Centre de Recherches Mathématiques
CRM Proceedings and Lecture Notes
Volume **45**, 2008

Bilevel Model Selection for Support Vector Machines

Gautam Kunapuli, Kristin P. Bennett, Jing Hu, and Jong-Shi Pang

Abstract. The successful application of Support Vector Machines (SVMs), kernel methods and other statistical machine learning methods requires selection of model parameters based on estimates of the generalization error. This paper presents a novel approach to systematic model selection through bilevel optimization. We show how modelling tasks for widely used machine learning methods can be formulated as bilevel optimization problems and describe how the approach can address a broad range of tasks—among which are parameter, feature and kernel selection In addition, we also discuss the challenges in implementing these approaches and enumerate opportunities for future work in this emerging research area.

1. Introduction

Currently, Support Vector Machines (SVM) [12] and kernel methods enjoy enormous popularity in the machine learning community. There are several reasons: theoretically, their foundations are rooted in Structural Risk Minimization (SRM) [54], which leads to excellent generalization capabilities; practically, they are very flexible owing to the fact that they are modular and can be "kernelized" to capture highly nonlinear relationships [51]; and computationally, they are tractable and can be applied to large high-dimensional data sets which contain several thousand points. The kernel framework can be readily adapted to many learning tasks, e.g. regression, classification, ranking, and novelty detection. One other reason why SVMs may be considered one of the pre-eminent methods is because, with regard to applicability, they have had wide ranging success on a variety of real-world applications. However, the many papers reporting the success of such methods frequently gloss over an important issue: model selection.

In kernel methods, the root learning task is to construct a linear function that minimizes a regularized convex loss function. Nonlinear functions can then be constructed using the so-called "kernel trick". The resulting optimization problem is convex, but it typically contains hyper-parameters that must be selected by the

2000 *Mathematics Subject Classification.* Primary: TO BE COMPLETED; Secondary: TO BE COMPLETED..

This work was supported in part by the Office of Naval Research under grant no. N00014-06-1-0014.

This is the final form of the paper.

user. For example, in SVMs, the appropriate kernel function and trade-off parameter between error and regularization must both be selected. Certain loss functions, such as the ε-insensitive loss, require the selection of additional hyper-parameters. There have been many interesting attempts to pick these hyper-parameters; notable among these are approaches that use bounds [10] or attempt to trace the complete regularization path of the SVM [31]. However, the most systematic, commonly used and widely accepted method for selecting these hyper-parameters is still T-fold cross validation (CV).

The focus of this chapter is to demonstrate how T-fold cross-validation model selection for many different learning tasks can be formulated as bilevel optimization problems. We begin with a review of the cross-validation problem. We illustrate the bilevel programming model using support vector regression as an example. Then, we introduce the generic T-fold CV formulation and discuss many possible variations. We briefly review possible methods for solving the models and direct the reader to references that present computational results for one approach applied to support vector regression and classification. Solution of the other resulting problems is left as a significant and compelling challenge to mathematical programming researchers.

1.1. Model selection. 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 [20]. 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. 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 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 [5, 30], kernel construction [37, 48], and multi-task learning [9, 21]. For such high-dimensional problems, greedy strategies such as stepwise regression, backward elimination, filter methods, or genetic algorithms are used. Yet, these heuristic methods, including grid search, have a fundamental deficiency in addition to their practical inefficiency; namely, they are incapable of assuring the overall quality of the produced "solution".

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 [31] 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. From the standpoint of "optimizing model selection" using out-of-sample estimates, there is an urgent need for improved methodologies that combine sound theoretical foundation and robust computational efficiency. This paper proposes one such methodology that is based on the methods of *bilevel optimization*.

In addition to model selection for support vector machines through continuous cross validation, the bilevel approach can also be applied to a wide variety of problems like semi-supervised learning, predicting missing values in the data, kernel selection, multi-task learning and complexity minimization. This is because each of these problems can be formulated as a bilevel program where 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. For example, in cross validation, training is performed in the inner level and validation in the outer level. Prior bilevel approaches have been developed and successfully used for inner-level problems with closed form solutions and a single parameter, e.g. the generalized cross-validation method for selecting the ridge parameter in ridge regression [28]. Still, these approaches are limited to a single hyper-parameter and inner-level function with a closed-form solution.

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.

1.2. Bilevel programming. 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 [6], and applied to defence 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 [53], whose origin predates the work of Bracken and McGill.

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.,

$$y(x) \in \Psi(x) = \arg\max_{y \in Y} f(x, y). \tag{1.1}$$

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

$$x \equiv \max_{x \in X} \{F(x, y) \mid y \in \Psi(x)\}. \tag{1.2}$$

Equations (1.1) and (1.2) can be combined to express the Stackelberg game compactly as

$$\begin{aligned} &\max_{x \in X, y} F(x, y) \\ &\text{s. t. } y \in \arg\max_{\eta \in Y} f(x, \eta). \end{aligned} \tag{1.3}$$

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 general bilevel program formulated by Bracken and McGill:

$$\begin{aligned} &\max_{x \in X, y} F(x, y) && \text{outer level} \\ &\text{s. t. } G(x, y) \leq 0, \\ &\qquad y \in \left\{ \begin{array}{l} \arg\max_{y \in Y} f(x, y) \\ \text{s. t. } g(x, y) \leq 0 \end{array} \right\}, && \text{inner level} \end{aligned} \tag{1.4}$$

The bilevel program, (1.4), is a generalization of several well-known optimization problems as noted in [18]. If $F(x, y) = -f(x, y)$, we have 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.

Now, a given model selection problem can be recast in the bilevel framework by optimizing the generalization criteria in the outer level while performing the required machine learning task in the inner level (see Figure 1). This reformulation

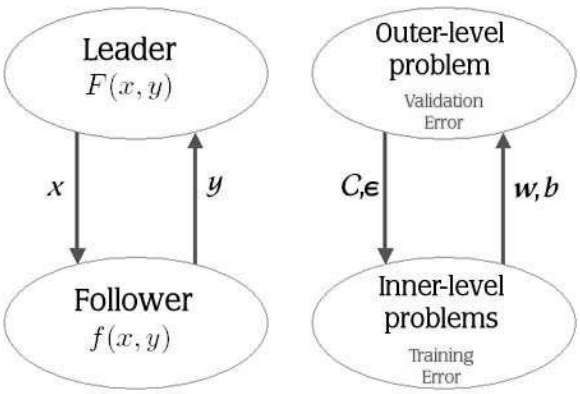

FIGURE 1. The Stackelberg game (left), showing the hierarchy between the leader and the follower; Cross validation modelled as a bilevel program (right), showing the interaction between the parameters, which are optimized in the outer level and the models which are trained in the inner level.

gives rise to a significant difficulty: many machine learning problems are convex and differentiable, while the bilevel program is non-convex and functions like $\Psi(x)$, which are not even Fréchet differentiable in general, occur in the constraints—for example, in (1.1). There exist techniques to convert bilevel programs into solvable optimization problems such as using implicit function theorems. However, we will restrict our attention to the approach where the inner-level problems are replaced by their Karush–Kuhn–Tucker conditions, a semi-infinite system of inequalities or finite-dimensional variational inequalities. Such an optimization problem is called a *Mathematical Program with Equilibrium Constraints* (MPEC).

