternately, we can also have *non-ignorable missingness at random* (NMAR) where the probability that a value is missing depends on the missing values. In this case, a learning model would have to include a model that accounts for the missing data.

There are several general approaches to handling data with missing values. When the proportion of the data with missing values is small, it is common to simply drop these points. This approach is called *Complete Case Analysis*[3] [73]. The most obvious disadvantages of this method are loss of efficiency and biases in the estimates, as mentioned above. Another approach is *imputation of missing data* [58, 74, 75, 76] which involves estimating the missing values or filling them in with plausible values. Then, the the analysis is carried out as if the imputed (estimated) values were the true data. This approach may reduce the bias of CCA or add bias of its own. A third approach is to formulate parametric models for the missing values and find the maximum likelihood estimates via a procedure such as expectation-maximization (EM) [24, 57]. The books by Little and Rubin [58] and Schafer [76] can be referenced for details regarding the approaches described and related techniques.

We consider the case of support vector regression. An alternative approach is proposed here where the missing values are imputed via bilevel support vector regression under cross-validation. In terms of the generalized formulation (1.41) the missing features in the data, $\hat{\mathbf{x}}$, are treated as hyper-parameters $\boldsymbol{\lambda}$ that are optimized in the outer-level, while solving T training problems under T-fold cross validation in the inner level. It is assumed that the data is missing at random (MAR) and the goal of the bilevel program is to determine the hyper-parameters C and ε as well as the missing data $\hat{\mathbf{x}}$ so as to obtain good generalization.

In addition to being a novel approach, this method is a holistic approach to estimating missing values which can be reformulated into an optimization problem. The approach is holistic in the sense that unlike prior approaches, imputation here is task-specific and tries to estimate the missing values by considering not only the available features but *also the labels*. Also, unlike most imputation methods, which are usually two phase procedures (estimation, to first fill in the missing attributes and learning, to train based on imputed data), the bilevel approach unifies these steps into one continuous optimization problem. The details of the approach are presented below.

[3] In the Statistics community, complete case analysis is abbreviated as CCA. To avoid confusion with Canonical Correlation Analysis, which is also abbreviated CCA in Machine Learning, complete case analysis is referred to as the baseline.

4.2 Notation and model

Let $\Omega_{\text{all}} := \{\mathbf{x}^i, y_i\}_{i=1}^{\ell} \in \mathbb{R}^{n+1}$ denote a given data set that may contain missing values i.e., missing components in $\mathbf{x}^i$ as well as missing labels, y_i. For a training vector, $\mathbf{x}^i$, we define the feature index sets $\mathcal{F}_i := \{j : x_j^i \text{ is known}\}$ and $\overline{\mathcal{F}}_i := \{j : x_j^i \text{ is missing}\}$. Then, $|\mathcal{F}_i| = n$ for training data with full feature information, while $|\mathcal{F}_i| < n$ for training data with missing feature information. Denote $\Psi := \{(\mathbf{x}^i, y_i) : |\mathcal{F}_i| < n, y_i \text{ is known}\} \subset \Omega_{\text{all}}$ to be the set of all points with missing $\mathbf{x}$-values only and $\Omega := \{(\mathbf{x}^i, y_i) : |\mathcal{F}_i| = n, y_i \text{ is known}\} \subset \Omega_{\text{all}}$ to be the set with complete information about features and labels. Without loss of generality, we assume that these sets are pairwise disjoint.

To incorporate T-fold cross validation onto the model, we partition Ω into T pairwise disjoint subsets, Ω_t, called the validation sets. We also define the training sets, $\overline{\Omega}_t := \Omega \setminus \Omega_t$. The index sets for Ω_t and $\overline{\Omega}_t$ are $\mathcal{N}_t$ and $\overline{\mathcal{N}}_t$ respectively. The set with missing data, Ψ is also partitioned into validation and training sets, Ψ_t and $\overline{\Psi}_t$, similar to the sets with full information. The index sets are $\mathcal{M}_t$ and $\overline{\mathcal{M}}_t$ respectively. Let $\mathcal{V} = \cup_{t=1}^{T}(\mathcal{M}_t \cup \overline{\mathcal{M}}_t)$. The vector of missing x-values is denoted $\hat{\mathbf{x}}$.

As was seen in Chapter 3, with a suitable choice of regularization and loss functions in the general formulation (1.41), we get the bilevel formulation (3.1). This model can be further modified to formulate the bilevel model for missing-value imputation. The box constraint parameter $\overline{\mathbf{w}}$ is dropped since we are only interested in selection of optimal parameters C and ε in addition to missing values $\hat{\mathbf{x}}$. This is achieved via T-fold cross validation if we

$$\begin{aligned}
&\underset{C,\varepsilon,\hat{\mathbf{x}}}{\text{minimize}} \quad \frac{1}{T}\sum_{t=1}^{T}\frac{1}{|\mathcal{M}_t \cup \mathcal{N}_t|}\left(\begin{array}{l}\displaystyle\sum_{i\in\mathcal{N}_t} \left|\, (\mathbf{x}^i)'\mathbf{w}^t - b_t - y_i \,\right| \\ \displaystyle + \sum_{i\in\mathcal{M}_t}\left|\sum_{f\in\mathcal{F}_i} x_f^i w_f^t + \sum_{f\in\overline{\mathcal{F}}_i} \boxed{\hat{x}_f^i w_f^t} - b_t - y_i\right|\end{array}\right) \qquad (4.1)\\
&\text{subject to} \quad \varepsilon, C, \geq 0,\\
&\text{and for} \quad t = 1,\ldots,T,
\end{aligned}$$

$$(\mathbf{w}^t, b_t) \in \arg\min\left\{\begin{array}{l} C\displaystyle\sum_{j\in\overline{\mathcal{N}}_t} \max(|(\mathbf{x}^j)'\mathbf{w} - b - y_j| - \varepsilon,\, 0) + \frac{1}{2}\,\|\,\mathbf{w}\,\|_2^2 \\ \displaystyle +C\sum_{j\in\overline{\mathcal{M}}_t} \max\left(\left|\sum_{f\in\mathcal{F}_j} x_f^j w_f + \sum_{f\in\overline{\mathcal{F}}_j}\boxed{\hat{x}_f^j w_f} - b - y_j\right| - \varepsilon, 0\right)\end{array}\right\}. \qquad (4.2)$$

where the terms containing the missing values are boxed. The features of $\mathbf{x}$ that are known are denoted x_f^j while the missing features, which are also outer-level variables, are denoted $\hat{x}_f^j$.

Now, in the t-th fold, the sets $\overline{\mathcal{N}}_t$ (full features) and $\overline{\mathcal{M}}_t$ (missing features) are used to train $(\mathbf{w}^t, b_t)$. Assuming the parameters C and ε and the missing values $\hat{\mathbf{x}}$ are fixed for the inner-level problems we can write down the complementarity conditions for the data points in $\overline{\mathcal{N}}_t$:

$$\left.\begin{aligned} &0 \le \alpha_j^{t,+} \perp y_j - (\mathbf{x}^j)'\mathbf{w}^t + b_t + \varepsilon + \xi_j^t \ge 0 \\ &0 \le \alpha_j^{t,-} \perp (\mathbf{x}^j)'\mathbf{w}^t - b_t - y_j + \varepsilon + \xi_j^t \ge 0 \\ &0 \le \xi_j^t \quad \perp C - \alpha_j^{t,+} - \alpha_j^{t,-} \ge 0 \end{aligned}\right\} \quad \forall\, j \in \overline{\mathcal{N}}_t. \tag{4.3}$$

The complementarity conditions for the data points in $\overline{\mathcal{M}}_t$ are shown below:

$$\left.\begin{aligned} &0 \le \alpha_j^{t,+} \perp y_j - \sum_{f\in\mathcal{F}_j} x_f^j w_f^t - \sum_{f\in\overline{\mathcal{F}}_j} \boxed{\hat{x}_f^j w_f^t} + b_t + \varepsilon + \xi_j^t \ge 0 \\ &0 \le \alpha_j^{t,-} \perp \sum_{f\in\mathcal{F}_j} x_f^j w_f^t + \sum_{f\in\overline{\mathcal{F}}_j} \boxed{\hat{x}_f^j w_f^t} - b_t - y_j + \varepsilon + \xi_j^t \ge 0 \\ &0 \le \xi_j^t \quad \perp C - \alpha_j^{t,+} - \alpha_j^{t,-} \ge 0 \end{aligned}\right\} \quad \forall\, j \in \overline{\mathcal{M}}_t. \tag{4.4}$$

We also have the first order conditions for the t-th inner-level problem:

$$\begin{aligned} &w_f^t + \sum_{j\in\overline{\mathcal{M}}_t\cup\overline{\mathcal{N}}_t\,:\, f\in\mathcal{F}_j} (\alpha_j^{t,+} - \alpha_j^{t,-})\, x_f^j + \sum_{j\in\overline{\mathcal{M}}_t\,:\, f\in\overline{\mathcal{F}}_j} \boxed{(\alpha_j^{t,+} - \alpha_j^{t,-})\, \hat{x}_f^j} = 0, \quad \forall\, f = 1\ldots n, \\ &\sum_{j\in\overline{\mathcal{M}}_t\cup\overline{\mathcal{N}}_t} (\alpha_j^{t,+} - \alpha_j^{t,-}) = 0. \end{aligned} \tag{4.5}$$

The equality conditions (4.5) and the complementarities (4.3–4.4) together constitute the first-order KKT conditions for the lower-level problem, with $\hat{\mathbf{x}}$, C and ε fixed. We can validate the hyperplane $(\mathbf{w}^t, b_t)$ trained on $\overline{\mathcal{M}}_t \cup \overline{\mathcal{N}}_t$ on the t-th validation set $M_t \cup N_t$ using the mean average deviation (MAD). This is achieved by the outer-level objective piece (4.1), where the absolute value function is not differentiable. As before, we introduce variables $z_i^t \ge 0$ for each validation point and measure the validation error through the

constraints below; for validation points with full information, $\mathcal{N}_t$,

$$\left.\begin{array}{l} y_i - (\mathbf{x}^i)'\mathbf{w}^t + b_t + z_i^t \geq 0 \\ (\mathbf{x}^i)'\mathbf{w}^t - b_t - y_i + z_i^t \geq 0 \end{array}\right\} \quad \forall\, i \in \mathcal{N}_t, \tag{4.6}$$

and for validation points with missing values, $\mathcal{M}_t$,

$$\left.\begin{array}{l} y_i - \displaystyle\sum_{f\in\mathcal{F}_i} x_f^i w_f^t - \sum_{f\in\overline{\mathcal{F}}_i} \boxed{\hat{x}_f^i w_f^t} + b_t + z_i^t \geq 0 \\ \displaystyle\sum_{f\in\mathcal{F}_i} x_f^i w_f^t + \sum_{f\in\overline{\mathcal{F}}_i} \boxed{\hat{x}_f^i w_f^t} - b_t - y_i + z_i^t \geq 0 \end{array}\right\} \quad \forall\, i \in \mathcal{M}_t. \tag{4.7}$$

Putting the above equations together, we have the model below that is intended to perform cross validation to determine not only the hyper-parameters in the regression model but also to identify all the missing data and labels.

$$\begin{array}{ll} \text{minimize} & \dfrac{1}{T}\displaystyle\sum_{t=1}^{T} \frac{1}{|\mathcal{M}_t \cup \mathcal{N}_t|} \sum_{i\in\mathcal{M}_t\cup\mathcal{N}_t} z_i^t \\ \text{subject to} & \varepsilon \in [\underline{\varepsilon}, \overline{\varepsilon}], \quad C \in [\underline{C}, \overline{C}], \\ & \hat{x}_f^j \in [\underline{x}, \overline{x}],\ j \in \mathcal{V}, f \in \overline{\mathcal{F}}_j, \\ & \text{for all } t = 1, \ldots, T, \\ & \left\{\begin{array}{l} \text{complementarity conditions (4.3)–(4.4)} \\ \text{equality conditions (4.5)} \\ \text{validation constraints (4.6)–(4.7)} \end{array}\right\} \end{array} \tag{4.8}$$

In the model above, the terms in the boxes represent bilinear terms which arise from the product between the missing features $\hat{x}_f^j$ and the primal components w_f^t or the dual components $(\alpha_j^{t,+} - \alpha_j^{t,-})$. These bilinear terms give rise to nonlinear complementarity conditions for the training points that contain missing data. In order to apply successive linearization, it is necessary to either linearize the constraints or move them into the objective via additional variables and a penalty function. We will perform the latter.

First, we introduce variables $r_f^{t,j}$ and $s_f^{t,j}$ to represent the bilinear "primal" and "dual" products. These enter (4.8) as (bilinear) non-convex, indefinite quadratic con-

straints.

$$\begin{aligned} r_f^{t,j} &= (\alpha_j^{t,+} - \alpha_j^{t,-})\,\hat{x}_f^j, \quad \forall t = 1, \cdots, T,\ j \in \overline{\mathcal{M}}_t,\ f \in \overline{\mathcal{F}}_j, \\ s_f^{t,j} &= w_f^t\,\hat{x}_f^j, \qquad\qquad \forall t = 1, \cdots, T,\ j \in (\mathcal{M}_t \cup \overline{\mathcal{M}}_t),\ f \in \overline{\mathcal{F}}_j. \end{aligned} \tag{4.9}$$

There are more $s_f^{t,j}$ variables because the products $w_f^t \hat{x}_f^j$ exist for both training and validation points with missing data while the $r_f^{t,j}$ variables represent $(\alpha_j^{t,+} - \alpha_j^{t,-})\hat{x}_f^j$ which exist only for training points with missing data. Thus, we introduce variables $r_f^{t,j}$ and $s_f^{t,j}$ and equations (4.9) to convert the first order complementarity and equality conditions in (4.8) into a linear complementary system.

The purpose of introducing equations (4.9) is to ensure that all the constraints including complementarities remain bilinear. The idea here is that the methods that have already been successful on similar MPECs—such as FILTER and SLA—can be used here.

4.3 Bilevel Optimization Methods

In this section, we describe two alternative methods for solving the model. In keeping with the proof-of-concept ideas, the off-the-shelf solver FILTER [31] on the online server NEOS [20] will be used to solve an inexact relaxed version of (4.8). Similarly, successive linearization is used to solve a penalized version of the problem over a linear, polyhedral set. The details of these methods are presented below.

4.3.1 Inexact Imputation

As seen in the previous chapters, this method employs a relaxation of nonlinear constraints and requires them to be satisfied within some prescribed tolerance, **tol**. There are two sources of hard constraints in the program: the linear complementarities and the bilinear equality constraints (4.9).