The systematic study of the bilevel optimization problem and its MPEC extension attracted the intensive attention of mathematical programmers about a decade ago with the publication of a focused monograph [40], which is followed by two related monographs, [49, 18]. During the past decade, there has been an explosion of research on these optimization problems. See the annotated bibliography [19] which contains many references. In general, bilevel programs/MPECs provide a powerful computational framework for dealing with hyper-parameter identification problems in an optimization setting. As such, they offer a novel paradigm for dealing with the model selection problems described in this chapter. We illustrate the bilevel paradigm on a specific learning task—linear SV regression—and then proceed with the more general formulation.

2. Parameter Selection for Linear SV Regression

Cross validation for support vector regression was the first problem to reformulated as a bilevel program that can perform parameter selection [3]. The learning task is to construct a linear regression function that minimizes the ε-insensitive error regularized by the 2-norm of the weights of the linear function. Thus, SV regression has two hyper-parameters to be picked by T-fold cross validation. More specifically the goal is to find a function, $f(x) = x'w$, such that the the function

generalizes well on future data e.g. $f(x) \approx y$ and, typically the generalization error is estimated by cross validation.

2.1. Cross validation. We start with a few words about our notation. The training set consists of ℓ pairs of data and labels, $\{(\mathbf{x}_i, y_i)\}_{i=1}^{\ell} \subset \mathbb{R}^{n+1}$, denoted by Ω; thus, $|\Omega| = \ell$. Let the set of indices for the points in Ω be $\mathcal{N} = \{1, \ldots, \ell\}$. Since we are interested in performing T-fold cross validation, Ω is partitioned into T pairwise disjoint subsets, Ω_t, the *validation sets* within each fold. The sets $\overline{\Omega}_t = \Omega \backslash \Omega_t$ 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 tth training set, $\overline{\Omega}_t$ is used to train the linear function $\mathbf{x}w^t \in \mathbb{R}^n$. For simplicity of presentation, we ignore the bias term for now. For compactness of notation, the vectors $\mathbf{x}w^t$ are collected, column-wise, into the matrix $\mathbf{x}W \in \mathbb{R}^{n \times T}$. Also, given two vectors $\mathbf{x}r, \mathbf{x}s \in \mathbb{R}^n$, the complementarity condition $\mathbf{x}r \perp \mathbf{x}s$ is equivalent to $\mathbf{x}r'\mathbf{x}s = 0$. A vector of ones of arbitrary dimension is denoted $\mathbb{1}$.

The support vector regression (SVR) problem contains two hyper-parameters: the regularization constant, C, and the tube width, ε, for the ε-insensitive loss function, which are to be selected by cross validation. This is typically accomplished using grid search. Grid search involves discretizing the C-ε space, typically on a logarithmic scale of base 2 or 10 (see Fig. 2). At each grid point, the parameters of the regression functions, $\{\mathbf{w}^t\}_{t=1}^{T}$, are trained on the corresponding training sets, $\overline{\Omega}_t$, using the quadratic program

$$\mathbf{w}^t \in \underset{\mathbf{w} \in \mathbb{R}^n}{\arg\min} \left\{ \frac{1}{2}\|\mathbf{w}\|_2^2 + \frac{C}{|\overline{\mathcal{N}}_t|} \sum_{j \in \overline{\mathcal{N}}_t} \max(|\mathbf{x}x_j'\mathbf{w} - y_j| - \varepsilon, 0) \right\}. \tag{2.1}$$

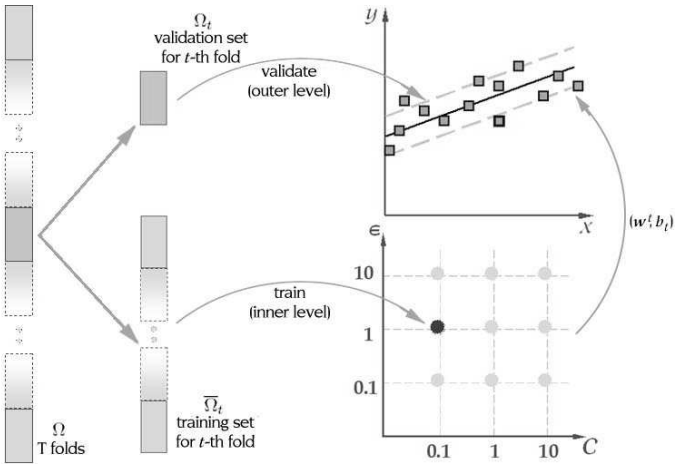

FIGURE 2. T-fold cross validation at a particular grid point, (C_k, ε_k), on a base-10 logarithmic grid.

Equation (2.1) represents the regression problem of finding a function, $f^\star : \mathbb{R}^n \to \mathbb{R}$, among a given class that minimizes the regularized risk functional,

$$R[f] \equiv \mathcal{P}[f] + \frac{C}{\ell} \sum_{j=1}^{\ell} \mathcal{L}\big(y_j, f(\mathbf{x}x_j)\big). \tag{2.2}$$

where $\mathcal{L}$ is the ε-insensitive loss function and $\mathcal{P}$ is the classic SVR ℓ_2-norm regularization operator. The regression functions are validated at each grid point by computing the generalization error, such as the mean average deviation (MAD), on the validation sets, Ω_t:

$$\Theta(\mathbf{W}) = \frac{1}{T} \sum_{t=1}^{T} \frac{1}{|\mathcal{N}_t|} \sum_{i \in \mathcal{N}_t} |\mathbf{x}_i' \mathbf{w} - y_i|, \tag{2.3}$$

or alternatively the mean squared error (MSE),

$$\Theta(\mathbf{W}) = \frac{1}{T} \sum_{t=1}^{T} \frac{1}{|\mathcal{N}_t|} \sum_{i \in \mathcal{N}_t} (\mathbf{x}_i' \mathbf{w} - y_i)^2. \tag{2.4}$$

The grid point with the smallest estimate of the generalization error, $\Theta(\mathbf{W})$, yields the choice of "optimal" hyper-parameters; this choice may be further refined by a local search. Assuming that each parameter is discretized to take on d distinct values, T-fold cross validation involves solving $O(Td^2)$ problems, a number that grows quickly as more accuracy is desired or if the number of parameters increases. This, combined with other issues described in Section 1.1, leads us to formulate the bilevel cross-validation model.