As in Section 3.3.1, the hard complementarities $a \perp b$ are replaced with their relaxed counterparts $a \perp_{\mathbf{tol}} b$ which requires the complementarities to hold within a tolerance i.e., $a'b \leq \mathbf{tol}$. The constraints (4.9), however, are equality constraints that depend on unbounded variables. Thus, these constraints of the type $a - b'c = 0$ are relaxed as $a - b'c =_{\mathbf{tol}} 0$, which compactly denotes $|a - b'c| \leq \mathbf{tol}$. This notation indicates that the relaxation actually consists of two constraints, $a - b'c \leq \mathbf{tol}$ and $a - b'c \geq -\mathbf{tol}$. Incorporating these relaxations yields a nonlinear program which can inexactly determine not only the missing values but also the parameters C and ε via cross validation so as to optimize

the cross-validation error in the objective. We term this approach *inexact imputation*. As with other inexact models in this thesis it is solved using FILTER.

$$
\begin{aligned}
\text{minimize} \quad & \frac{1}{T}\sum_{t=1}^{T}\frac{1}{|\mathcal{M}_t \cup \mathcal{N}_t|}\sum_{i\in\mathcal{M}_t\cup\mathcal{N}_t} z_i^t \\
\text{subject to} \quad & \varepsilon \in [\underline{\varepsilon}, \overline{\varepsilon}], \quad C \in [\underline{C}, \overline{C}], \quad \hat{x}_f^j \in [\underline{x}, \overline{x}],\ j \in \mathcal{V}, f \in \overline{\mathcal{F}}_j, \\
& \text{for all } t = 1, \ldots, T, \\
& \left.\begin{array}{l} y_i - (\mathbf{x}^i)'\mathbf{w}^t + b_t + z_i^t \geq 0 \\ (\mathbf{x}^i)'\mathbf{w}^t - b_t - y_i + z_i^t \geq 0 \end{array}\right\} \quad \forall\, i \in \mathcal{N}_t, \\
& \left.\begin{array}{l} y_i - \sum_{f\in\mathcal{F}_i} x_f^i w_f^t - \sum_{f\in\overline{\mathcal{F}}_i} s_f^{t,i} + b_t + z_i^t \geq 0 \\ \sum_{f\in\mathcal{F}_i} x_f^i w_f^t + \sum_{f\in\overline{\mathcal{F}}_i} s_f^{t,i} - b_t - y_i + z_i^t \geq 0 \end{array}\right\} \quad \forall\, i \in \mathcal{M}_t, \\
& \left.\begin{array}{l} 0 \leq \alpha_j^{t,+} \perp_{\mathbf{tol}} y_j - (\mathbf{x}^j)'\mathbf{w}^t + b_t + \varepsilon + \xi_j^t \geq 0 \\ 0 \leq \alpha_j^{t,-} \perp_{\mathbf{tol}} (\mathbf{x}^j)'\mathbf{w}^t - b_t - y_j + \varepsilon + \xi_j^t \geq 0 \\ 0 \leq \xi_j^t \quad \perp_{\mathbf{tol}} C - \alpha_j^{t,+} - \alpha_j^{t,-} \geq 0 \end{array}\right\} \quad \forall\, j \in \overline{\mathcal{N}}_t, \\
& \left.\begin{array}{l} 0 \leq \alpha_j^{t,+} \perp_{\mathbf{tol}} y_j - \sum_{f\in\mathcal{F}_j} x_f^j w_f^t - \sum_{f\in\overline{\mathcal{F}}_j} s_f^{t,j} + b_t + \varepsilon + \xi_j^t \geq 0 \\ 0 \leq \alpha_j^{t,-} \perp_{\mathbf{tol}} \sum_{f\in\mathcal{F}_j} x_f^j w_f^t + \sum_{f\in\overline{\mathcal{F}}_j} s_f^{t,j} - b_t - y_j + \varepsilon + \xi_j^t \geq 0 \\ 0 \leq \xi_j^t \quad \perp_{\mathbf{tol}} C - \alpha_j^{t,+} - \alpha_j^{t,-} \geq 0 \end{array}\right\} \quad \forall\, j \in \overline{\mathcal{M}}_t, \\
& r_f^{t,j} - (\alpha_j^{t,+} - \alpha_j^{t,-})\, x_f^j =_{\mathbf{tol}} 0, \quad \forall\, t = 1, \cdots, T,\ j \in \overline{\mathcal{M}}_t,\ f \in \overline{\mathcal{F}}_j, \\
& s_f^{t,j} - w_j^t\, x_f^j =_{\mathbf{tol}} 0, \qquad\qquad \forall\, t = 1, \cdots, T,\ j \in \mathcal{M}_t \cup \overline{\mathcal{M}}_t,\ f \in \overline{\mathcal{F}}_j, \\
& w_f^t + \sum_{j\in\overline{\mathcal{M}}_t\cup\overline{\mathcal{N}}_t\,:\, f\in\mathcal{F}_j} (\alpha_j^{t,+} - \alpha_j^{t,-})\, x_f^j + \sum_{j\in\overline{\mathcal{M}}_t\,:\, f\in\overline{\mathcal{F}}_j} r_f^{t,j} = 0, \quad \forall\, f = 1 \ldots n, \\
& \sum_{j\in\overline{\mathcal{M}}_t\cup\overline{\mathcal{N}}_t} (\alpha_j^{t,+} - \alpha_j^{t,-}) = 0.
\end{aligned}
\tag{4.10}
$$

4.3.2 Penalty Formulation

The other approach considered is the successive linearization approach which was successfully applied to construct local solutions to the LPEC that arose from bilevel model selection for regression in Section 3.3.3. This was done by lifting the bilinear complemen-

tarity constraints into the objective via an exact quadratic penalty which was linearized by the Frank-Wolfe (FW) method. However, for the problem at hand, in addition to bilinearities that arise from the linear complementarities, the equality constraints (4.9) also pose a significant difficulty. Rather than linearize the constraints, it is preferred that they be linearized in the objective in order to maintain a polyhedral set of constraints.

This is achieved by lifting the equality constraints (4.9) into the objective via a squared penalty function. Now, the complexity of the penalized objective (4.12) is increased significantly as it is quartic (degree 4).

$$\begin{aligned} Q_t^r(\boldsymbol{\alpha}^{t,\pm}, \hat{\mathbf{x}}, \mathbf{r}^t) &= \frac{1}{2} \sum_{j \in \overline{\mathcal{M}}_t \,:\, f \in \overline{\mathcal{F}}_j} \left(r_f^{t,j} \; - \; (\alpha_j^{t,+} - \alpha_j^{t,-})\, \hat{x}_f^j \right)^2, \\ Q_t^s(\mathbf{w}^t, \hat{\mathbf{x}}, \mathbf{s}^t) &= \frac{1}{2} \sum_{j \in \mathcal{V} \,:\, f \in \overline{\mathcal{F}}_j} (s_f^{t,j} \; - \; \hat{x}_f^j w_f^t)^2. \end{aligned} \tag{4.11}$$

The linear complementarities in (4.8). can be handled via the quadratic exact penalty functions (as seen in Section 3) yielding the quadratic complementarity penalty:

$$\begin{aligned} P_t(\boldsymbol{\zeta}^t, \varepsilon, C) = \;& \varepsilon \sum_{j \in \overline{\mathcal{M}}_t \cup \overline{\mathcal{N}}_t} (\alpha_j^{t,+} + \alpha_j^{t,-}) + C \sum_{j \in \overline{\mathcal{M}}_t \cup \overline{\mathcal{N}}_t} \xi_j^t + \sum_{j \in \overline{\mathcal{M}}_t \cup \overline{\mathcal{N}}_t} (\alpha_j^{t,+} - \alpha_j^{t,-})\, y_j \\ & - \sum_{j \in \overline{\mathcal{N}}_t} (\alpha_j^{t,+} - \alpha_j^{t,-})(\mathbf{x}^j)' \mathbf{w}^t - \sum_{j \in \overline{\mathcal{M}}_t} \sum_{f \in \mathcal{F}_j} (\alpha_j^{t,+} - \alpha_j^{t,-}) x_f^j w_f^t \\ & - \sum_{j \in \overline{\mathcal{M}}_t} \sum_{f \in \overline{\mathcal{F}}_j} (\alpha_j^{t,+} - \alpha_j^{t,-}) s_f^{t,j}. \end{aligned} \tag{4.12}$$

where all the inner-level variables are collected into $\boldsymbol{\zeta}^t \equiv [\boldsymbol{\alpha}^{t,\pm}, \boldsymbol{\xi}^t, \mathbf{w}^t, b_t, \mathbf{r}^t, \mathbf{s}^t]$. When all the complementarities are removed from (4.8) and the bilinear variables are replaced using (4.9), we get the following constraints. For the t-th training set, the regression training constraints for known data are

$$\left.\begin{array}{l} y_j - (\mathbf{x}^j)' \mathbf{w}^t + b_t + \varepsilon + \xi_j^t \geq 0 \\ (\mathbf{x}^j)' \mathbf{w}^t - b_t - y_j + \varepsilon + \xi_j^t \geq 0 \\ C - \alpha_j^{t,+} - \alpha_j^{t,-} \geq 0 \end{array}\right\} \; \forall\, j \in \overline{\mathcal{N}}_t, \tag{4.13}$$

and for the missing data are

$$\left.\begin{array}{l} y_j - \sum_{f\in\mathcal{F}_j} x_f^j w_f^t - \sum_{f\in\overline{\mathcal{F}}_j} s_f^{t,j} + b_t + \varepsilon + \xi_j^t \geq 0 \\ \sum_{f\in\mathcal{F}_j} x_f^j w_f^t + \sum_{f\in\overline{\mathcal{F}}_j} s_f^{t,j} - b_t - y_j + \varepsilon + \xi_j^t \geq 0 \\ C - \alpha_j^{t,+} - \alpha_j^{t,-} \geq 0 \end{array}\right\} \forall\, j \in \overline{\mathcal{M}}_t. \tag{4.14}$$

Similarly, for the t-th validation set, the validation constraints for known data are

$$\left.\begin{array}{l} y_i - (\mathbf{x}^i)'\mathbf{w}^t + b_t + z_i^t \geq 0 \\ (\mathbf{x}^i)'\mathbf{w}^t - b_t - y_i + z_i^t \geq 0 \end{array}\right\} \forall\, i \in \mathcal{N}_t, \tag{4.15}$$

and for the missing data are

$$\left.\begin{array}{l} y_i - \sum_{f\in\mathcal{F}_i} x_f^i w_f^t - \sum_{f\in\overline{\mathcal{F}}_i} s_f^{t,i} + b_t + z_i^t \geq 0 \\ \sum_{f\in\mathcal{F}_i} x_f^i w_f^t + \sum_{f\in\overline{\mathcal{F}}_i} s_f^{t,i} - b_t - y_i + z_i^t \geq 0 \end{array}\right\} \forall\, i \in \mathcal{M}_t. \tag{4.16}$$

We also have the first-order constraints:

$$\begin{array}{l} w_f^t + \sum_{j\in\overline{\mathcal{M}}_t\cup\overline{\mathcal{N}}_t\,:\,f\in\mathcal{F}_j} (\alpha_j^{t,+} - \alpha_j^{t,-})\, x_f^j + \sum_{j\in\overline{\mathcal{M}}_t\,:\,f\in\overline{\mathcal{F}}_j} r_f^{t,j} = 0, \quad \forall\, f = 1\ldots n, \\ \sum_{j\in\overline{\mathcal{M}}_t\cup\overline{\mathcal{N}}_t} (\alpha_j^{t,+} - \alpha_j^{t,-}) = 0, \quad \boldsymbol{\alpha}^{t,\pm}, \boldsymbol{\xi}^t \geq 0. \end{array} \tag{4.17}$$

The constraint set S_t can now be defined as the polyhedral set defined by (4.13)–(4.17).The constraint set S_0 is defined as:

$$S_0 \equiv \left\{ \begin{array}{l} \varepsilon \in [\underline{\varepsilon}, \overline{\varepsilon}], \quad C \in [\underline{C}, \overline{C}], \\ \hat{x}_f^j \in [\underline{x}, \overline{x}],\ j \in \mathcal{V}, f \in \overline{\mathcal{F}}_j \end{array} \right\}. \tag{4.18}$$

Using (4.13)–(4.17) and (4.18) we can define the set $S_{\mathsf{LP}} = \bigcap_{t=0}^{T} S_t$ as the feasible region where the linearized penalty problem may be solved. The penalty problem, with appro-

priate penalty parameters μ and λ is described below:

$$\begin{array}{ll} \text{minimize} & \Theta(\mathbf{z}) + \mu \sum_{t=1}^{T} P_t(\boldsymbol{\zeta}^t, \varepsilon, C) + \lambda \sum_{t=1}^{T} \left(Q_t^r(\boldsymbol{\alpha}^{t,\pm}, \hat{\mathbf{x}}, \mathbf{r}^t) \; + \; Q_t^s(\mathbf{w}^t, \hat{\mathbf{x}}, \mathbf{s}^t) \right) \\ \text{subject to} & (\mathbf{z}, \boldsymbol{\zeta}, \hat{\mathbf{x}}, C, \varepsilon) \in S_{\mathsf{LP}}. \end{array} \tag{4.19}$$

where the LPEC objective is the mean average deviation measured on the validation sets, $\mathcal{M}_t \cup \mathcal{N}_t$. Thus, the goal of the LPEC is to minimize

$$\Theta(\mathbf{z}) = \frac{1}{T} \sum_{t=1}^{T} \frac{1}{|\mathcal{M}_t \cup \mathcal{N}_t|} \sum_{i \in \mathcal{M}_t \cup \mathcal{N}_t} z_i^t. \tag{4.20}$$

4.3.3 Bounding the Feasible Region

There are two key assumptions that successive linearization requires: that the objective is bounded below by zero and that there be no lines going to infinity on both sides. The latter assumption does not hold for the LPEC (4.8) or for the penalty problem (4.19) because the variables $\mathbf{r}^t$ and $\mathbf{s}^t$ are unbounded. This necessitates the introduction of additional bounds on $\mathbf{r}^t$ and $\mathbf{s}^t$ such that the linearized problem is not unbounded.