2.2. Bilevel cross validation. In a fairly general formulation in which we list only the essential constraints, the model selection bilevel program is to find the hyper-parameters ε and C and the hyperplanes, $\mathbf{w}^t$ in order to

$$\begin{aligned}
&\underset{C,\varepsilon,\mathbf{w}^t}{\text{minimize}}\ \Theta(\mathbf{W}) \\
&\text{subject to} \varepsilon, C \geq 0, \\
&\qquad \text{and for } t = 1, \ldots, T, \\
&\qquad \mathbf{w}^t \in \underset{\mathbf{w} \in \mathbb{R}^n}{\arg\min} \left\{ \frac{1}{2} \|\mathbf{w}\|_2^2 + \frac{C}{|\overline{\mathcal{N}}_t|} \sum_{j \in \overline{\mathcal{N}}_t} \max(|\mathbf{x}_j' \mathbf{w} - y_j| - \varepsilon, 0) \right\},
\end{aligned} \tag{2.5}$$

where the outer-level objective function, $\Theta(\mathbf{W})$, is an estimate of the generalization error and a performance measure as described in (2.3) and (2.4). Thus, the hyper-parameters are optimized in the outer level while the learning is performed in the inner level. It should be noted that the inner- and outer-level loss functions do not need to match. The inner-level optimization problem is the classic SV regression machine which uses ε-insensitive loss because it produces robust solutions that are sparse in the dual space. The ε-insensitive loss is not typically used for the generalization estimate in cross validation since it contains an unknown parameter ε; instead, we will use the MAD, (2.3), in the analysis below. The bilevel program that uses MSE can easily be formulated with only minor changes.

To solve (2.5), we rewrite each of the T inner-level problems, assuming that the hyper-parameters, C and ε, are fixed. In particular, for the tth problem, we introduce slack variables, $\boldsymbol{\xi}^t \geq 0$, to reformulate the max function in the inner level

using the standard trick from linear programming, to give the following convex, quadratic program:

$$
\begin{aligned}
&\min_{\mathbf{w}^t, \boldsymbol{\xi}^t} \frac{1}{2}\|\mathbf{w}\|_2^2 + \frac{C}{|\overline{\mathcal{N}}_t|} \sum_{j \in \overline{\mathcal{N}}_t} \xi_j^t \\
&\text{s. t.} \\
&\left.\begin{aligned} \xi_j^t &\geq y_j - \mathbf{x}_j' \mathbf{w} - \varepsilon, \\ \xi_j^t &\geq \mathbf{x}_j' \mathbf{w} - y_j - \varepsilon, \\ \xi_j^t &\geq 0, \end{aligned}\right\}, \quad \forall j \in \overline{\mathcal{N}}_t.
\end{aligned} \tag{2.6}
$$

This is the SV regression problem that trains $\mathbf{w}^t$ using the training set, $\overline{\Omega}_t$, within the tth fold. Let $\alpha_j^{t,+}$, $\alpha_j^{t,-} \geq 0$ be the Lagrange multipliers for the upper and lower hyperplane constraints respectively. Using these multipliers, we can write down the primal and dual feasibility and complementarity slackness conditions of (2.6) as follows.

$$
\left.\begin{aligned} 0 \leq \alpha_j^{t,+} &\perp y_j - \mathbf{x}_j' \mathbf{w}^t + \varepsilon + \xi_j^t \geq 0, \\ 0 \leq \alpha_j^{t,-} &\perp \mathbf{x}_j' \mathbf{w}^t - y_j + \varepsilon + \xi_j^t \geq 0, \\ 0 \leq \xi_j^t &\perp \frac{C}{|\overline{\mathcal{N}}_t|} - \alpha_j^{t,+} - \alpha_j^{t,-} \geq 0, \end{aligned}\right\} \quad \forall j \in \overline{\mathcal{N}}_t. \tag{2.7}
$$

There also exists the following first-order condition:

$$
\mathbf{w}^t + \sum_{j \in \overline{\mathcal{N}}_t} (\alpha_j^{t,+} - \alpha_j^{t,-}) \mathbf{x}_j = 0. \tag{2.8}
$$

Equations (2.7) and (2.8) together constitute the well-known Karush–Kuhn–Tucker first-order optimality conditions for (2.6). The outer-level objective also needs to be rewritten as it contains the absolute value function, which is not everywhere differentiable. This is easily achieved by introducing additional variables $\mathbf{z}^t$ and the constraints,

$$
-z_i^t \leq \mathbf{x}_i' \mathbf{w}^t - y_i \leq z_i^t, \quad \forall i \in \mathcal{N}_t, \tag{2.9}
$$

into the outer level. Here, $\mathbf{z}^t$ measures the mean absolute deviation of each point in the validation set, Ω_t, from the hyperplane, $\mathbf{w}^t$, trained on $\overline{\Omega}_t$. Thus, the overall two-level regression problem can be converted to a one-level problem as shown

below

$$
\begin{aligned}
&\min_{\substack{C,\varepsilon,\mathbf{w}^t,\mathbf{z}^t,\\ \boldsymbol{\alpha}^{t,\pm},\boldsymbol{\xi}^t}} \frac{1}{T}\sum_{t=1}^{T}\frac{1}{|\mathcal{N}_t|}\sum_{i\in\mathcal{N}_t} z_i^t \\
&\quad \text{s. t. } \varepsilon, C \geq 0, \\
&\qquad \text{and for } t = 1,\ldots,T, \\
&\qquad -z_i^t \leq \mathbf{x}_i'\mathbf{w}^t - y_i \leq z_i^t, && \forall i \in \mathcal{N}_t, \\
&\qquad \left.\begin{aligned} 0 \leq \alpha_j^{t,+} \perp y_j - \mathbf{x}_j'\mathbf{w}^t + \varepsilon + \xi_j^t \geq 0, \\ 0 \leq \alpha_j^{t,-} \perp \mathbf{x}_j'\mathbf{w}^t - y_j + \varepsilon + \xi_j^t \geq 0, \\ 0 \leq \xi_j^t \perp \frac{C}{|\overline{\mathcal{N}_t}|} - \alpha_j^{t,+} - \alpha_j^{t,-} \geq 0, \end{aligned}\right\} && \forall j \in \overline{\mathcal{N}}_t, \\
&\qquad \mathbf{w}^t + \sum_{j\in\overline{\mathcal{N}_t}} (\alpha_j^{t,+} - \alpha_j^{t,-})\mathbf{x}_j = 0.
\end{aligned} \tag{2.10}
$$

The optimization problem (2.10) is an instance of a *Linear Program with Equilibrium Constraints* (LPEC); it is a nonlinearly constrained problem and is non-convex in general because of the presence of the complementarity constraints. Several approaches can be used to solve (2.10) including exact penalty methods, integer programming approaches and nonlinear programming techniques, to name a few. We defer a discussion of these techniques until Section 8.

3. Generalized Bilevel Cross validation

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. 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. Keeping this in mind, T-fold cross validation for some general machine learning problem can be written as

$$
\begin{aligned}
&\underset{f^t,\boldsymbol{\lambda}}{\text{minimize}}\, \Theta(f^1|_{\Omega_1},\ldots,f^T|_{\Omega_T};\boldsymbol{\lambda}) \\
&\text{subject to } \boldsymbol{\lambda} \in \Lambda, \\
&\qquad \text{and for } t = 1,\ldots,T, \\
&\qquad f^t \in \arg\min_{f\in\mathcal{F}} \left\{ \mathcal{P}(f,\boldsymbol{\lambda}) + \sum_{j\in\overline{\mathcal{N}}_t} \mathcal{L}(y_j, f(\mathbf{x}_j),\boldsymbol{\lambda}) \right\}.
\end{aligned} \tag{3.1}
$$

Here, $f^t : (\mathbb{R}^n \times \mathbb{R}) \cap \mathcal{F} \to \mathbb{R}$ is the learning function trained within the tth 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. We provide some more examples below.

4. Parameter Selection for Linear SV Classification

In linear SV classification (SVC), the learning task is to construct a decision function that distinguishes one class from another. In this section, we extend the parameter selection idea introduced in the previous section to support vector classification and show how the bilevel formulation can handle a large number of hyper-parameters; this is a review of work that was first introduced in [36]. The inner-level problem is the standard SVC model [12] augmented with additional feature selection constraints. The outer-level objective minimizes the number of points misclassified in each of the T folds.

We again consider a labelled data set, $\Omega = \{(\mathbf{x}_i, y_i)\}_{i=1}^{\ell} \subset \mathbb{R}^{n+1}$, containing ℓ training examples. However, the labels y_i are restricted to ± 1 as we are interested in binary classification. We also introduce the bias into the decision rule: the hyperplane, $\mathbf{x}'\mathbf{w}^t - b_t = 0$, is to be represented by the pair $(\mathbf{w}^t, b_t)$. If $\mathbf{x}'\mathbf{w}^t > 0$, then $\mathbf{x}$ is predicted to be in class 1 by decision rule t, otherwise $\mathbf{x}$ is predicted to be in class -1. As before, the vectors, $\mathbf{w}^t$, are collected column-wise into the matrix $\mathbf{W} \in \mathbb{R}^{n\times T}$. The scalars, b_t, are collected into the vector $\mathbf{b} \in \mathbb{R}^T$.

Using this additional notation, the formulation of cross validation for SV classification as a bilevel program [36] is shown below.

$$
\begin{aligned}
&\underset{C,\overline{\mathbf{w}},\mathbf{w}^t,b_t}{\text{minimize}}\ \Theta(\mathbf{W},\mathbf{b}) \\
&\text{subject to } C_{\mathbf{lb}} \le C \le C_{\mathbf{ub}}, \\
&\qquad\qquad \overline{\mathbf{w}}_{\mathbf{lb}} \le \overline{\mathbf{w}} \le \overline{\mathbf{w}}_{\mathbf{ub}}, \\
&\qquad\qquad \text{and for } t = 1,\ldots,T, \\
&(\mathbf{w}^t, b_t) \in \underset{\substack{-\overline{\mathbf{w}}\le\mathbf{w}\le\overline{\mathbf{w}} \\ b\in\mathbb{R}}}{\arg\min} \left\{ \frac{1}{2}\|\mathbf{w}\|_2^2 + \frac{C}{|\overline{\mathcal{N}}_t|} \sum_{j\in\overline{\mathcal{N}}_t} \max(1 - y_j(\mathbf{x}_j'\mathbf{w} - b), 0) \right\}.
\end{aligned}
\tag{4.1}
$$

It should be noted that the T inner-level problems are nearly identical to the classical support vector classification problem except for the introduction of the additional constraints $-\overline{\mathbf{w}} \le \mathbf{w} \le \overline{\mathbf{w}}$. These constraints can be thought of as a weighted ℓ_∞-norm on $\mathbf{w}$ and provide additional regularization while also performing feature selection. To intuitively see how the box-constrained SV classifier (BoxSVC) performs feature selection, consider the following. An irrelevant feature would induce the corresponding weight in $\overline{\mathbf{w}}$ to be small, which, in turn, forces the respective weights in each fold, $\mathbf{w}^t$, to be small. Therefore, in addition to ensuring consistency across the folds, the box constraint performs capacity control leading to potentially improved overall generalization.

As a consequence, $\overline{\mathbf{w}} \in \mathbb{R}^n$ enters the outer level as a n-dimensional feature selecting hyper-parameter, with the box constraint entering each inner-level problem. Here, n is the dimension of the input space. The bilevel program (4.1) contains $n + v/1$ hyper-parameters that need to be optimized and model selection entails simultaneous parameter and input-space feature selection. In addition to the box constraints, bounds, $0 < C_{\mathbf{lb}} \le C_{\mathbf{ub}}$ and $0 < \overline{\mathbf{w}}_{\mathbf{lb}} \le \overline{\mathbf{w}}_{\mathbf{ub}}$, have also been introduced on the hyper-parameters in the outer level. This is to cut-off degenerate stationary points that are meaningless in the machine learning context.

4.1. The inner-level problems. Just as in the regression case, we can replace the inner-level problems with their corresponding constraints in order to convert the bilevel program to an MPEC. First, slack variables, $\boldsymbol{\xi}^t \geq 0$, are introduced to reformulate the non-differentiable max function in the inner-level objective giving the following quadratic program to train a decision rule within the tth fold:

$$\begin{aligned} &\min_{\mathbf{w}^t, b_t, \boldsymbol{\xi}^t} \frac{1}{2}\|\mathbf{w}^t\|_2^2 + \frac{C}{|\overline{\mathcal{N}}_t|}\sum_{j\in\overline{\mathcal{N}}_t}\xi_j^t \\ &\text{s. t. } -\overline{\mathbf{w}} \leq \mathbf{w}^t \leq \overline{\mathbf{w}}, \\ &\qquad \left.\begin{aligned} y_j(\mathbf{x}_j'\mathbf{w}^t - b_t) &\geq 1 - \xi_j^t, \\ \xi_j^t &\geq 0, \end{aligned}\right\}, \quad \forall j \in \overline{\mathcal{N}}_t. \end{aligned} \tag{4.2}$$