It is known that $-C \leq (\alpha_j^{t,+} - \alpha_j^{t,-}) \leq C$ and that the hyper-parameter $C \in [\underline{C}, \overline{C}]$, where $\underline{C}$ and $\overline{C}$ are user-defined. Using this, the definition (4.18) and introducing user-defined bounds on the variables $\mathbf{w}^t$, we have the following bounds on the variables that appear in the quartic penalty terms,

$$\begin{array}{ll} -\overline{C} \leq (\alpha_j^{t,+} - \alpha_j^{t,-}) \leq \overline{C}, & \forall\, t = 1 \ldots T,\; j \in \overline{\mathcal{M}}_t \cup \overline{\mathcal{N}}_t, \\ \underline{x}_f \leq \hat{x}_f^j \leq \overline{x}_f, & \forall\, j \in \mathcal{V},\; f \in \overline{\mathcal{F}}_j, \\ \underline{w}_f \leq w_f^t \leq \overline{w}_f, & \forall\, t = 1 \ldots T,\; f = 1 \ldots n. \end{array} \tag{4.21}$$

We wish to derive a bound on a, given a bilinear constraint of the form $a = bc$, where $(b, c) \in [\underline{b}, \overline{b}] \times [\underline{c}, \overline{c}]$. A coarse approximated bound on a can be computed based on a relaxation of the hard constraint as $a \in [\underline{bc}, \overline{bc}]$; relaxations such as this have been used in branch-and-bound approaches [40]. However, a much tighter relaxation can be derived based on the Taylor series expansion of $a = bc$ to give the bounds [45, 82]:

$$\max\{\underline{c}b + \underline{b}c - \underline{bc},\; \overline{c}b + \overline{b}c - \overline{bc}\} \;\leq\; a \;\leq\; \min\{\overline{c}b + \underline{b}c - \underline{b}\overline{c},\; \underline{c}b + \overline{b}c - \overline{b}\underline{c}\} \tag{4.22}$$

In fact, it was shown in [2, 86] that the functions max and min in (4.22) are concave and involve convex envelopes of the bilinear function $a = bc$. More specifically, (4.22) can be expressed as the constraint

$$(b,\, c,\, a) \;\in\; \text{convex hull} \left\{ \begin{array}{ll} (\underline{b},\, \underline{c},\, \underline{bc}), & (\underline{b},\, \overline{c},\, \underline{b}\overline{c}), \\ (\overline{b},\, \underline{c},\, \overline{b}\underline{c}), & (\overline{b},\, \overline{c},\, \overline{bc}) \end{array} \right\}. \tag{4.23}$$

While (4.23) is an explicit description of (4.22), the latter is easier to implement. Using (4.9), the bounds (4.21) and (4.22), we derive the following bounds on $r_f^{t,j}$,

$$\left.\begin{array}{l} r_f^{t,j} \geq \underline{x}_f(\alpha_j^{t,+} - \alpha_j^{t,-}) - \overline{C}\hat{x}_f^j + \overline{C}\underline{x}_f \\ r_f^{t,j} \geq \overline{x}_f(\alpha_j^{t,+} - \alpha_j^{t,-}) + \overline{C}\hat{x}_f^j - \overline{C}\overline{x}_f \\ r_f^{t,j} \leq \overline{x}_f(\alpha_j^{t,+} - \alpha_j^{t,-}) - \overline{C}\hat{x}_f^j + \overline{C}\overline{x}_f \\ r_f^{t,j} \leq \underline{x}_f(\alpha_j^{t,+} - \alpha_j^{t,-}) + \overline{C}\hat{x}_f^j - \overline{C}\underline{x}_f \end{array}\right\} \forall\, t = 1 \ldots T,\, j \in \overline{\mathcal{M}}_t,\, f \in \overline{\mathcal{F}}_j, \tag{4.24}$$

and the following bounds on $s_f^{t,j}$,

$$\left.\begin{array}{l} s_f^{t,j} \geq \underline{x}_f w_f^t + \underline{w}_f \hat{x}_f^j - \underline{x}_f \underline{w}_f \\ s_f^{t,j} \geq \overline{x}_f w_f^t + \overline{w}_f \hat{x}_f^j - \overline{x}_f \overline{w}_f \\ s_f^{t,j} \leq \overline{x}_f w_f^t + \underline{w}_f \hat{x}_f^j - \overline{x}_f \underline{w}_f \\ s_f^{t,j} \leq \underline{x}_f w_f^t + \overline{w}_f \hat{x}_f^j - \underline{x}_f \overline{w}_f \end{array}\right\} \forall\, t = 1 \cdots T,\, j \in \mathcal{M}_t \cup \overline{\mathcal{M}}_t,\, f \in \overline{\mathcal{F}}_j. \tag{4.25}$$

These bounds in (4.24)-(4.25) are represented by the constraint set S_{Bds}. In order to ensure that there are no lines going to infinity on both sides when applying successive linearization, the polyhedral constraint set can be closed using the bounds i.e., by solving on $S_{\mathsf{LP}} \cap S_{\mathsf{Bds}}$. The AMPL model (constraints and objectives) for this approach is given in Appendix A.3.

4.3.4 Successive Linearization

As in Chapter 3, we employ the successive linearization algorithm to solve the penalized problem (4.19) on the bounded feasible region constructed as explained in Section 4.3.3. The successive linearization procedure is as shown in the Algorithm 4.1, where, with a slight abuse of notation, $\boldsymbol{\zeta}^k$ denotes the vector containing all the variables of the

Algorithm 4.1 Successive linearization algorithm for missing-value imputation.
Fix μ, $\lambda > 0$.

1. *Initialization*:
   Start with an initial point, $\boldsymbol{\zeta}^0 \in S_{\mathsf{LP}} \cap S_{\mathsf{Bds}}$.

2. *Solve Linearized Problem*:
   Generate an intermediate iterate, $\bar{\boldsymbol{\zeta}}^k$, from the previous iterate, $\boldsymbol{\zeta}^k$, by solving the linearized penalty problem, $\bar{\boldsymbol{\zeta}}^k \in \underset{\boldsymbol{\zeta} \in S_{\mathsf{LP}} \cap S_{\mathsf{Bds}}}{\arg \operatorname{vertex} \min} \nabla_{\boldsymbol{\zeta}} P(\boldsymbol{\zeta}^k; \mu, \lambda)' (\boldsymbol{\zeta} - \boldsymbol{\zeta}^k)$.

3. *Termination Condition*:
   Stop if an appropriate termination condition holds.

4. *Compute Step Size*:
   Compute step length $\tau \in \underset{0 \le \tau \le 1}{\arg \min} P\Big((1-\tau)\, \boldsymbol{\zeta}^k + \tau\, \bar{\boldsymbol{\zeta}}^k; \mu, \lambda \Big)$,
   and get the next iterate, $\boldsymbol{\zeta}^{k+1} = (1-\tau)\, \boldsymbol{\zeta}^k + \tau\, \bar{\boldsymbol{\zeta}}^k$.

problem. The objective of the penalty formulation in (4.19) is denoted P.

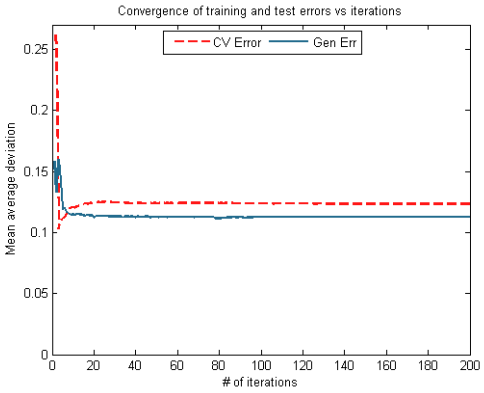
The main difference between this version of the algorithm and the one proposed in Chapter 3 is that the termination criteria employed are much more relaxed. This is because the contours of the objective contain steep and narrow valley-like regions which causes a descent algorithm like SLA to zig-zag as it approaches a local minima (see Figure 4.1d). Consequently, the algorithm converges very slowly. A common loose termination criterion that is based on the relative difference between the objective values of two successive iterations [50] is:

$$|P(\zeta^{k+1}) - P(\zeta^k)| \le \mathbf{tol}\, |P(\zeta^k)|. \tag{4.26}$$

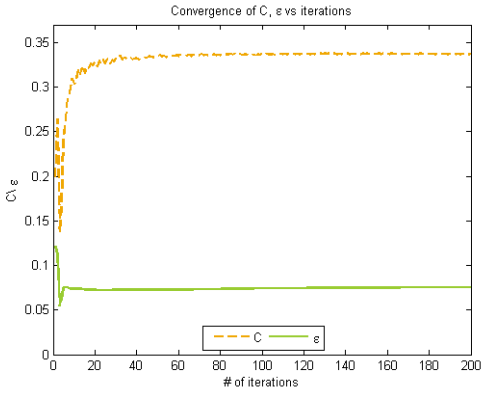
Another loose criterion is based on the error between iterates [68],

$$|\boldsymbol{\zeta}^{k+1} - \boldsymbol{\zeta}^k| \le \mathbf{tol}. \tag{4.27}$$

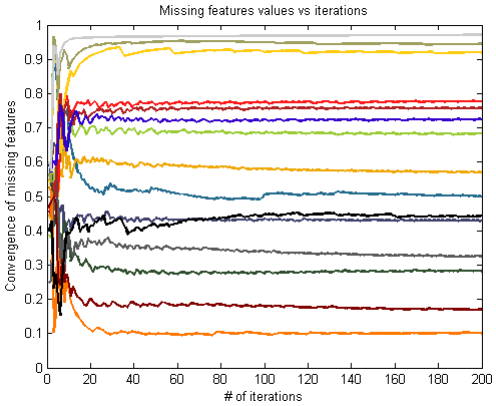
A glance at Figure 4.1 (a)–(c) shows that either of these heuristics can be used to terminate the algorithm as the cross validation objective and values of the outer-level variables C, ε and $\hat{\mathbf{x}}$ stabilize within 100 iterations. It is also interesting to note that the generalization (test) error stabilizes over relatively few iterations. This suggests that, from a machine learning point of view, the termination criteria is effective since reaching the minimizer of $P(\boldsymbol{\zeta})$ too closely is not important. A similar approach was adopted by Keerthi et. al., [50] to terminate their BFGS-based quasi-Newton algorithm to compute SVM



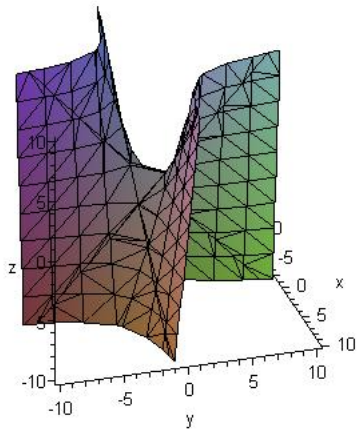
(a) CV error and Generalization Error



(b) Parameters C, ε



(c) Missing features



(d) Bilinear constraint $z = xy$

Figure 4.1: (a)–(c) Convergence of various variables and objectives for bilevel missing value imputation on a 5-d synthetic data set over 200 iterations; (d) Surface plot for $z = xy$

hyper-parameters. In the experiments reported, the criterion (4.27) was used. Successive linearization was implemented in AMPL and the resultant linear programs were solved using CPLEX 9.0. The exact line search, which involves solving a fourth-order polynomial in τ was solved using the non-linear programming solver IPOPT 3.3.3. This approach is called *Successive Linearization AlgorithM for Missing-value Estimation of Data*, SLAMMED.

The methods presented above, inexact imputation and SLAMMED are compared with two commonly used approaches, mean-value imputation and least-squares imputation which are described below.

4.3.5 Other Imputation Approaches

Mean-value imputation [74], as the name suggests, simply involves substituting the missing values with the mean of all the corresponding known features. For instance, if in the data set $\mathcal{M}$, the f-th feature in the vector $\mathbf{x}^i$ is missing i.e., $\hat{x}_f^i$ is unknown, it is replaced thus:

$$\hat{x}_f^i = \frac{1}{|\mathcal{M}| - 1} \sum_{j \in \mathcal{M}, j \neq i} x_f^j. \tag{4.28}$$

Mean substitution was once the most common method of imputation of missing values owing to its simplicity and the fact that the mean will reduce the variance of the variable. A serious problem with this approach is that reduced variance can bias correlation downward or, if the same cases are missing for two variables and means are substituted, correlation can be inflated.

In order to address the problems above, a more preferred approach is least-squares imputation [51]. We can assume, without loss of generality, that x_1^1, the first feature of $\mathbf{x}^1$, is missing. First, the k nearest neighbors to $\mathbf{x}^1$ in $\mathcal{N}$ are identified based on the Pearson correlation between $\mathbf{x}^1$ and the remaining vectors $\mathbf{x}^j$, $j \in \mathcal{N}$ using all but the first feature which $\mathbf{x}_1$ is missing. Let these be $\mathbf{x}^2, \cdots, \mathbf{x}^k$. Define

$$\left[\begin{array}{c|c} \hat{x}_1^1 & \mathbf{w} \\ \hline \mathbf{b} & A \end{array}\right] \equiv \left[\begin{array}{c|ccc} \hat{x}_1^1 & x_2^1 & \dots & x_D^1 \\ \hline x_1^2 & x_2^2 & \dots & x_D^2 \\ \vdots & \vdots & & \vdots \\ x_1^k & x_2^k & \dots & x_D^k \end{array}\right]$$

The idea is to try to express $\mathbf{w}$ as a linear combination of the rows of A by solving the least squares problem

$$\min_{\mathbf{u}} \|A\mathbf{u} - \mathbf{w}\|_2^2, \tag{4.29}$$

so that the missing value $\hat{x}_1^1$ can be computed as

$$\hat{x}_1^1 = \mathbf{b}'\mathbf{u} = \mathbf{b}'(A')^\dagger \mathbf{w}, \tag{4.30}$$

where $(A')^\dagger$ is the pseudo-inverse of A'. The procedure above may be appropriately modified for multiple missing values. It should be noted that this approach may over-correct the estimates by introducing unrealistically low levels of noise in the data. The follow-

ing two-step procedure is employed in order to compare these two methods to the bilevel methods described in the previous sections:

- *Imputation*: Missing values $\hat{\mathbf{x}}$ are imputed using each method
- *Model Selection*: Estimated values are used as though they were the known values. Using this augmented data set, bilevel cross validation (Chapter 3) is performed to compute optimal hyper-parameters C and ε.

Note that unlike the bilevel approaches, both these methods do not use the label information to impute data. Furthermore, they are two-step processes whereas the bilevel formulation is able to solve for $\hat{\mathbf{x}}$, C and ε simultaneously by searching a continuous hyper-parameter space.

4.4 Numerical Experiments

In this section, we compare the performance of the bilevel approaches to mean-value and least-squares imputation on both synthetic and real-world regression data sets. The four methods are compared on three different criteria: cross-validation error, generalization error and imputation error.

4.4.1 Experimental Design

Three different data sets are considered: a synthetic data set, and two real world data sets: IRIS and Automobile MPG. In all cases, data were removed such that, for the purposes of imputation, the data appears to be missing at random (MAR). It is also assumed that a data vector contains at most one missing feature. This is not a necessary restriction however and was only introduced in order to control the experimental setup better.

The synthetic data set contains 120 data points with 5 features and was generated from a uniform distribution on $[0, 1]$. The weight vector, $\mathbf{w} = [1, -1, 1, -1, 1]'$ and the bias, $b = \sqrt{5}$. The noiseless labels were computed as $y = \mathbf{w}'\mathbf{x} - b$ and Gaussian noise drawn from $N(0, 0.2)$ was added to the labels. Fifteen points were chosen randomly and are assumed to contain full feature information; this data subset is called the *baseline set*. The other 105 points have one feature uniformly and randomly removed from each of them such that the resultant data set is MAR. In the experimental results, missing points are added 15 at a time such that the total data set size (with known and missing values) is

30, 45, $\cdots$, 120. A separate hold-out test set of 1000 points was also generated in order to compute the generalization error.

The two real world data sets used were taken from the UCI Online Repository.

- Auto-MPG: The data concerns city-cycle fuel consumption in miles per gallon, to be predicted in terms of 7 attributes. There are a total of 398 instances of which 15 are used to construct the baseline set and 120 more had features removed as in the synthetic case giving a maximum training set size of 135 points. The remaining 263 points make up the test set.

- Iris: The data concerns predicting petal width of an iris flower given three continuous numerical attributes and one multi-valued discrete attribute that can take values of 1, 2 or 3 representing each of the three Iris plant types. There are a total of 150 data points (50 of each plant type). Of these, 15 are used for the baseline set, 105 more to construct the additional training points by with data removed so as to be MAR; the last 30 make up the test set.

In both cases, min-max scaling was applied to the data so that all the features are in $[0, 1]$. In all the cases, for synthetic and real-world data sets, each data set, for each set size i.e., $15, 30, \cdots$ was randomly permuted five times to generate five instances. This was done so that points from the same set are in different training and validation sets during 3-fold cross-validation. The results over these five instances were averaged. This procedure was adopted with a view toward consistency i.e., so that a randomly lucky or unlucky choice of data set would not bias the results.

4.4.2 Computational Results and Discussion

As mentioned before, four methods viz., inexact imputation (4.10) solved using FILTER, SLAMMED, mean-value imputation and least-squares imputation are compared. The three main points of comparison are training error (based on cross validation), test error (based on the hold out test sets) and efficacy of missing value imputation. The last statistic can be computed since the missing values were removed when the data sets were constructed and are known. Under normal circumstances, this information would not be available to the user. For each data set, complete case analysis is also performed, i.e., cross validation and generalization are reported for the case where *all* the training data with missing features are dropped. These results are reported as *baseline*. All the figures may be found at the end of the chapter.

With regard to cross validation error (Figures 4.2a, 4.3a, 4.4a), it is clear that the bilevel approaches perform significantly better than classical imputation approaches. In fact, they perform better than the baseline case and the errors generally decrease as more missing points are added to the training set, mirroring behavior that is generally observed in semi-supervised learning with missing variables.

A similar trend is observed with respect to generalization (Figures 4.2b, 4.3b, 4.4b), though the curves are not as smooth owing to larger variances. The only exception to this trend is the Iris data set (Figure 4.4b), where all approaches train well but generalize poorly with regard to the baseline. However, the generalization of SLAMMED improves steadily toward the baseline as more points are added. A possible reason for this might be because one of the features takes on discrete class values of 1, 2 or 3 making it hard for the continuous-value based approaches to estimate this class information accurately. This opens up avenues to for improving the model so that it can deal with continuous and discrete-valued data.

When it comes to imputing missing values (Figures 4.2c, 4.3c, 4.4c), again the bilevel approaches perform well. The exception here is the Auto-MPG (Figure 4.3c) data set where they impute values worse than the classical approaches, especially mean value deviation. However, better generalization performance suggests that these values might be acceptable if generalization is more important than imputation, which is usually the case. The performance may be explained by the fact that bilevel methods impute values based on the labels and minimizing cross-validation error. Again, this suggests that the model might be improved so that missing-value estimation is based on some combination of cross validation error and imputation error.

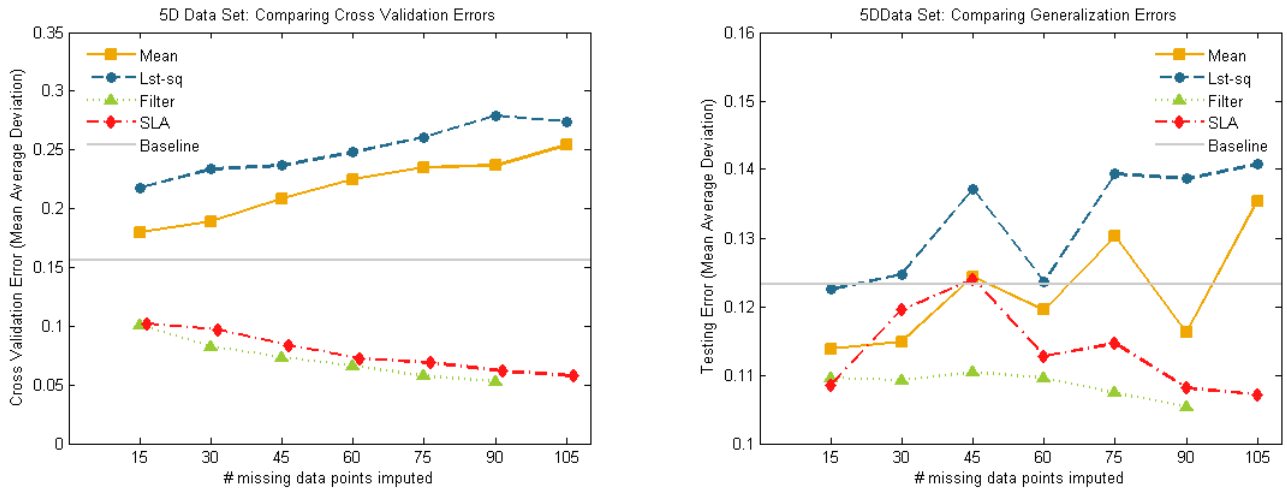
Finally, comparing the two bilevel approaches, the performance of FILTER is usually slightly better than SLAMMED. This is not surprising because SLAMMED does not solve to local minima but only to within a certain closeness to it. Both methods have their limitations. FILTER is unable to solve larger problems as seen in the synthetic data set (Figure 4.2a), while SLAMMED displays very slow convergence especially as problem size grows. An investigation of other algorithmic approaches to problems of this type merits further study.

4.5 Chapter Conclusions

It was demonstrated that bilevel approaches can be applied, not just to parameter and feature selection, but also to problems like missing-value imputation within the cross validation setting. The flexibility of the bilevel approach allows it to handle models and problems of this type by estimating several missing values as outer-level variables. This is in addition to performing simultaneous parameter selection.

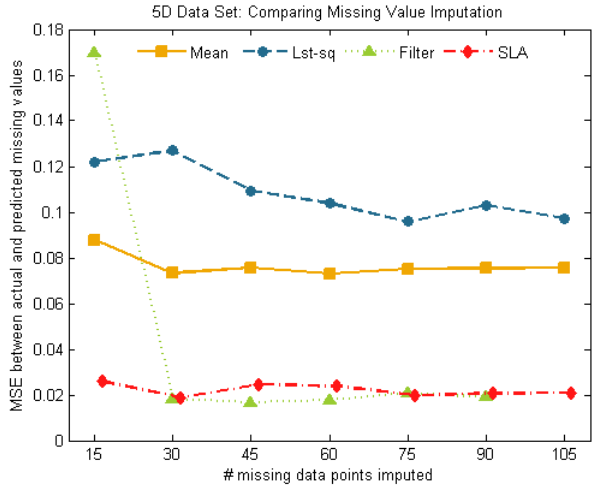
A bilevel approach to missing value imputation was formulated and relaxed in order to apply two algorithmic approaches—a relaxed NLP approach and a successive linearization approach—were formulated. Preliminary empirical results demonstrate that this is a viable approach that outperforms classical approaches and serves as proof-of-concept. The MPEC resulting from the bilevel formulation is far more complex and highly non-convex due to the nonlinear complementarities and bilinear equalities in the constraints. While the algorithmic approaches utilized here were successful, it is apparent that they are limited to solving small problems. The most pressing concern is for more powerful approaches to tackling these bilevel programs.

While support vector regression was chosen as the machine learning problem to demonstrate the potency of the bilevel approach, the methodology can be extended to several machine learning problems including classification and semi-supervised learning. Some of these formulations have been presented in Chapter 5 and [54], while others remain open problems. Aside from discriminative methods, bilevel programming can also be applied to generative methods such as Bayesian techniques. Furthermore, the ability to optimize a large number of parameters allows one to consider new forms of models, loss functions and regularization.



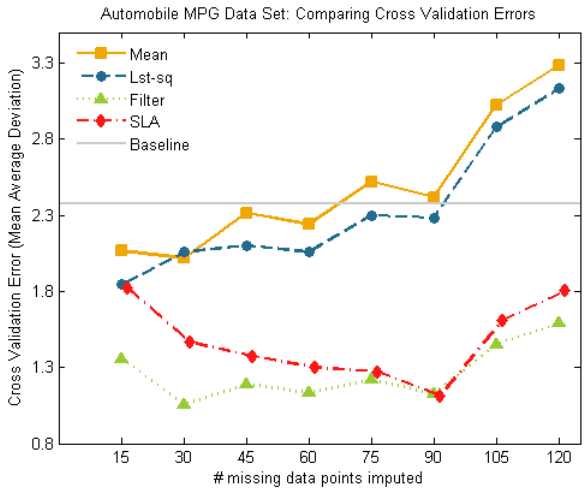
(a) Cross Validation Error

(b) Generalization Error

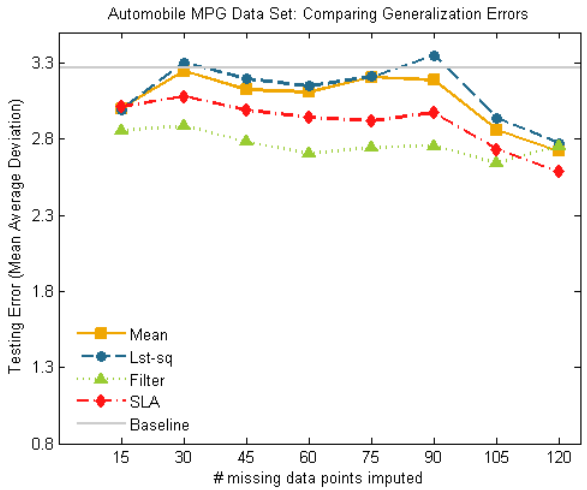


(c) Missing-Value Imputation

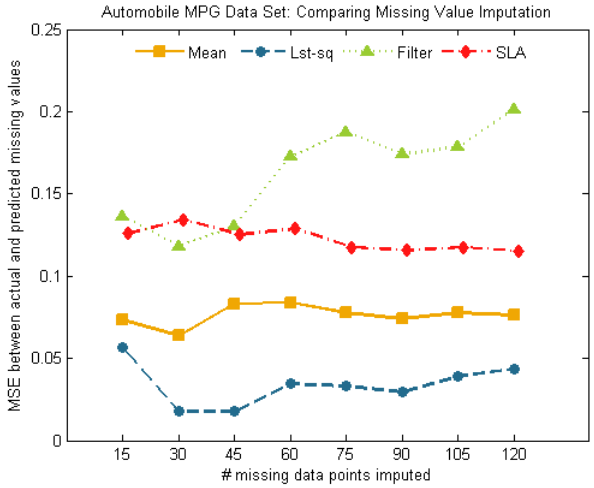
Figure 4.2: 5d synthetic data, as the number of points with missing values increases in the training set.



(a) Cross Validation Error

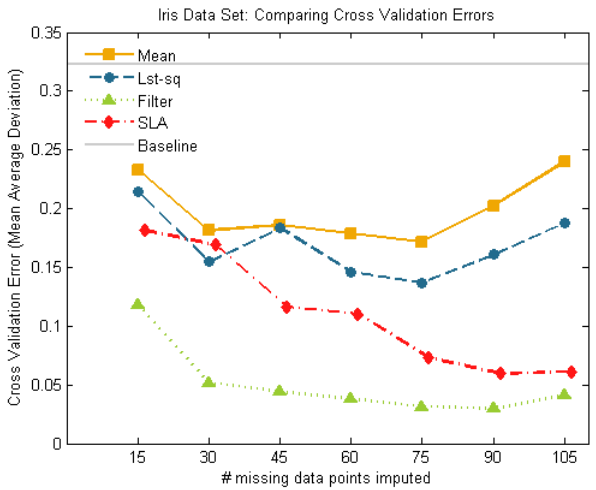


(b) Generalization Error

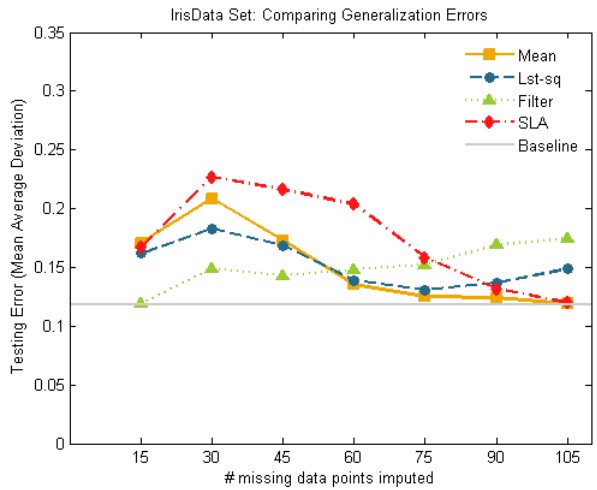


(c) Missing-Value Imputation

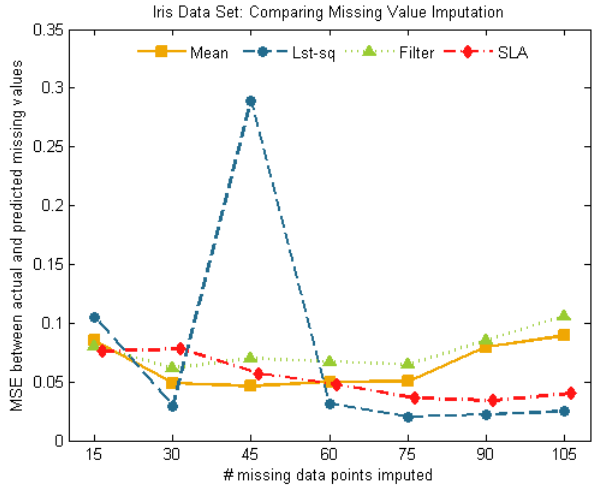
Figure 4.3: Auto-MPG data set, as the number of points with missing values increases in the training set.



(a) Cross Validation Error



(b) Generalization Error



(c) Missing-Value Imputation

Figure 4.4: Iris data set, as the number of points with missing values increases in the training set.

CHAPTER 5
Conclusions and Further Applications

The work presented in this thesis was motivated by the need for methodologies to deal with several open issues in extant support-vector-machine-based learning approaches such as parameter and feature selection. The goal was to develop a paradigm that could provide a unified framework under which these issues could be addressed and applied to a multitude of machine learning problem types. To this end, a novel methodology based on bilevel optimization was proposed and investigated.

Bilevel-based machine learning offers many fundamental advantages over conventional approaches, the most important of which is the ability to systematically treat models with multiple hyper-parameters. In addition, the bilevel approach provides precisely the framework in which problems of various types such as cross validation for support vector classification (Chapter 2) and regression (Chapter 3) in order to perform parameter and feature selection, and learning from missing data (Chapter 4) could be formulated and solved.

The computational challenge of a bilevel optimization problem is its non-convexity, which is due to the complementarity slackness property that is part of the optimality conditions of the lower-level optimization problem; as such, it is not easy to obtain globally optimal solutions. Combining the understanding of the basic theory of the bilevel program that the optimization community has gained over the last twenty years or so with significant advances in the numerical implementation of nonlinear programming solvers allows us to apply the bilevel approach to machine learning effectively. This is most evident in the optimization procedure employed in the bilevel models: replacing inner-level problems with their corresponding first order conditions to obtain a MPEC/LPEC and applying various solution techniques to solve the latter to local optimality.

A bilevel formulation for performing model and feature selection for support vector classification via classification was proposed in Chapter 2 as an alternative to the classical grid search, a coarse and expensive procedure. The flexibility of the bilevel approach facilitates feature selection via the box parameter $\overline{\mathbf{w}}$ and box constraints on the feature vector: $-\overline{\mathbf{w}} \leq \mathbf{w} \leq \overline{\mathbf{w}}$. In addition to the regularization parameter, λ, the model is capable of simultaneously determining feature selection parameters $\overline{\mathbf{w}}$, thus performing complete model selection. The resulting LPECs were relaxed and solved using the publicly available

NLP solver, FILTER (inexact cross validation). Numerical experiments on real-world data sets demonstrate that the bilevel approach generalizes at least as well, if not better than grid search but at greater efficiency.

In Chapter 3, the paradigm was extended to support vector regression where the goal was to determine the hyper-parameters C and ε as well as the feature selection vector $\overline{\mathbf{w}}$. The familiar optimization procedure was employed to derive an LPEC from the bilevel formulation which was solved using FILTER. In addition, an exact penalty formulation was derived and the successive linearization approach was employed to solve it. This gave rise to two algorithms, SLAMS and EZ-SLAMS, the latter being based on the machine learning principle of "early stopping" (here, stopping when complementarity is reached rather than solving to local optimality). Again, the performance of the bilevel approaches was superior to the grid-based approaches. It was also demonstrated that the SLAMS approaches were scalable to large data sets containing hundreds of points.