In addition to the Lagrange multipliers, α^t, for the hyperplane constraints and η^t for the non-negativity of ξ^t constraints, we introduce $\boldsymbol{\gamma}^{t,+}$ and $\boldsymbol{\gamma}^{t,-}$ for the lower and upper box constraints. Thus, the KKT complementarity conditions for (4.2) are

$$\left.\begin{aligned} &0 \leq \alpha_j^t \perp y_j(\mathbf{x}_j'\mathbf{w}^t - b_t) - 1 + \xi_j^t \geq 0, \\ &0 \leq \xi_j^t \perp \frac{C}{|\overline{\mathcal{N}}_t|} - \alpha_j^t \geq 0, \end{aligned}\right\}, \quad \forall j \in \overline{\mathcal{N}}_t,$$

$$\begin{aligned} &0 \leq \boldsymbol{\gamma}^{t,+} \perp \overline{\mathbf{w}} - \mathbf{w}^t \geq 0, \\ &0 \leq \boldsymbol{\gamma}^{t,-} \perp \overline{\mathbf{w}} + \mathbf{w}^t \geq 0, \end{aligned}$$

and the KKT first order conditions are

$$\begin{aligned} \mathbf{w}^t - \sum_{j\in\overline{\mathcal{N}}_t} y_j\alpha_j^t\mathbf{x}_j + \boldsymbol{\gamma}^{t,+} - \boldsymbol{\gamma}^{t,-} &= 0, \\ \sum_{j\in\overline{\mathcal{N}}_t} y_j\alpha_j^t &= 0. \end{aligned} \tag{4.3}$$

We now look at various outer-level objectives.

4.2. The outer-level objective. The standard performance measure used to validate classification models is the average misclassification error on the validation sets, i.e.,

$$\Theta(\mathbf{W}, \mathbf{b}) = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{|\mathcal{N}_t|}\sum_{i\in\mathcal{N}_t}[-y_i(\mathbf{x}_i'\mathbf{w}^t - b_t)]_\star, \tag{4.4}$$

where the inner summand counts the number of misclassifications of points in the validation set within each fold and the outer summand sums the averaged misclassification error over each fold. The objective, (4.4), is formulated in terms of the step function, which, for a vector, $\mathbf{r}_\star$, is defined as

$$(\mathbf{r}_\star)_i = \begin{cases} 1, & r_i > 0, \\ 0, & r_i \leq 0. \end{cases} \tag{4.5}$$

The step function, unlike the absolute value or the max functions, is not even continuous. This poses a problem, since now, the outer-level objective is not differentiable. However, the step function can be characterized as the solution to a

linear program [41]:

$$\mathbf{r}_\star = \arg\min_{\boldsymbol{\zeta}} \{-\boldsymbol{\zeta}'\mathbf{r} \mid 0 \leq \boldsymbol{\zeta} \leq \mathbb{1}\}. \tag{4.6}$$

Thus, in order to count the number of misclassifications in the T validation sets, it is necessary to introduce T additional inner-level problems of the type (4.6),

$$\boldsymbol{\zeta}^t \in \arg\min_{0 \leq \boldsymbol{\zeta} \leq \mathbb{1}} \left\{ \sum_{i \in \mathcal{N}_t} \zeta_i y_i (\mathbf{x}_i' \mathbf{w}^t - b_t) \right\}. \tag{4.7}$$

Then, as in the case of the inner-level problems (4.2), we can replace (4.6) with linear complementarity conditions. This necessitates the introduction of additional Lagrange multipliers, (4.6) also satisfies

$$\begin{aligned} 0 \leq \boldsymbol{\zeta} \perp -\mathbf{r} + \mathbf{z} \geq 0, \\ 0 \leq \mathbf{z} \perp \mathbb{1} - \boldsymbol{\zeta} \geq 0. \end{aligned} \tag{4.8}$$

Thus, combining (4.3)–(4.3), and (4.7)–(4.8), we can replace the bilevel program with a single-level program, which is also an instance of an MPEC. Note that the variables, $\mathbf{z}^t$, measure the distance of the misclassified validation points from the trained hyperplane within each fold

$$\begin{aligned} \min_{\substack{C, \overline{\mathbf{w}}, \mathbf{w}^t, b_t, \mathbf{z}^t, \\ \boldsymbol{\xi}^t, \boldsymbol{\zeta}^t, \alpha b^t, \boldsymbol{\gamma}^{t,\pm}}} \; & \frac{1}{T} \sum_{t=1}^{T} \frac{1}{|\mathcal{N}_t|} \sum_{i \in \mathcal{N}_t} \zeta_i^t \\ & \text{s. t. } C_{\mathbf{lb}} \leq C \leq C_{\mathbf{ub}}, \overline{\mathbf{w}}_{\mathbf{lb}} \leq \overline{\mathbf{w}} \leq \overline{\mathbf{w}}_{\mathbf{ub}}, \\ & \text{and for } t = 1, \ldots T, \\ & \left. \begin{aligned} 0 \leq \zeta_i^t \perp y_i(\mathbf{x}_i' \mathbf{w}^t - b_t) + z_i^t \geq 0, \\ 0 \leq z_i^t \perp 1 - \zeta_i^t \geq 0, \end{aligned} \right\}, \qquad \forall i \in \mathcal{N}_t, \\ & \left. \begin{aligned} 0 \leq \alpha_j^t \perp y_j(\mathbf{x}_j' \mathbf{w}^t - b_t) - 1 + \xi_j^t \geq 0, \\ 0 \leq \xi_j^t \perp \frac{C}{|\overline{\mathcal{N}}_t|} - \alpha_j^t \geq 0, \end{aligned} \right\}, \qquad \forall j \in \overline{\mathcal{N}}_t, \\ & 0 \leq \boldsymbol{\gamma}^{t,+} \perp \overline{\mathbf{w}} - \mathbf{w}^t \geq 0, \\ & 0 \leq \boldsymbol{\gamma}^{t,-} \perp \overline{\mathbf{w}} + \mathbf{w}^t \geq 0, \\ & \mathbf{w}^t - \sum_{j \in \overline{\mathcal{N}}_t} y_j \alpha_j^t \mathbf{x}_j + \boldsymbol{\gamma}^{t,+} - \boldsymbol{\gamma}^{t,-} = 0, \\ & \sum_{j \in \overline{\mathcal{N}}_t} y_j \alpha_j^t = 0, \end{aligned} \tag{4.9}$$

The outer-level objective, $\Theta(\mathbf{W}, \mathbf{b})$, can be treated not just as performance measure that can estimate the generalization error but also as a loss function. This observation combined with the flexibility of the bilevel approach allows us to formulate several different approaches to model selection.