The work in Chapter 4 showed that the bilevel approach is not restricted to model selection alone. In Chapter 4, a problem that is apposite to many real-world applications, that of learning with missing values was formulated as a bilevel program. Again, the flexibility of the bilevel approach in dealing with multiple parameters allows us to formulate the missing values as outer-level variables and estimate them via cross validation. The resultant MPEC is of far greater complexity than the ones arising from the model selection problems of the previous Chapters. In addition to a relaxed approach (inexact imputation) that was solved using FILTER, a SLA approach was also employed (SLAMMED). Empirical results demonstrate that the bilevel approaches perform far better than classical approaches like mean value imputation and least-squares imputation as serve as an important proof-of-concept. Both approaches have serious limitations: FILTER being unable to solve larger problems and SLAMMED having very slow convergence rates. More research is required to investigate the current algorithmic shortcomings of bilevel imputation and future work in this direction entails devising more powerful algorithms.

We have seen how model selection for various important machine learning problems can be cast as bilevel programs. This is certainly not an exhaustive list of machine learning problems that bilevel programming can be applied to. In concluding this thesis, we look at some more models that provide opportunities for future work and the challenges in implementing them.

5.1 Kernel Bilevel Cross Validation

The models considered thus far have all been linear machines and as such are unable to handle non-linear data sets effectively; this severely limits their usefulness to real data sets. We now demonstrate how one of the most powerful features of SVMs — their ability to deal with high-dimensional, highly nonlinear data using the *kernel trick* — can be incorporated into the bilevel model. We continue this discussion using the bilevel classification example, (2.11), though the results below can easily be generalized to other kernel methods.

5.1.1 Applying the Kernel Trick

The classification model was formulated to perform parameter and feature selection, taking advantage of the ability of the bilevel framework to handle multiple parameters. However, a glance at the first-order conditions, (2.4–2.5), shows that $\mathbf{w}^t$ depends, not only on the training data, but also on the multipliers, $\boldsymbol{\gamma}^{t,\pm}$, of the box constraints. In order to apply the kernel trick and construct RKHS spaces for the kernel methods to operate in, it is essential that the hyperplane, $\mathbf{w}^t$, be expressed solely as a linear combination of the training data. This is a fundamental assumption that is at the heart of all kernel methods through the representer theorem. In order to make this so, we temporarily set aside feature selection, drop the box constraints (effectively causing $\gamma^{t,\pm}$ to drop out of the program) and work with the classical SV classifier (1.10),

$$
\begin{aligned}
\min_{\substack{C,b_t,\mathbf{z}^t \\ \boldsymbol{\zeta}^t,\boldsymbol{\alpha}^t,\boldsymbol{\xi}^t}} \quad & \frac{1}{T}\sum_{t=1}^{T}\frac{1}{|\mathcal{N}_t|}\sum_{i\in\mathcal{N}_t}\zeta_i^t \\
\text{s. t.} \quad & C \geq 0, \\
& \text{and for } t = 1\dots T, \\
& \left.\begin{aligned} & 0 \leq \zeta_i^t \perp y_i\left((\mathbf{x}^i)'\mathbf{w}^t - b_t\right) + z_i^t \geq 0 \\ & 0 \leq z_i^t \perp 1 - \zeta_i^t \geq 0 \end{aligned}\right\} \forall\, i \in \mathcal{N}_t \\
& \left.\begin{aligned} & 0 \leq \alpha_j^t \perp y_j\left((\mathbf{x}^i)'\mathbf{w}^t - b_t\right) - 1 + \xi_j^t \geq 0 \\ & 0 \leq \xi_j^t \perp C - \alpha_j^t \geq 0 \end{aligned}\right\} \forall\, j \in \overline{\mathcal{N}}_t \\
& \sum_{j\in\overline{\mathcal{N}}_t} y_j\alpha_j^t = 0,
\end{aligned}
\tag{5.1}
$$

and also including the constraint,

$$\mathbf{w}^t = \sum_{j \in \overline{\mathcal{N}}_t} y_j \alpha_j^t \mathbf{x}^j, \quad \forall\, t = 1, \ldots, T. \tag{5.2}$$

In order to handle data that is separable only by a nonlinear function, we transform each data point in the input space $\mathbb{R}^n$ to a high dimensional feature space $\mathbb{R}^m$ via $\phi : \mathbb{R}^n \to \mathbb{R}^m$, where the data is now linearly separable. This means that the first order conditions become

$$\mathbf{w}^t = \sum_{j \in \overline{\mathcal{N}}_t} y_j \alpha_j^t \phi(\mathbf{x}^j), \quad \forall\, t = 1, \ldots, T. \tag{5.3}$$

Now, we can eliminate $\mathbf{w}^t$ within each fold of (5.4) using (5.3) and then apply the kernel trick, i.e., the resulting inner-product terms, $\phi(\mathbf{x}^i)'\phi(\mathbf{x}^j)$, are replaced with symmetric, positive semi-definite kernel functions $\kappa(\mathbf{x}^i, \mathbf{x}^j)$. The final bilevel cross-validation model for SV classification *when the kernel is fixed* can be computed if we

$$\begin{array}{ll}
\underset{C, b_t, \mathbf{z}^t, \boldsymbol{\zeta}^t, \boldsymbol{\alpha}^t, \boldsymbol{\xi}^t}{\text{minimize}} & \dfrac{1}{T} \displaystyle\sum_{t=1}^{T} \dfrac{1}{|\mathcal{N}_t|} \sum_{i \in \mathcal{N}_t} \zeta_i^t \\
\text{s. t.} & C \geq 0, \\
& \text{and for } t = 1 \ldots T, \\
& \left.\begin{array}{l}
0 \leq \zeta_i^t \perp y_i \left(\displaystyle\sum_{j \in \overline{\mathcal{N}}_t} y_j \alpha_j^t \kappa(\mathbf{x}^i, \mathbf{x}^j) - b_t \right) + z_i^t \geq 0 \\
0 \leq z_i^t \perp 1 - \zeta_i^t \geq 0
\end{array}\right\} \forall\, i \in \mathcal{N}_t \\
& \left.\begin{array}{l}
0 \leq \alpha_i^t \perp y_i \left(\displaystyle\sum_{j \in \overline{\mathcal{N}}_t} y_j \alpha_j^t \kappa(\mathbf{x}^i, \mathbf{x}^j) - b_t \right) - 1 + \xi_i^t \geq 0 \\
0 \leq \xi_i^t \perp C - \alpha_i^t \geq 0
\end{array}\right\} \forall\, i \in \overline{\mathcal{N}}_t \\
& \displaystyle\sum_{i \in \overline{\mathcal{N}}_t} y_i \alpha_i^t = 0.
\end{array} \tag{5.4}$$

While it may not appear so at first glance, the optimization problem above is still an instance of an LPEC. Unfortunately, it is usually unreasonable to expect ready-made kernels for most machine learning tasks; in fact, most kernel families are parameterized, and the kernel parameters are typically determined via cross validation. Also, unlike its

linear counterpart, this model is not capable of performing feature selection.

5.1.2 Designing Unknown Kernels

The issues of parameter selection (for regularization and the kernel) and feature selection can be combined as in the linear model if we use a parameterized kernel of the form $\kappa(\mathbf{x}^i, \mathbf{x}^j; \mathbf{p}, \mathbf{q})$. The nonnegative vector, $\mathbf{p} \in \mathbb{R}^n_+$, is the feature selection or scaling vector, and $\mathbf{q} \geq 0$ is a vector of kernel parameters. Let $P = \mathbf{diag}(\mathbf{p})$. The parameterized versions of some commonly used kernels are shown below.

$$\begin{aligned} \text{Linear kernel} \quad & \kappa(\mathbf{x}^i, \mathbf{x}^j; \mathbf{p}) = \mathbf{x}^{i\prime} P \, \mathbf{x}^j \\ \text{Polynomial kernel} \quad & \kappa(\mathbf{x}^i, \mathbf{x}^j; \mathbf{p}, c, d) = (\mathbf{x}^{i\prime} P \, \mathbf{x}^j + c)^d \\ \text{Gaussian kernel} \quad & \kappa(\mathbf{x}^i, \mathbf{x}^j; \mathbf{p}) = \exp\left((\mathbf{x}^i - \mathbf{x}^j)' P \, (\mathbf{x}^i - \mathbf{x}^j)\right) \end{aligned} \tag{5.5}$$

Other kernels can be similarly extended and used in the model. Consequently, the new kernel parameters, $\mathbf{p}$ and $\mathbf{q}$, enter the outer level of the kernel model as variables in the problem. The introduction of the parameterized kernel is a very powerful extension to the linear model (5.1) as it is capable of determining the kernel parameters (*model design for unknown kernels*, DUNK) and also picking the regularization parameters and features leaving only the choice of kernel family to the user. The optimization problem (5.4) is an MPEC with non-linear complementarity constraints and in general is a very difficult problem to solve.

5.1.3 Solving the Kernel MPEC

We can employ the same strategy used in SLAMS to solve the DUNK problem, namely to lift the penalizations of the nonlinear constraints into the objective and then apply a Frank-Wolfe approach. To isolate the nonlinearities we add variables that represent the product of $\boldsymbol{\alpha}^t$ trained with each fold with each row of the kernel matrix $\mathbf{K}_t$ corresponding to the training data within each fold. For the *training* data in the t-th fold:

$$K^{trn}_{t,i} = \sum_{j \in \overline{\mathcal{N}}_t} y_j \alpha^t_j \kappa(\mathbf{x}^j, \mathbf{x}^i; \mathbf{p}, \mathbf{q}), \quad \forall \, i \in \overline{\mathcal{N}}_t, \tag{5.6}$$

and for the *validation* data

$$K_{t,i}^{val} = \sum_{j \in \overline{\mathcal{N}}_t} y_j \alpha_j^t \kappa(\mathbf{x}^j, \mathbf{x}^i; \mathbf{p}, \mathbf{q}), \quad \forall\, i \in \mathcal{N}_t. \tag{5.7}$$

Notice that the summation in (5.7) is still over the training points as it is only possible to compute $\boldsymbol{\alpha}$s for these points within each fold. With this substitution, the constraint region for the MPEC (5.4) becomes a polyhedral set represented by the linear complementarity system as shown below:

$$\begin{aligned}
&\text{for } t = 1 \ldots T, \\
&\left.\begin{aligned} &0 \le \zeta_i^t \perp y_i(K_{t,i}^{val} - b_t) + z_i^t \ge 0 \\ &0 \le z_i^t \perp 1 - \zeta_i^t \ge 0 \end{aligned}\right\} \forall\, i \in \mathcal{N}_t \\
&\left.\begin{aligned} &0 \le \alpha_i^t \perp y_i(K_{t,i}^{trn} - b_t) - 1 + \xi_i^t \ge 0 \\ &0 \le \xi_i^t \perp C - \alpha_i^t \ge 0 \end{aligned}\right\} \forall\, i \in \overline{\mathcal{N}}_t \\
&\sum_{i \in \overline{\mathcal{N}}_t} y_i \alpha_i^t = 0.
\end{aligned} \tag{5.8}$$

We denote the constraints above as S_0. The constraint set without the complementarities is denoted S_{LP}. Now, we introduce penalty functions for the nonlinear equality constraints arising from the transformations (5.6) and (5.7):

$$Q_t = \sum_{i \in \mathcal{N}_t} \left\| \mathbf{K}_t^{val} - \sum_{j \in \overline{\mathcal{N}}_t} y_j \alpha_j^t \kappa(\mathbf{x}^j, \mathbf{x}^i; \mathbf{p}, \mathbf{q}) \right\|_2^2 + \sum_{i \in \overline{\mathcal{N}}_t} \left\| \mathbf{K}_t^{trn} - \sum_{j \in \overline{\mathcal{N}}_t} y_j \alpha_j^t \kappa(\mathbf{x}^j, \mathbf{x}^i; \mathbf{p}, \mathbf{q}) \right\|_2^2, \tag{5.9}$$

Alternately, the ℓ_1-norm penalty can also be used. In addition, we can define exact penalty functions for the complementarities in order to lift them into the objective as well. As before, we also have the quadratic penalty,

$$\begin{aligned}
P_t \;&= \sum_{i \in \mathcal{N}_t} \zeta_i^t y_i (K_{t,i}^{val} - b_t) + \sum_{i \in \mathcal{N}_t} z_i^t \\
&\quad + \sum_{i \in \overline{\mathcal{N}}_t} \alpha_i^t y_i (K_{t,i}^{trn} - b_t) + C \sum_{i \in \overline{\mathcal{N}}_t} \xi_i^t - \sum_{i \in \overline{\mathcal{N}}_t} \alpha_i^t.
\end{aligned} \tag{5.10}$$

Thus, the overall non-linear penalty problem is given below and can be solved using successive linearization.

$$\begin{aligned} \underset{C,\mathbf{p},\mathbf{q},b_t,\mathbf{z}^t,\boldsymbol{\zeta}^t,\boldsymbol{\alpha}^t,\boldsymbol{\xi}^t}{\text{minimize}} \quad & \frac{1}{T}\sum_{t=1}^{T}\frac{1}{|\mathcal{N}_t|}\sum_{i\in\mathcal{N}_t}\zeta_i^t + \mu\sum_{t=1}^{T}P_t + \lambda\sum_{t=1}^{T}Q_t \\ \text{s. t.} \quad & (C,\, \mathbf{p},\, \mathbf{q},\, b_t,\, \mathbf{z}^t,\, \boldsymbol{\zeta}^t,\, \boldsymbol{\alpha}^t,\, \boldsymbol{\xi}^t) \in S_{\mathsf{LP}}. \end{aligned} \tag{5.11}$$

5.2 Semi-supervised Learning

We have, thus far, focussed on model selection for *supervised* learning tasks such as classification and regression, with the label information available for all training points. Frequently, however, in applications like text classification, drug design, medical diagnosis, and graph and network search, the training set consists of a large number of unlabeled data points and a relatively small number of labeled training points. This necessitates *semi-supervised* learning, where training is performed using both the labeled and unlabeled data. If all the training data is unlabeled, the problem becomes one of *unsupervised* learning, e.g., clustering.

The concept of semi-supervised learning is closely related to that of *transductive learning*, which can be contrasted with the more typically performed *inductive learning*. In induction, the given labeled data is used to construct a robust decision rule that is valid everywhere. This rule is fixed after training and can subsequently be applied to the future test data. In transduction, the labeled training data and the unlabeled test data are both given. All available data is used to construct the decision rule in order to avoid overfitting. The learning task here is to not only predict labels for the test data available but for all future data as well. Performing transductive learning may result in improvement in generalization error bounds [87], thus reducing the number of labeled data required for good generalization. This is a very important learning task as there exist many applications where labeled data are expensive to generate whereas unlabeled data are abundant (e.g. drug design, medical diagnosis, web search).

Some additional notation is now introduced. As before, $\Omega = \{\mathbf{x}_i, y_i\}_{i=1}^{\ell}$ represents the set of labeled data, with $\ell = |\Omega|$. Let $\Psi = \{\mathbf{x}_i\}_{i=1}^{u}$ represent the unlabeled training data, with the corresponding labels (to be determined) being z_i, and $u = |\Psi|$. The sets, Ω and Ψ, are indexed by $\mathcal{N}$ and $\mathcal{M}$ respectively.

5.2.1 Semi-supervised Regression

In bilevel semi-supervised regression, the labels of the unlabeled training data are treated as control variables, $\mathbf{z}$. The general bilevel model for semi-supervised machine learning problems can be formulated as

$$\begin{aligned} \underset{f,\,\mathbf{z},\,\boldsymbol{\lambda}}{\text{minimize}} \quad & \Theta(f,\,\mathbf{z};\,\Omega,\,\Psi,\,\boldsymbol{\lambda}) \\ \text{subject to} \quad & \boldsymbol{\lambda} \in \Lambda, \\ & f \in \underset{f\in\mathcal{F}}{\arg\min} \left\{ \mathcal{P}(f,\,\boldsymbol{\lambda}) + \sum_{j\in\mathcal{N}} \mathcal{L}_l\left(y_j,\,f(\mathbf{x}_j),\,\boldsymbol{\lambda}\right) + \sum_{j\in\mathcal{M}} \mathcal{L}_u\left(z_j,\,f(\mathbf{x}_j),\,\boldsymbol{\lambda}\right) \right\}. \end{aligned} \tag{5.12}$$

In the model above, the loss functions, $\mathcal{L}_l$ and $\mathcal{L}_u$, are applied to the labeled and unlabeled data respectively, while $\mathcal{P}$ performs regularization. All the appropriate parameters, $\boldsymbol{\lambda}$, are optimized in the outer level; these parameters can include the regularization constant, tube width (for regression) and feature selection vectors among others. Optimizing the unknown labels, $\mathbf{z}$ in the outer level corresponds to inductive learning, while optimizing them in the inner level corresponds to transductive learning. An interesting variant that combines both types of learning occurs if $\mathbf{z}$ is optimized in both levels.

For semi-supervised support vector regression, we can choose both loss functions to be ε-insensitive and ℓ_2-norm regularization. For the case of one labeled training set, one unlabeled training set, and one test set, this yields the following bilevel program:

$$\begin{aligned} \underset{C,D,\varepsilon,\mathbf{w},b,\mathbf{z}}{\text{minimize}} \quad & \sum_{i\in\mathcal{N}} |\mathbf{x}_i'\mathbf{w} - b - y_i| \\ \text{subject to} \quad & \varepsilon, C, D \geq 0, \\ & (\mathbf{w}, b) \in \underset{(\mathbf{w},b)\in\mathbb{R}^{n+1}}{\arg\min} \left\{ \begin{array}{r} \frac{1}{2}\|\,\mathbf{w}\,\|_2^2 + \frac{C}{|\mathcal{N}|} \sum_{j\in\mathcal{N}} \max\left(|\mathbf{x}_j'\mathbf{w} - b - y_j| - \varepsilon, 0\right) \\ + \frac{D}{|\mathcal{M}|} \sum_{j\in\mathcal{M}} \max\left(|\mathbf{x}_j'\mathbf{w} - b - z_j| - \varepsilon, 0\right) \end{array} \right\}. \end{aligned} \tag{5.13}$$

The outer-level objective is simply the mean average deviation (MAD) on all the labeled data. The inner-level objective uses both the labeled and unlabeled data sets making this an instance of transductive learning. The labels, $\mathbf{z}$, are used in the inner-level loss function but are optimized as outer-level variables along with the hyper-parameters ε, C, and D. Additional upper and lower bounds can be imposed on these parameters if desired. This program can be converted to an LPEC as before. It should be noted that in typical semi-

supervised learning problems, the number of unlabeled examples, u is far greater than the number of labeled examples, ℓ. This means that (5.13) will have a large number of outer-level variables ($\mathbf{z}$) and complementarity constraints arising from the unlabeled data points.

The model (5.13) performs simultaneous transductive learning and parameter selection. The quality of the "optimal" parameters can potentially be improved by combining semi-supervised learning with T-fold cross validation. This can be achieved if we

$$\begin{array}{ll} \underset{C,D,\varepsilon,\mathbf{w}^t,b_t,\mathbf{z}}{\text{minimize}} & \dfrac{1}{T}\displaystyle\sum_{t=1}^{T}\dfrac{1}{|\mathcal{N}_t|}\sum_{i\in\mathcal{N}_t}|\mathbf{x}_i'\mathbf{w}-b-y_i| \\ \text{subject to} & \varepsilon, C, D \geq 0, \\ & \text{and for } t=1,\ldots,T, \\ & (\mathbf{w}^t,b_t)\in \underset{(\mathbf{w},b)\in\mathbb{R}^{n+1}}{\arg\min}\left\{\begin{array}{r} \dfrac{1}{2}\|\,\mathbf{w}\,\|_2^2+\dfrac{C}{|\mathcal{N}_t|}\displaystyle\sum_{j\in\mathcal{N}_t}\max\left(|\mathbf{x}_j'\mathbf{w}-b-y_j|-\varepsilon,0\right) \\ +\dfrac{D}{|\mathcal{M}|}\displaystyle\sum_{j\in\mathcal{M}}\max\left(|\mathbf{x}_j'\mathbf{w}-b-z_j|-\varepsilon,0\right) \end{array}\right\}, \end{array} \tag{5.14}$$

so that the resultant program is again a novel combination of inductive and transductive learning. Here, the unlabeled data is used to train the decision rule *for each fold*. As there are T inner level problems, the complementarity conditions containing the unlabeled data will occur T times, though each time with a different $(\mathbf{w}^t, b_t)$ in the constraints.

5.2.2 Semi-supervised Classification

Turning our attention to classification problems, we encounter several choices for both the inner- and outer-level loss functions. As always, we use the hinge loss for the labeled points. We look at three loss functions that were introduced in [21] for the unlabeled points in the inner level. The first is the so-called *hard-margin* loss,

$$\mathcal{L}_u(\mathbf{w},b)=\left\{\begin{array}{ll} \infty, & \text{for } -1<\mathbf{x}'\mathbf{w}-b<1, \\ 0, & \text{otherwise.} \end{array}\right. \tag{5.15}$$

This can be introduced into the inner level through the very hard constraint $\max(1 - |\mathbf{x}'\mathbf{w} - b|, 0) = 0$, resulting in the following inner-level optimization problem:

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi},\mathbf{z}^+,\mathbf{z}^-} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{j\in\mathcal{N}} \xi_j^t \\ \text{s. t.} \quad & y_i(\mathbf{x}_i'\mathbf{w} - b) \geq 1 - \xi_i, \;\; \xi_i \geq 0, \qquad \forall\, i \in \mathcal{N} \\ & \left.\begin{aligned} -(\mathbf{x}_j'\mathbf{w} - b) &\geq 1 - z_j^+, \quad z_j^+ \geq 0, \\ (\mathbf{x}_j'\mathbf{w} - b) &\geq 1 - z_j^-, \quad z_j^- \geq 0, \\ z_j^+ \, z_j^- &= 0 \end{aligned}\right\} \quad \forall\, j \in \mathcal{M}. \end{aligned} \tag{5.16}$$

This results in a non-convex, quadratically-constrained quadratic program (QCQP) which is hard to solve in general. Furthermore, the hard-margin condition might be too strong to allow for feasible solutions, leading us to consider soft-margin variants: the quadratic-margin penalty,

$$\mathcal{L}_u(\mathbf{w}, b) = \max(1 - (\mathbf{x}'\mathbf{w} - b)^2, 0), \tag{5.17}$$

and the non-convex hat-loss function,

$$\mathcal{L}_u(\mathbf{w}, b) = \max(1 - |\mathbf{x}'\mathbf{w} - b|, 0). \tag{5.18}$$

These loss functions arise from the relaxing the hard constraint $z_j^+ \, z_j^- = 0$ in (5.16) by moving it into the inner-level objective; if the product, $z_j^+ \, z_j^-$, is used directly, a quadratic penalty function, (5.17), results, and if the minimum error, $\min(z_j^+, z_j^-)$ is used, the hat loss function results. Using the quadratic penalty function for the unlabeled data is precisely the transductive idea proposed by Vapnik [87]. The "optimal" labels on the unlabeled data can be calculated as $\mathsf{sign}(z_j^+ \; - z_j^-)$.

Finally, we can use the step function to formulate loss functions that use the number

of misclassifications for both the labeled and unlabeled data sets if we

$$
\begin{aligned}
\underset{C,D,\mathbf{w},b,\boldsymbol{\zeta},\mathbf{z}}{\text{minimize}} \quad & \frac{1}{|\mathcal{N}|}\sum_{i\in\mathcal{N}}\zeta_i \\
\text{subject to} \quad & C, D \geq 0, \\
& \boldsymbol{\zeta} \in \underset{0\leq\boldsymbol{\zeta}\leq\mathbb{1}}{\arg\min} \left\{ \sum_{i\in\mathcal{N}} \zeta_i y_i \left(\mathbf{x}_i'\mathbf{w} - b\right) \right\} \\
& \mathbf{z} \in \underset{0\leq\mathbf{z}\leq\mathbb{1}}{\arg\min} \left\{ \sum_{i\in\mathcal{M}} -\mathbf{z}_i \left(\mathbf{x}_i'\mathbf{w} - b\right) \right\} \\
& (\mathbf{w}, b) \in \underset{(\mathbf{w},b)\in\mathbb{R}^{n+1}}{\arg\min} \left\{ \begin{array}{r} \frac{1}{2}\|\,\mathbf{w}\,\|_2^2 + \frac{C}{|\mathcal{N}|}\sum_{j\in\mathcal{N}} \max\left(1 - y_j(\mathbf{x}_j'\mathbf{w} - b), 0\right) \\ + \frac{D}{|\mathcal{M}|}\sum_{j\in\mathcal{M}} \max\left(1 - z_j(\mathbf{x}_j'\mathbf{w} - b), 0\right) \end{array} \right\}.
\end{aligned}
\tag{5.19}
$$

The outer-level objective performs misclassification minimization on the labeled data, with the first inner-level problem counting the number of misclassifications. The second inner-level problem computes the labels on the unlabeled data which are used to perform learning in the third inner-level problem. As in the regression case, the problem (5.19) and its variants that use the various loss functions above can be combined with cross validation to perform more effective parameter selection. Feature selection can also be incorporated into these models by adding extra constraints on $\mathbf{w}$ or by changing the regularization as discussed in the previous sections. It is also relatively straightforward to kernelize the models discussed above as per the discussion in Section 5.1.2, as long as care is taken in dealing with the labeled and unlabeled kernels.

5.3 Incorporating Multitask Learning

We return to the problem of cross validation to demonstrate that *multitask learning* concepts can easily be incorporated in the bilevel setting. Multitask learning [13] is defined as learning multiple related tasks simultaneously. This type of learning is an instance of *inductive transfer*, otherwise called *transfer learning*, where the knowledge learned from some tasks may be applied to learning a related task more efficiently.

In the T-fold bilevel cross validation setting, each of the T inner-level problems attempts to construct a decision rule on subsets of the same training sample, which, by statistical learning theory assumptions, are drawn i.i.d. from the same distribution. Thus, the tasks of training within each fold are related and amenable to incorporating multitask

principles. We do this by introducing new variables, $(\mathbf{w}^0, b_0)$, into the inner-level problems. For example, consider the following SV regression inner level, (5.20) with added multi-task terms (and including the bias term):

$$(\mathbf{w}^t, b_t) \in \underset{(\mathbf{w},b)\in\mathbb{R}^{n+1}}{\arg\min} \left\{ \begin{array}{c} \frac{1}{2}\| \mathbf{w} \|_2^2 + \frac{C}{|\overline{\mathcal{N}}_t|} \sum\limits_{j\in\overline{\mathcal{N}}_t} \max\left(|\mathbf{x}_j'\mathbf{w} - y_j| - \varepsilon, 0\right) \\ + \frac{\lambda_{\mathbf{w}}}{2}\|\mathbf{w} - \mathbf{w}^0\|_2^2 + \frac{\lambda_b}{2}(b - b_0)^2 \end{array} \right\}. \tag{5.20}$$

The variables $(\mathbf{w}^0, b_0)$ enter the bilevel model as outer-level variables as do the parameters $\lambda_{\mathbf{w}}$ and λ_b. The multitask terms provide variance control by making each of the individual hyperplanes less susceptible to variations within their respective training sets. They also provide additional regularization. Finally, they ensure fold consistency because of the enforced task relatedness. We can replace (5.20) with its corresponding KKT conditions:

$$\begin{array}{l} 0 \;=\; (\,1 + \lambda_{\mathbf{w}}\,)\mathbf{w}^t - \lambda_w\,\mathbf{w}^0 + \sum\limits_{i\in\overline{\mathcal{N}}_t} (\,\alpha_i^{t,+} - \alpha_i^{t,-}\,)\,\mathbf{x}_i, \\ 0 \;=\; \lambda_b(b_t - b_0) \;+\; \sum\limits_{i\in\overline{\mathcal{N}}_t} (\,\alpha_i^{t,+} - \alpha_i^{t,-}\,), \\ \left.\begin{array}{lll} 0 \;\le\; \xi_i^t & \perp & \dfrac{C}{|\overline{\mathcal{N}}_t|} - \alpha_i^{t,+} - \alpha_i^{t,-} \;\ge\; 0, \\ 0 \;\le\; \alpha_i^{t,+} & \perp & \xi_i^t + \varepsilon - \mathbf{x}_i'\mathbf{w}^t + b_t + y_i \;\ge\; 0, \\ 0 \;\le\; \alpha_i^{t,-} & \perp & \xi_i^t + \varepsilon + \mathbf{x}_i'\mathbf{w}^t - b_t - y_i \;\ge\; 0, \end{array}\right\} \quad \forall\, i \;\in \overline{\mathcal{N}}_t. \end{array} \tag{5.21}$$

From (5.21), we deduce

$$\begin{array}{l} \mathbf{w}^t \;=\; \dfrac{1}{1 + \lambda_{\mathbf{w}}} \left[\lambda_{\mathbf{w}}\,\mathbf{w}^0 - \sum\limits_{i\in\overline{\mathcal{N}}_t} (\,\alpha_i^{t,+} - \alpha_i^{t,-}\,)\,\mathbf{x}^i \right], \\ b_t \;=\; b_0 - \dfrac{1}{\lambda_b} \sum\limits_{i\in\overline{\mathcal{N}}_t} (\alpha_i^{t,+} - \alpha_i^{t,-}), \end{array} \tag{5.22}$$

where it is understood that if $\lambda_b = 0$, then the latter expression for b_t reduces to

$$\sum_{i\in\overline{\mathcal{N}}_t} (\alpha_i^{t,+} - \alpha_i^{t,-}) = 0, \tag{5.23}$$

which does not involve b_t. In the interest of kernelizing (5.21), we postulate that

$$\mathbf{w}^0 \equiv \sum_{j \in \mathcal{N}} \beta_j \, \mathbf{x}^j, \tag{5.24}$$

for some scalars, β_j, to be determined. We obtain

$$\mathbf{w}^t \equiv \frac{1}{1+\lambda_{\mathbf{w}}} \left[\lambda_{\mathbf{w}} \sum_{j \in \mathcal{N}} \beta_j \, \mathbf{x}^j - \sum_{j \in \overline{\mathcal{N}}_t} (\, \alpha_j^{t,+} - \alpha_j^{t,-} \,) \, \mathbf{x}^j \right]. \tag{5.25}$$

This last expression can be substituted into the complementarities in (5.21) to give

$$\begin{aligned}
&0 \leq \xi_i^t \quad \perp \; \frac{C}{|\overline{\mathcal{N}}_t|} - \alpha_i^{t,+} - \alpha_i^{t,-} \; \geq \; 0, \\
&0 \leq \alpha_i^{t,+} \perp \; \xi_i^t + \varepsilon - \frac{1}{1+\lambda_{\mathbf{w}}} \left[\lambda_{\mathbf{w}} \sum_{j \in \mathcal{N}} \beta_j \mathbf{x}_i' \mathbf{x}_j - \sum_{j \in \overline{\mathcal{N}}_t} (\alpha_j^{t,+} - \alpha_j^{t,-}) \mathbf{x}_i' \mathbf{x}_j \right] + b_t + y_i \geq 0, \\
&0 \leq \alpha_i^{t,-} \perp \; \xi_i^t + \varepsilon + \frac{1}{1+\lambda_{\mathbf{w}}} \left[\lambda_{\mathbf{w}} \sum_{j \in \mathcal{N}} \beta_j \mathbf{x}_i' \mathbf{x}_j - \sum_{j \in \overline{\mathcal{N}}_t} (\alpha_j^{t,+} - \alpha_j^{t,-}) \mathbf{x}_i' \mathbf{x}_j \right] - b_t - y_i \geq 0.
\end{aligned} \tag{5.26}$$

The "kernel trick" can now be applied to (5.26); see Section 5.1.2 for details.