The most straightforward variation is to use the average distance of the misclassified validation point from the trained decision rule within each fold i.e.,

$$\Theta(\mathbf{W}, \mathbf{b}) = \frac{1}{T} \sum_{t=1}^{T} \frac{1}{|\mathcal{N}_t|} \sum_{i \in \mathcal{N}_t} z_i^t. \tag{4.10}$$

Alternately, the *hinge loss* could be used in the outer-level objective. The hinge loss is a relaxed overestimate of the misclassification loss. Using this loss in the outer level effectively matches the inner- and outer-level loss functions.

$$\Theta(\mathbf{W},\mathbf{b}) = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{|\mathcal{N}_t|}\sum_{i\in\mathcal{N}_t}\max(1-y_i(\mathbf{x}_i'\mathbf{w}^t-b_t),0). \tag{4.11}$$

4.3. ℓ_1-norm regularization. Different norms can be used to regularize the inner-level problem. It is well-known that ℓ_1-norm regularization produces sparser solutions than its ℓ_2 counterpart, hence performing better feature selection if used for regularization. This can easily be incorporated into, say, the classification bilevel formulation (4.1) by replacing $\frac{1}{2}\|\mathbf{w}\|_2^2$ with $\|\mathbf{w}\|_1$, yielding the following inner-level problems:

$$\underset{\substack{-\overline{\mathbf{w}}\le\mathbf{w}^t\le\overline{\mathbf{w}}\\ b_t\in\mathbb{R}}}{\arg\min}\left\{\|\mathbf{w}\|_1+\frac{C}{|\overline{\mathcal{N}}_t|}\sum_{j\in\overline{\mathcal{N}}_t}\max(1-y_j(\mathbf{x}_j'\mathbf{w}^t-b_t),0)\right\}. \tag{4.12}$$

Since the $\|\mathbf{w}^t\|_1$ is not differentiable at 0, we introduce two new variables, $\mathbf{w}^{t,\pm}$, such that $\mathbf{w}^t=\mathbf{w}^{t,+}-\mathbf{w}^{t,-}$. If we choose to retain the box constraints as well, then the bounds, $0\le\mathbf{w}^{t,\pm}\le\overline{\mathbf{w}}$, must also hold. Since the inner-level problems are simply LPs, all the KKT conditions can be simplified to complementarity constraints,

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

$$\begin{aligned}&0\le\boldsymbol{\gamma}^{t,+}\perp\overline{\mathbf{w}}-\mathbf{w}^{t,+}\ge 0,\\ &0\le\boldsymbol{\gamma}^{t,-}\perp\overline{\mathbf{w}}-\mathbf{w}^{t,-}\ge 0,\\ &0\le\mathbf{w}^{t,+}\perp\mathbb{1}-\sum_{j\in\overline{\mathcal{N}}_t}y_j\alpha_j^t\mathbf{x}_j+\boldsymbol{\gamma}^{t,+}\ge 0,\\ &0\le\mathbf{w}^{t,-}\perp\mathbb{1}+\sum_{j\in\overline{\mathcal{N}}_t}y_j\alpha_j^t\mathbf{x}_j+\boldsymbol{\gamma}^{t,-}\ge 0,\end{aligned} \tag{4.13}$$

except for the equality constraints that are dual to b_t,

$$\sum_{j\in\overline{\mathcal{N}}_t}y_j\alpha_j^t=0. \tag{4.14}$$

If the box constraint in the inner-level problems was to be dropped, then their corresponding dual variables, $\boldsymbol{\gamma}^{\pm}$, would disappear from the complementarities above. Many other regularization schemes could also be incorporated into (3.1)) such as ones which use hinge loss combined with the so-called lasso penalty, $\mu_1\|\mathbf{w}\|_2^2+\mu_2\|\mathbf{w}\|_1$ (elastic nets, [56]), ℓ_∞-norm regularization [2, 57] or the modified support vector machine which uses $\frac{1}{2}(\|\mathbf{w}\|_2^2+b^2)$ for regularization [44]. The last regularization operator is nearly identical to classical SVMs except that the equality constraint, (4.14), vanishes from the dual (and subsequently from the first order conditions of the inner-level problems). This is useful if it is desired that all equality constraints be removed in order to apply solution techniques such as successive over-relaxation or linear programming chucking [8].

4.4. ℓ_0-norm Regularization. Let us assume that it was known *a priori* for the given n-dimensional data set, that at most n_{max} features are desired. This can be incorporated into the model by introducing the constraint, $\|\overline{\mathbf{w}}\|_0 \leq n_{\mathsf{max}}$, into the outer-level problem, where $\|\cdot\|_0$ is the so-called zero-norm[1], or the cardinality of a vector, i.e., it counts the number of non-zero elements in its argument. This constraint bounds the number of allowable features and causes the smallest feature weights to be dropped from the model. Since $\|\overline{\mathbf{w}}\|_0 = \mathbb{1}'\overline{\mathbf{w}}_\star$, the constraint can be reformulated using the $(\ )_\star$ function and its LPEC conditions, (4.6). Thus, the following complementarity constraints are added to (4.9), with $\boldsymbol{\delta}$ counting the selected features of $\overline{\mathbf{w}}$ and $\mathbf{d}$ being the multipliers to the constraints $\mathbb{1} - \boldsymbol{\delta} \geq 0$

$$\begin{aligned} &\sum_{m=1}^{n} \delta_m \leq n_{\mathsf{max}}, \\ &0 \leq \boldsymbol{\delta} \perp -\overline{\mathbf{w}} + \mathbf{d} \geq 0, \\ &0 \leq \mathbf{d} \perp \mathbb{1} - \boldsymbol{\delta} \geq 0. \end{aligned} \tag{4.15}$$

If the resulting LPEC can be solved efficiently, then the optimal value of n_{max} can be found using a line search or an scheme similar to recursive feature elimination [30], i.e., by successively decreasing n_{max} as long as the error rate continues to decrease. Alternately, the $\|\overline{\mathbf{w}}\|_0$ term can be moved into the inner-level objective as the regularization operator [7],

$$\underset{\substack{-\overline{\mathbf{w}} \leq \mathbf{w}^t \leq \overline{\mathbf{w}} \\ b_t \in \mathbb{R}}}{\arg\min} \left\{ \lambda \|\overline{\mathbf{w}}\|_0 + \frac{1-\lambda}{\ell} \sum_{j \in \overline{\mathcal{N}}_t} \max(1 - y_j(\mathbf{x}_j' \mathbf{w}^t - b_t), 0) \right\}. \tag{4.16}$$

This, however, converts the overall cross-validation problem into a *trilevel* problem, since the $(\ )_\star$ function, which is used to rewrite $\|\cdot\|_0$, adds a nested inner-level problem of its own to (4.16).

5. Kernel Bilevel Cross Validation

The SVMs 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, (4.1), though the results below can easily be generalized to other kernel methods. 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, (4.3), 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

[1]It should be noted that the zero-norm is not really a norm because, the positive homogeneity condition does not hold except for very special cases, i.e., $\|a\mathbf{w}\|_0 \neq |a|\|\mathbf{w}\|_0$, in general. However, the term has found widespread use in both the optimization and machine learning communities.