All the models that have been implemented in this thesis have been discriminative i.e., they attempt to learn a direct map from the data x to the labels, y, or model the posterior probability $p(y|\mathbf{x})$ directly. This is in contrast to generative methods which try to learn and maximize the probability $p(y|\mathbf{x})$ via the Bayes Rule so that the results are used to approximate the behavior of the learner to the joint probability $p(\mathbf{x}, y)$. Noting that all the methods proposed here were nonparametric methods, an interesting avenue of further research with regard to modeling is the incorporation of parametric or generative methods based on probability models into the bilevel framework. Preliminary work in this direction by Epshteyn and DeJong [27] indicates that bilevel approaches can be effective in the generative setting as well; this, however, is out of scope for this thesis, where the focus is on discriminative approaches.

The flexibility of the bilevel approach is such that a seemingly endless number of machine learning formulations can be cast into this framework. Some of these models were investigated in this thesis; incorporation of other models (for instance, sub-sampling methods other than cross validation, generative models and so on) into this framework is left as a viable area of research to the machine learning community. Two algorithms

were implemented to solve the LPECs/MPECs arising from the bilevel machine learning programs and were successful for moderately-sized data sets. The need of the hour is scalability: for algorithms that can handle hundreds of thousands of data points. Development of such scalable algorithms for bilevel machine learning remains a significant open challenge to the optimization community.

REFERENCES

[1] M. A. Aizerman, E. M. Braverman, and L. I. Rozonoer. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.

[2] Faiz Al-Khayyal. Jointly constrained bilinear programs and related problems: an overview. *Computers and Mathematics with Applications*, 19(11):53–62, 1990.

[3] Mihai Anitescu. On solving mathematical programs with complementarity constraints as nonlinear programs. *SIAM Journal on Optimization*, 15:1203–1236, 2005.

[4] Mihai Anitescu, Paul Tseng, and Stephen J. Wright. Elastic-mode algorithms for mathematical programs with equilibrium constraints: Global convergence and stationarity properties. *Mathematical Programming (in print)*, 2005.

[5] Charles Audet, Pierre Hansen, Brigitte Jaumard, and Gilles Savard. Links between linear bilevel and mixed 0-1 programming problems. *Journal of Optimization Theory and Applications*, 93(2):273–300, 1997.

[6] Kristin P. Bennett and Ayhan Demiriz. Semi-supervised support vector machines. In *Proceedings of the 1998 conference on Advances in neural information processing systems II*, pages 368–374, Cambridge, MA, USA, 1999. MIT Press.

[7] Kristin P. Bennett, Jing Hu, Xiaoyun Ji, Gautam Kunapuli, and Jong-Shi Pang. Model selection via bilevel optimization. *International Joint Conference on Neural Networks, (IJCNN) '06.*, pages 1922–1929, 2006.

[8] Kristin P. Bennett and Olvi L. Mangasarian. Bilinear separation of two sets in n-space. *Computational Optimization and Applications*, 2(3):207–227, 1993.

[9] Jinbo Bi, Kristin P. Bennett, Mark Embrechts, Curt Breneman, and Minghu Song. Dimensionality reduction via sparse support vector machines. *Journal of Machine Learning Research*, 3:1229–1243, 2003.

[10] Asa Ben-Hur Biowulf, David Horn, Hava T. Siegelmann, and Vladimir Vapnik. Support vector clustering. *Journal of Machine Learning Research*, 2:125–137, 2001.

[11] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *COLT '92: Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, New York, NY, USA, 1992. ACM Press.

[12] Jerome Bracken and James T. McGill. Mathematical programs with optimization problems in the constraints. *Operations Research*, 21(1):37–44, 1973.

[13] Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

[14] Olivier Chapelle, Vladimir Vapnik, Olivier Bousquet, and Sayan Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1–3):131–159, 2002.

[15] Chunhui Chen and Olvi L. Mangasarian. A class of smoothing functions for nonlinear and mixed complementarity problems. *Computational Optimization and Applications*, 5:97–138, 1996.

[16] Pai-Hsuen Chen, Chih-Jen Lin, and Bernhard Schölkopf. A tutorial on nu-support vector machines. *Applied Stochastic Models in Business and Industry*, 21(2):111–136, 2005.

[17] Xiaojun Chen and Masao Fukushima. A smoothing method for a mathematical program with p-matrix linear complementarity constraints. *Computational Optimization and Applications*, 27(3):223–246, 2004.

[18] S.-J. Chung. Np-completeness of the linear complementarity problem. *Journal of Optimization Theory and Applications*, 60(3):393–400, 1989.

[19] Corrina Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[20] Joseph Czyzyk, Michael P. Mesnier, and Jorge J. Moré. The NEOS server. *IEEE Computational Science and Engineering*, 5(3):68–75, 1998.

[21] Ayhan Demiriz, Kristin P. Bennett, Curt M. Breneman, and Mark Embrechts. Support vector regression methods in cheminformatics. *Computer Science and Statistics*, 33, 2001.

[22] Stephan Dempe. *Foundations of Bilevel Programming*. Kluwer Academic Publishers, Dordrecht, 2002.

[23] Stephan Dempe. Annotated bibliography on bilevel programming and mathematical programs with equilibrium constraints. *Optimization*, 52:333–359, 2003.

[24] Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B (Methodological)*, 39(1):1–38, 1977.

[25] Harris Drucker, Chris J. C. Burges, Linda Kaufman, Alex Smola, and Vladimir Vapnik. Support vector regression machines. In Michael C. Mozer, Michael I. Jordan, and Thomas Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, page 155. The MIT Press, 1997.

[26] Kaibo Duan, Sathiya S. Keerthi, and Aun N. Poo. Evaluation of simple performance measures for tuning SVM hyperparameters. *Neurocomputing*, 51:41–59, 2003.

[27] Arkady Epshteyn and Gerald DeJong. Generative prior knowledge for discriminative classification. *Journal of Machine Learning Research*, 27:25–53, 2006.

[28] Theodoros Evgeniou and Massimiliano Pontil. Regularized multi–task learning. In *KDD '04: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 109–117, New York, NY, USA, 2004. ACM Press.

[29] Michael C. Ferris and Todd S. Munson. Interior-point methods for massive support vector machines. *SIAM Journal on Optimization*, 13(3):783–804, 2002.

[30] Andreas Fischer. A special newton-type optimization method. *Optimization*, 24:269–282, 1992.

[31] Roger Fletcher and Sven Leyffer. User manual for filtersqp. Technical Report NA/181, Department of Mathematics, University of Dundee, 1999.

[32] Roger Fletcher and Sven Leyffer. Nonlinear programming without a penalty function. *Mathematical Programming*, 91:239–269, 2002.

[33] Roger Fletcher and Sven Leyffer. Solving mathematical program with complementarity constraints as nonlinear programs. *Optimization Methods and Software*, 19:15–40, 2004.

[34] Roger Fletcher, Sven Leyffer, Daniel Ralph, and Stefan Scholtes. Local convergence of sqp methods for mathematical programs with equilibrium constraints. *SIAM Journal on Optimization*, 17(1):259–286, 2006.

[35] Roger Fletcher, Sven Leyffer, and Philippe L. Toint. On the global convergence of a Filter–SQP algorithm. *SIAM Journal on Optimization*, 13(1):44–59, 2002.

[36] Masao Fukushima, Zhi-Quan Luo, and Jong-Shi Pang. A globally convergent sequential quadratic programming algorithm for mathematical programs with linear complementarity constraints. *Computational Optimization and Applications*, 10(1):5–34, 1998.

[37] Masao Fukushima and Jong-Shi Pang. Convergence of a smoothing continuation method for mathematical programs with complementarity constraints. In *In Michel Théra and Rainer Tichatschke, editors. Ill-posed variational problems and regularization techniques (Trier 1998) [Lecture Notes in Economics and Mathematical Systems 477.]*, pages 99–110, Berlin, 1999. Springer.

[38] Masao Fukushima and Paul Tseng. An implementable active-set algorithm for computing a b-stationary point of a mathematical program with linear complementarity constraints. *SIAM Journal on Optimization*, 12(3):724–739, 2002.

[39] Philip E. Gill, Walter Murray, and Michael A. Saunders. User's guide for snopt version 6: A fortran package for large-scale nonlinear programming. Technical report, Systems Optimization Laboratory, Stanford University, 2002.

[40] K.-C. Goh, M.G. Safonov, and G.P. Papavassilopoulos. A global optimization approach for the bmi problem. *Decision and Control, 1994., Proceedings of the 33rd IEEE Conference on*, 3:2009–2014, 14-16 Dec 1994.

[41] Gene H. Golub, Michael Heath, and Grace Wahba. Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics*, 21:215–223, 1979.

[42] Isabelle Guyon and Andre Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[43] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422, 2002.

[44] Trevor Hastie, Saharon Rosset, Robert Tibshirani, and Ji Zhu. The entire regularization path for the support vector machine. *Journal of Machine Learning Research*, 5:1391–1415, 2004.

[45] Reiner Horst and Hoang Tuy. *Global Optimization: Deterministic Approaches*. Springer, New York, 1996.

[46] Jing Hu, John E. Mitchell, Jong-Shi Pang, Kristin P. Bennett, and Gautam Kunapuli. On the global solution of linear programs with complementarity constraints. *Manuscript, Department of Mathematical Sciences, Rensselaer Polytechnic Institute, Troy, NY*, 2007.

[47] X. X. Huang, X. Q. Yang, and K. L. Teo. Partial augmented lagrangian method and mathematical programs with complementarity constraints. *Journal of Global Optimization*, 35:235–254, 2006.

[48] Houyuan Jiang and Daniel Ralph. Smooth SQP methods for mathematical programs with nonlinear complementarity constraints. *SIAM Journal on Optimization*, 10(3):779–808, 2000.

[49] Thorsten Joachims. Training linear svms in linear time. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–226, New York, NY, USA, 2006. ACM.

[50] Sathiya S. Keerthi, Vikas Sindhwani, and Oliver Chapelle. An efficient method for gradient-based adaptation of hyperparameters in svm models. In *Advances in Neural Information Processing Systems 19*, pages 673–680, Cambridge, MA, 2007. MIT Press.

[51] Hyunsoo Kim, Gene H. Golub, and Haesun Park. Missing value estimation for DNA microarray gene expression data: local least squares imputation. *Bioinformatics*, 21(2):187–198, 2005.

[52] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *International Joint Conference on Artificial Intelligence*, pages 1137–1145, 1995.

[53] Ron Kohavi and George H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.

[54] Gautam Kunapuli, Kristin P. Bennett, Jing Hu, and Jong-Shi Pang. Bilevel model selection for support vector machines. In Pierre Hansen and Panos Pardalos, editors, *Data Mining and Mathematical Programming [CRM Proceedings and Lecture Notes]*, volume 45. American Mathematical Society, 2008.

[55] Gert R. G. Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El-Ghaoui, and Michael I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Maching Learning Research*, 5:27–72, 2004.

[56] G. Lin and Masao Fukushima. Hybrid algorithms with active set identification for mathematical programs with complementarity constraints. Technical Report 2002-008, Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University, Kyoto, Japan, 2002.

[57] Roderick J. A. Little. Regression with missing x's: A review. *Journal of the American Statistical Association*, 87(420):1227–1237, 1992.

[58] Roderick J. A. Little and Donald B. Rubin. *Statistical analysis with missing data, 2nd Ed.* John Wiley and Sons, Chichester, 2002.

[59] Xinwei Liu and Jie Sun. Generalized stationary points and a robust interior point method for mathematical programs with equilibrium constraints. *Mathematical Programming*, 101:231–261, 2004.

[60] Zhi-Quan Luo, Jong-Shi Pang, and Daniel Ralph. *Mathematical Programs With Equilibrium Constraints*. Cambridge University Press, Cambridge, 1996.

[61] Olvi L. Mangasarian. Misclassification minimization. *Journal of Global Optimization*, 5:309–323, 1994.

[62] Olvi L. Mangasarian. The linear complementarity problem as a separable bilinear program. *Journal of Global Optimization*, 6:153–161, 1995.

[63] Olvi L. Mangasarian. Solution of general linear complementarity problems via nondifferentiable concave minimization. *Acta Mathematica Vietnamica*, 22(1):199–205, 1997.

[64] Olvi L. Mangasarian. Exact 1-norm support vector machines via unconstrained convex differentiable minimization. *Journal of Machine Learning Research*, 7:1517–1530, 2006.

[65] Olvi L. Mangasarian and M. E. Thompson. Massive data classification via unconstrained support vector machines. *Journal of Optimization Theory and Applications*, 131(3):315–325, 2006.

[66] Cheng Soon Ong, Alexander J. Smola, and Robert C. Williamson. Learning the kernel with hyperkernels. *Journal of Machine Learning Research*, 6:1043–1071, 2005.

[67] Jiri V. Outrata, Michal Kocvara, and Jochem Zowe. *Nonsmooth Approach to Optimization Problems with Equilibrium Constraints: Theory, Applications and Numerical Results*. Kluwer Academic Publishers, Dordrecht, 1998.

[68] Jong-Shi Pang. private communication, 2008.

[69] Joao Pedro Pedroso and Noboru Murata. Support vector machines with different norms: motivation, formulations and results. *Pattern Recognition Letters*, 22(12):1263–1272, 2001.

[70] John C. Platt. Fast training of support vector machines using sequential minimal optimization. *Advances in kernel methods: support vector learning*, pages 185–208, 1999.

[71] Tomaso Poggio, Vincent Torre, and Christof Koch. Computational vision and regularization theory. *Nature*, 317(26):314–319, 1985.

[72] Daniel Ralph and Stephen J. Wright. Some properties of regularization and penalization schemes for mpecs. *Optimization Methods and Software*, 19:527–556, 2004.

[73] Donald B. Rubin. Inference and missing data. *Biometrika*, 63:581–592, 1976.

[74] Donald B. Rubin. *Multiple imputation for nonresponse in surveys*. John Wiley and Sons, New York; Chichester, 1987.

[75] Donald B. Rubin. Multiple imputation after 18+ years (with discussion). *Journal of the American Statistical Association*, 91:473–520, 1996.

[76] Joseph L. Schafer. *Analysis of Incomplete Multivariate Data*. Chapman & Hall, London, 1997.

[77] Holger Scheel and Stefan Scholtes. Mathematical programs with complementarity constraints: Stationarity, optimality, and sensitivity. *Mathematics of Operations Research*, 25(1):1–22, 2000.

[78] Bernhard Schölkopf, Robert C. Williamson, Alex J. Smola, John Shawe-Taylor, and John C. Platt. Support vector method for novelty detection. In S.A. Solla, T.K. Leen, and Klaus-Robert Müller, editors, *Advances in Neural Information Processing Systems*, volume 12, pages 582–588. MIT Press, 2000.