(effectively causing $\gamma^{t,\pm}$ to drop out of the program) and work with the classical SV classifier. The resulting first order conditions within each fold include the following constraints:

$$\mathbf{w}^t = \sum_{j \in \overline{\mathcal{N}}_t} y_j \alpha_j^t \mathbf{x}_j, \quad \forall t = 1, \ldots, T. \tag{5.1}$$

Now, we can eliminate $\mathbf{w}^t$ within each fold of (4.9) using (5.1) and then apply the kernel trick, i.e., the resulting linear inner-product terms, $\mathbf{x}_i'\mathbf{x}_j$, 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

$$\begin{aligned}
&\min_{\substack{C, b_t, \mathbf{z}^t, \\ \boldsymbol{\zeta}^t, \boldsymbol{\alpha}^t, \boldsymbol{\xi}^t}} \frac{1}{T} \sum_{t=1}^{T} \frac{1}{|\mathcal{N}_t|} \sum_{i \in \mathcal{N}_t} \zeta_i^t \\
&\text{s. t. } C \geq 0, \\
&\qquad \text{and for } t = 1, \ldots T, \\
&\qquad \left.\begin{aligned}
&0 \leq \zeta_i^t \perp y_i \left[\sum_{k \in \overline{\mathcal{N}}_t} y_k \alpha_k^t \kappa(\mathbf{x}_i, \mathbf{x}_k) - b_t \right] + z_i^t \geq 0, \\
&0 \leq z_i^t \perp 1 - \zeta_i^t \geq 0,
\end{aligned}\right\}, \qquad \forall i \in \mathcal{N}_t, \\
&\qquad \left.\begin{aligned}
&0 \leq \alpha_j^t \perp y_j \left[\sum_{k \in \overline{\mathcal{N}}_t} y_k \alpha_k^t \kappa(\mathbf{x}_j, \mathbf{x}_k) - b_t \right] - 1 + \xi_j^t \geq 0, \\
&0 \leq \xi_j^t \perp \frac{C}{|\overline{\mathcal{N}}_t|} - \alpha_j^t \geq 0,
\end{aligned}\right\}, \quad \forall j \in \overline{\mathcal{N}}_t, \\
&\qquad \sum_{j \in \overline{\mathcal{N}}_t} y_j \alpha_j^t = 0.
\end{aligned} \tag{5.2}$$

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 parametrized, and the kernel parameters are typically determined via cross validation. Also, unlike its linear counterpart, this model is not capable of performing feature selection.

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 parametrized kernel of the form $\kappa(\mathbf{x}_i, \mathbf{x}_k; \mathbf{p}, \mathbf{q})$. 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 $P = \operatorname{diag}(\mathbf{p})$. The parametrized versions of some commonly used kernels are shown below.

$$\begin{aligned}
&\text{Linear kernel} && \kappa(\mathbf{x}_i, \mathbf{x}_k; \mathbf{p}) = \mathbf{x}_i' P \mathbf{x}_k, \\
&\text{Polynomial kernel} && \kappa(\mathbf{x}_i, \mathbf{x}_k; \mathbf{p}, c, d) = (\mathbf{x}_i' P \mathbf{x}_k + c)^d, \\
&\text{Gaussian kernel} && \kappa(\mathbf{x}_i, \mathbf{x}_k; \mathbf{p}) = \exp\bigl(-(\mathbf{x}_i - \mathbf{x}_k)' P (\mathbf{x}_i - \mathbf{x}_k)\bigr).
\end{aligned} \tag{5.3}$$

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 parametrized kernel is a

very powerful extension to the linear model (4.9) as it is capable of picking the regularization parameters, kernel parameters and features leaving only the choice of kernel family to the user. Problem (5.2), optimized with added variables $\mathbf{p}$, $\mathbf{q}$ in the parametrized kernel, is an MPEC with non-linear complementarity constraints and in general is a very difficult problem to solve.

Alternately, taking advantage of the ability of the bilevel model to deal with a large number of hyper-parameters, the following kernel can be used:

$$\kappa(\mathbf{x}_i, \mathbf{x}_k; \beta_{i,k}, \mathbf{p}_k) = \exp\left((\mathbf{x}_i - \mathbf{x}_k)' P_k (\mathbf{x}_i - \mathbf{x}_k) big\right). \tag{5.4}$$

This is a generalization of the commonly used Gaussian kernel in that it contains a different width parameter, β_k, for each support vector which has to be determined. The resulting function becomes a Radial Basis Function Network. Kernels of this type are useful for time-series analysis and function approximations owing to the excellent fitting properties of the Gaussian.

6. Semi-supervised Learning and Transduction

We have, thus far, focused 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 unlabelled data points and a relatively small number of labelled training points. This necessitates *semi-supervised* learning, where training is performed using both the labelled and unlabelled data. If all the training data is unlabelled, 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 labelled 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 labelled training data and the unlabelled test data are both given. All available data is used to construct the decision rule in order to avoid over-fitting. The learning task in transduction is to only predict the labels for the specific test points and not for all future data. Performing transductive learning may result in improvement in generalization error bounds [54], thus reducing the number of labelled data required for good generalization.

Some additional notation is now introduced. As before, $\Omega = \{\mathbf{x}_i, y_i\}_{i=1}^{\ell}$ represents the set of labelled data, with $\ell = |\Omega|$. Let $\Psi = \{\mathbf{x}_i\}_{i=1}^{u}$ represent the unlabelled 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.

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

$$\begin{aligned} &\underset{f,\mathbf{z},\boldsymbol{\lambda}}{\text{minimize}}\ \Theta(f, \mathbf{z}; \Omega, \Psi, \boldsymbol{\lambda}) \\ &\text{subject to } \boldsymbol{\lambda} \in \Lambda, \\ &f \in \underset{f \in \mathcal{F}}{\arg\min} \left\{ \mathcal{P}(f, \boldsymbol{\lambda}) + \sum_{j \in \mathcal{N}} \mathcal{L}_l(y_j, f(\mathbf{x}_j), \boldsymbol{\lambda}) + \sum_{j \in \mathcal{M}} \mathcal{L}_u(z_j, f(\mathbf{x}_j), \boldsymbol{\lambda}) \right\}. \end{aligned} \tag{6.1}$$

In the model above, the loss functions, $\mathcal{L}_l$ and $\mathcal{L}_u$, are applied to the labelled and unlabelled 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. A more robust solution can be found by combining both types of learning by optimizing $\mathbf{z}$ 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 [34]:

$$\begin{aligned}
&\underset{C,D,\varepsilon,\mathbf{w},b,\mathbf{z}}{\text{minimize}} \sum_{i\in\mathcal{N}} |\mathbf{x}_i'\mathbf{w} - b - y_i| \\
&\text{subject to } \varepsilon, C, D \geq 0, \\
&(\mathbf{w}, b) \in \underset{(\mathbf{w},b)\in\mathbb{R}^{n+1}}{\arg\min} \left\{ \begin{array}{r} \frac{1}{2}\|\mathbf{w}\|_2^2 + \frac{C}{|\mathcal{N}|} \sum_{j\in\mathcal{N}} \max(|\mathbf{x}_j'\mathbf{w} - b - y_j| - \varepsilon, 0) \\ + \frac{D}{|\mathcal{M}|} \sum_{j\in\mathcal{M}} \max(|\mathbf{x}_j'\mathbf{w} - b - z_j| - \varepsilon, 0) \end{array} \right\}.
\end{aligned} \tag{6.2}$$

The outer-level objective is simply the mean average deviation (MAD) on all the labelled data. The inner-level objective uses both the labelled and unlabelled 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 unlabelled examples, u is far greater than the number of labelled examples, ℓ. This means that (6.2) will have a large number of outer-level variables ($\mathbf{z}$) and complementarity constraints arising from the unlabelled data points.