[79] Stefan Scholtes. Convergence properties of a regularization scheme for mathematical programs with complementarity constraints. *SIAM Journal on Optimization*, 11(4):918–936, 2000.

[80] Stefan Scholtes and Michael Stöhr. Exact penalization of mathematical programs with equilibrium constraints. *SIAM Journal on Control and Optimization*, 37(2):617–652, 1999.

[81] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[82] Hanif D. Sherali and Amine Alameddine. A new reformulation-linearization technique for bilinear programming problems. *Journal of Global Optimization*, 2(4):379–410, 1992.

[83] Alex J. Smola. Regression estimation with support vector learning machines (master's thesis). Technical report, Technische Universität München., 1996.

[84] Alex J. Smola and Bernhard Schölkopf. A tutorial on support vector regression. *NeuroCOLT2 Technical Report NC2-TR-1998-030*, 1998.

[85] H. Van Stackelberg. *The Theory of Market Economy.* Oxford University Press, Oxford, 1952.

[86] Hoang Tuy. *Convex Analysis and Global Optimization.* Kluwer Academic, Dordrecht, The Netherlands, 1998.

[87] Vladimir Vapnik. *The Nature of Statistical Learning Theory.* Springer-Verlag, New York, 2000.

[88] Gang Wang, Dit-Yan Yeung, and Frederick H. Lochovsky. Two-dimensional solution path for support vector regression. In *ICML '06: Proceedings of the 23rd International Conference on Machine Learning*, pages 993–1000, 2006.

[89] Jane J. Ye. Necessary and sufficient optimality conditions for mathematical programs with equilibrium constraints. *Journal of Mathematical Analysis and Applications*, 307:350–369, 2005.

[90] Ji Zhu, Saharon Rosset, Trevor Hastie, and Robert Tibshirani. 1-norm support vector machines. In Sebastian Thrun, Lawrence K. Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16 [Neural Information Processing Systems, NIPS 2003, December 8-13, 2003, Vancouver and Whistler, British Columbia, Canada]*. MIT Press, 2004.

[91] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal Of The Royal Statistical Society Series B*, 67(2):301–320, 2005.

APPENDIX A
Code Fragments of AMPL Models

The AMPL code for the various models in the thesis are presented here for completeness. Certain parameters like **tol**, $\underline{C}$ and $\overline{C}$ have to be defined by the user.

A.1 Model Selection for SV Classification

```
#######################################################################
# Bilevel cross-validation for support vector classification

# Data dimensions
param N;      # number of dimensions
param T;      # number of folds
param Ltrain; # total number of points available for CV

# Index sets
set N{1..T};    # indexes the validation sets
set Nbar{1..T}; # indexes the training sets

# Training data
param X{1..Ltrain, 1..D};
param y{1..Ltrain};

# Parameter Bounds
param CLower;
param CUpper;
param wbarLower;
param wbarUpper;

# Tolerance for complementarity conditions
param tol;

# Outer-level variables
var C >= CLower, <= CUpper;
var wbar{1..D} >= wbarLower, <= wbarUpper;


# Inner-level primal variables
var z{t in 1..T, i in N[t]};
var w{1..D, 1..T};
var b{1..T};
var xi{t in 1..T, i in Nbar[t]}
```

```
# Inner-level dual variables
var zeta{t in 1..T, i in N[t]};
var alpha{t in 1..T, i in Nbar[t]};
var gammaP{1..D, 1..T} >= 0;
var gammaM{1..D, 1..T} >= 0;

# Outer-level Objective
minimize GeneralizationError:
(1/T) * sum{t in 1..T, i in N[t]} (1/card(N[t])) * z[t,i];

# Misclassification minimization constraints
subject to stepLowerComplementarity{t in 1..T, i in N[t]}:
0 <= z[t,i] complements
 (y[i]*(sum{f in 1..D} X[i,f]*w[f,t] - b[t])+ zeta[t,i]) >= 0;

subject to stepUpperComplementarity{t in 1..T, i in N[t]}:
0 <= zeta[t,i] complements 1 - z[t,i] >= 0;

# Distance minimization constraint
subject to distanceMinConstraint{t in 1..T, i in N[t]}:
z[t,i] >= 1 - y[i]*(sum{f in 1..D} X[i,f]*w[f,t] - b[t]);

# Hyperplane constraints and complementarity
subject to HyperplaneComplementarity{t in 1..T, i in Nbar[t]}:
0 <= alpha[t,i] complements
    (y[i]*(sum{f in 1..D} X[i,f]*w[f,t] - b[t]) - 1 + xi[t,i]) >= 0;

# Error constraints and Multiplier bounds
subject to ErrorComplementarity{t in 1..T, i in Nbar[t]}:
0 <= xi[t,i] complements C - alpha[t,i]) >= 0;

# Box constraints on w[f,t]
subject to BoxNegativeComplementarity{f in 1..D, t in 1..T}:
0 <= gammaM[f,t] complements wbar[f] + w[f,t] >= 0;

subject to BoxPositiveComplementarity{f in 1..D, t in 1..T}:
0 <= gammaP[f,t] complements wbar[f] - w[f,t] >= 0;

# First order conditions
subject to KKTConditionWRTw{f in 1..D, t in 1..T}:
w[f,t] - sum{i in Nbar[t]} y[i]*alpha[t,i]*X[i,f]
                           + gammaP[f,t] - gammaM[f,t] = 0;

subject to KKTConditionWRTb{t in 1..T}:
sum{i in Nbar[t]} y[i]*alpha[t,i] = 0;
```

A.2 Model Selection for SV Regression

```
#######################################################################
# Bilevel cross-validation for support vector regression

# Data dimensions
param D;      # number of dimensions
param T;      # number of folds
param Ltrain; # size of training set

# Setup index sets
set N{1..T};    # indexes the validation sets
set Nbar{1..T}; # indexes the training sets

# Training data
param X{1..Ltrain, 1..D};
param y{1..Ltrain};

# Parameter Bounds
param CLower;
param CUpper;
param epsilonLower;
param epsilonUpper;
param wBarLower;
param wBarUpper;

# Tolerance for complementarity conditions
param tol;

# Outer-level variables
var C >= CLower, <= CUpper;
var epsilon >= epsilonLower, <= epsilonUpper;
var wbar{1..D} >= wbarLower, <= wbarUpper;

# Inner-level primal variables
var z{t in 1..T, i in N[t]};
var w{1..D, 1..T};
var b{1..T};
var xi{t in 1..T, i in Nbar[t]};

# Inner-level dual variables
var gammaP{1..D, 1..T};
var gammaM{1..D, 1..T};
var alphaP{t in 1..T, i in Nbar[t]};
var alphaM{t in 1..T, i in Nbar[t]};
```

```
# Outer-level Objective
minimize GeneralizationError:
(1/T) * (sum{t in 1..T, i in N[t]} (1/card(N[t])) * z[t,i]);

# Validation errors
subject to zLowerConstraint{t in 1..T, i in N[t]}:
z[t,i] >= -sum{f in 1..D} X[i,f]*w[f,t] + b[t] + y[i];

subject to zUpperConstraint{t in 1..T, i in N[t]}:
z[t,i] >= sum{f in 1..D} X[i,f]*w[f,t] - b[t] - y[i];

# Box constraints on w[f,t]
subject to BoxNegativeComplementarity{f in 1..D, t in 1..T}:
0 <= gammaM[f,t] complements wbar[f] + w[f,t] >= 0;

subject to BoxPositiveComplementarity{f in 1..D, t in 1..T}:
0 <= gammaP[f,t] complements wbar[f] - w[f,t] >= 0;

# Tube constraints
subject to UpperHyperplaneComplementarity{t in 1..T, i in Nbar[t]}:
0 <= alphaM[t,i] complements
  (sum{f in 1..D} X[i,f]*w[f,t] - b[t] - y[i] + epsilon + xi[t,i]) >= 0;

subject to LowerHyperplaneComplementarity{t in 1..T, i in Nbar[t]}:
0 <= alphaP[t,i] complements
 (-sum{f in 1..D} X[i,f]*w[f,t] + b[t] + y[i] + epsilon + xi[t,i]) >= 0;

# Error Constraints
subject to ErrorComplementarity{t in 1..T, i in Nbar[t]}:
0 <= xi[t,i] complements C - alphaP[t,i] - alphaM[t,i] >= 0;

# Classifier component constraints
subject to KKTConditionwrtW{f in 1..D, t in 1..T}:
w[f, t] + (sum{i in Nbar[t]} (alphaP[t,i] - alphaM[t,i])*X[i,f])
                                    + gammaP[f, t] - gammaM[f, t] = 0;

# KKT with respect to b[t] constraint
subject to KKTConditionwrtB{t in 1..T}:
sum{i in Nbar[t]} (alphaP[t,i] - alphaM[t,i]) = 0;
```

A.3 Missing-value Imputation for SV Regression

```
#######################################################################
# Bilevel missing-value imputation for support vector regression

# Data dimensions
param D;      # number of dimensions
param T;      # number of folds
param Ltrain; # total number of points available for crossvalidation

# Setup index sets (to be initialized in io.script)
set V ordered;          # indices for all the missing data
set M{1..T};            # validation sets for missing data
set Mbar{1..T};         # training sets for missing data
set N{1..T};            # validation sets for full data
set Nbar{1..T};         # training sets for full data
set F{1..Ltrain};       # indices for the known x features
set Fbar{1..Ltrain};    # indices for missing x features

# Training data
param x{1..Ltrain, 1..D};
param y{1..Ltrain};

# Hyper-parameter Bounds
param CLower;
param CUpper;
param epsLower;
param epsUpper;
param xLower;
param xUpper;
param wLower{1..D};
param wUpper{1..D};

# Tolerance for complementarity conditions
param tol;

# Inner-level variables
var aP{t in 1..T, j in Mbar[t] union Nbar[t]};
var aM{t in 1..T, j in Mbar[t] union Nbar[t]};
var xi{t in 1..T, j in Mbar[t] union Nbar[t]};
var z{t in 1..T, i in M[t] union N[t]};
var w{t in 1..T, f in 1..D};
var b{t in 1..T};
var r{t in 1..T, j in Mbar[t], f in Fbar[j]};
var s{t in 1..T, j in M[t] union Mbar[t], f in Fbar[j]};
```

```
# Outer-level variables
var C >= CLower, <= CUpper;
var epsilon >= epsLower, <= epsUpper;
var xm{j in V, f in Fbar[j]} >= xLower[f], <= xUpper[f];

# Objective function
minimize GeneralizationError:
sum{t in 1..T, i in M[t] union N[t]} z[t,i] / (card(M[t] union N[t]) * T);

# Constraints
subject to ValidationUpper{t in 1..T, i in N[t]}:
y[i] - sum{f in 1..D} (x[i,f]*w[t,f]) + b[t] + z[t,i] >= 0;

subject to ValidationLower{t in 1..T, i in N[t]}:
sum{f in 1..D} (x[i,f]*w[t,f]) - b[t] - y[i] + z[t,i] >= 0;

subject to ValidationUpperMissing{t in 1..T, i in M[t]}:
y[i] - sum{f in F[i]} (x[i,f]*w[t,f])
     - sum{f in Fbar[i]} s[t,i,f] + b[t] + z[t,i] >= 0;

subject to ValidationLowerMissing{t in 1..T, i in M[t]}:
sum{f in F[i]} (x[i,f]*w[t,f]) +
sum{f in Fbar[i]} s[t,i,f] - b[t] - y[i] + z[t,i] >= 0;

subject to TrainingUpperComplementarity{t in 1..T, j in Nbar[t]}:
0 <= aP[t,j] complements
  y[j] - sum{f in 1..D} x[j,f]*w[t,f] + b[t] + epsilon + xi[t,j] >= 0;

subject to TrainingLowerComplementarity{t in 1..T, j in Nbar[t]}:
0 <= aM[t,j] complements
  sum{f in 1..D} x[j,f]*w[t,f] - b[t] - y[j] + epsilon + xi[t,j] >= 0;

subject to TrainingUpperMissingComplementarity{t in 1..T, j in Mbar[t]}:
0 <= aP[t,j] complements
 y[j] - sum{f in F[j]} (x[j,f]*w[t,f])
      - sum{f in Fbar[j]} s[t,j,f] + b[t] + e[t,j] + xi[t,j] >= 0;

subject to TrainingLowerMissingComplementarity{t in 1..T, j in Mbar[t]}:
0 <= aM[t,j] complements
   sum{f in F[j]} (x[j,f]*w[t,f]) +
   sum{f in Fbar[j]} s[t,j,f] - b[t] - y[j] + e[t,j] + xi[t,j] >= 0;

subject to MultiplierBoundsComp{t in 1..T, j in Mbar[t] union Nbar[t]}:
0 <= xi[t,j] complements (C - aP[t,j] - aM[t,j]) >= 0;
```

```
subject to KKTConditionWRTw{t in 1..T, f in 1..D}:
w[t,f]
 + sum{j in Mbar[t] union Nbar[t]:f in F[j]} (aP[t,j] - aM[t,j])*x[j,f]
 + sum{j in Mbar[t] union Nbar[t]:f in Fbar[j]} r[t,j,f] = 0;

subject to KKTConditionWRTb{t in 1..T}:
sum{j in Mbar[t] union Nbar[t]} (aP[t,j] - aM[t,j]) = 0;

subject to PrimalImputationError{t in 1..T,
    j in M[t] union Mbar[t], f in Fbar[j]}: s[t,j,f] - w[t,f]*xm[j,f] = 0;

subject to DualImputationError{t in 1..T, j in Mbar[t], f in Fbar[j]}:
r[t,j,f] - (aP[t,j] - aM[t,j])*xm[j,f] = 0;

subject to rBoundLL{t in 1..T, j in Mbar[t], f in Fbar[j]}:
r[t,j,f] >= xLower[f]*(aP[t,j] - aM[t,j])
                   - CUpper*xm[j,f] + CUpper*xLower[f];

subject to rBoundUU{t in 1..T, j in Mbar[t], f in Fbar[j]}:
r[t,j,f] >= xUpper[f]*(aP[t,j] - aM[t,j])
                   + CUpper*xm[j,f] - CUpper*xUpper[f];

subject to rBoundLU{t in 1..T, j in Mbar[t], f in Fbar[j]}:
r[t,j,f] <= xUpper[f]*(aP[t,j] - aM[t,j])
                   - CUpper*xm[j,f] + CUpper*xUpper[f];

subject to rBoundUL{t in 1..T, j in Mbar[t], f in Fbar[j]}:
r[t,j,f] <= xLower[f]*(aP[t,j] - aM[t,j])
                   + CUpper*xm[j,f] - CUpper*xLower[f];

subject to sBoundLL{t in 1..T, j in M[t] union Mbar[t], f in Fbar[j]}:
s[t,j,f] >= xLower[f]*w[t,f] + wLower[f]*xm[j,f] - xLower[f]*wLower[f];

subject to sBoundUU{t in 1..T, j in M[t] union Mbar[t], f in Fbar[j]}:
s[t,j,f] >= xUpper[f]*w[t,f] + wUpper[f]*xm[j,f] - xUpper[f]*wUpper[f];

subject to sBoundLU{t in 1..T, j in M[t] union Mbar[t], f in Fbar[j]}:
s[t,j,f] <= xUpper[f]*w[t,f] + wLower[f]*xm[j,f] - xUpper[f]*wLower[f];

subject to sBoundUL{t in 1..T, j in M[t] union Mbar[t], f in Fbar[j]}:
s[t,j,f] <= xLower[f]*w[t,f] + wUpper[f]*xm[j,f] - xLower[f]*wUpper[f];
```