The model (6.2) 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

$$\begin{aligned}
&\underset{C,D,\varepsilon,\mathbf{w}^t,b_t,\mathbf{z}}{\text{minimize}} \frac{1}{T}\sum_{t=1}^{T} \frac{1}{|\mathcal{N}_t|} \sum_{i\in\mathcal{N}_t} |\mathbf{x}_i'\mathbf{w} - b - y_i| \\
&\text{subject to } \varepsilon, C, D \geq 0, \\
&\quad \text{and for } t = 1, \ldots, T, \\
&(\mathbf{w}^t, b_t) \in \underset{(\mathbf{w},b)\in\mathbb{R}^{n+1}}{\arg\min} \left\{ \begin{array}{r} \frac{1}{2}\|\mathbf{w}\|_2^2 + \frac{C}{|\mathcal{N}_t|} \sum_{j\in\mathcal{N}_t} \max(|\mathbf{x}_j'\mathbf{w} - b - y_j| - \varepsilon, 0) \\ + \frac{D}{|\mathcal{M}|} \sum_{j\in\mathcal{M}} \max(|\mathbf{x}_j'\mathbf{w} - b - z_j| - \varepsilon, 0) \end{array} \right\},
\end{aligned} \tag{6.3}$$

so that the resultant program is again a novel combination of inductive and transductive learning. Here, the unlabelled data is used to train the decision rule *for*

each fold. As there are T inner level problems, the complementarity conditions containing the unlabelled data will occur T times, though each time with a different $(\mathbf{w}^t, b_t)$ in the constraints.

6.2. 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 labelled points. We look at three loss functions that were introduced in [16] for the unlabelled points in the inner level. The first is the so-called *hard-margin* loss,

$$\mathcal{L}_u(\mathbf{w}, b) = \begin{cases} \infty, & \text{for } -1 < \mathbf{x}'\mathbf{w} - b < 1, \\ 0, & \text{otherwise.} \end{cases} \tag{6.4}$$

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:

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi},\mathbf{z}^+,\mathbf{z}^-} \ & \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{j\in\mathcal{N}} \xi_j^t \\ \text{s. t.} \quad & y_i(\mathbf{x}_i'\mathbf{w} - b) \geq 1 - \xi_i, \xi_i \geq 0, && \forall i \in \mathcal{N} \\ & \left.\begin{aligned} & -(\mathbf{x}_j'\mathbf{w} - b) \geq 1 - z_j^+, z_j^+ \geq 0, \\ & (\mathbf{x}_j'\mathbf{w} - b) \geq 1 - z_j^-, z_j^- \geq 0, \\ & z_j^+ z_j^- = 0 \end{aligned}\right\}, && \forall j \in \mathcal{M}. \end{aligned} \tag{6.5}$$

This results in a non-convex, quadratically-constrained quadratic program (QCQP) which is challenging 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,

$$\mathcal{L}_u(\mathbf{w}, b) = \max(1 - (\mathbf{x}'\mathbf{w} - b)^2, 0), \tag{6.6}$$

and the non-convex hat-loss function,

$$\mathcal{L}_u(\mathbf{w}, b) = \max(1 - |\mathbf{x}'\mathbf{w} - b|, 0). \tag{6.7}$$

These loss functions arise from the relaxing the hard constraint $z_j^+ z_j^- = 0$ in (6.5) by moving it into the inner-level objective; if the product, $z_j^+ z_j^-$, is used directly, a quadratic penalty function, (6.6), 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 unlabelled data is precisely the transductive idea proposed by Vapnik [54]. The "optimal" labels on the unlabelled data can be calculated as $\text{sign}(z_j^+ - z_j^-)$.

Finally, we can use the step function to formulate loss functions that use the number of misclassifications for both the labelled and unlabelled data sets if we

solve

$$
\begin{aligned}
&\underset{C,D,\mathbf{w},b,\boldsymbol{\zeta},\mathbf{z}}{\text{minimize}} \ \frac{1}{|\mathcal{N}|}\sum_{i\in\mathcal{N}}\zeta_i \\
&\text{subject to } C, D \geq 0, \\
&\quad \boldsymbol{\zeta} \in \underset{0\leq\boldsymbol{\zeta}\leq\mathbb{1}}{\arg\min}\left\{\sum_{i\in\mathcal{N}}\zeta_i y_i(\mathbf{x}_i'\mathbf{w}-b)\right\}, \\
&\quad \mathbf{z} \in \underset{0\leq\mathbf{z}\leq\mathbb{1}}{\arg\min}\left\{\sum_{i\in\mathcal{M}}-\mathbf{z}_i(\mathbf{x}_i'\mathbf{w}-b)\right\}, \\
&\quad (\mathbf{w},b) \in \underset{(\mathbf{w},b)\in\mathbb{R}^{n+1}}{\arg\min}\left\{\begin{array}{r}\frac{1}{2}\|\mathbf{w}\|_2^2+\frac{C}{|\mathcal{N}|}\sum_{j\in\mathcal{N}}\max(1-y_j(\mathbf{x}_j'\mathbf{w}-b),0)\\ +\frac{D}{|\mathcal{M}|}\sum_{j\in\mathcal{M}}\max(1-z_j(\mathbf{x}_j'\mathbf{w}-b),0)\end{array}\right\}.
\end{aligned}
\tag{6.8}
$$

The outer-level objective performs misclassification minimization on the labelled data, with the first inner-level problem counting the number of misclassifications. The second inner-level problem computes the labels on the unlabelled data which are used to perform learning in the third inner-level problem. As in the regression case, the problem (6.8) 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 Section 3. It is also relatively straightforward to kernelize the models discussed above as per the discussion in Section 5, as long as care is taken in dealing with the labelled and unlabelled kernels.

7. 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 [9] 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, (2.1) with added multi-task terms (and including the bias term):

$$
(\mathbf{w}^t, b_t) \in \underset{(\mathbf{w},b)\in\mathbb{R}^{n+1}}{\arg\min}\left\{\begin{array}{r}\frac{1}{2}\|\mathbf{w}\|_2^2+\frac{C}{|\overline{\mathcal{N}}_t|}\sum_{j\in\overline{\mathcal{N}}_t}\max(|\mathbf{x}_j'\mathbf{w}-y_j|-\varepsilon,0)\\ +\frac{\lambda_{\mathbf{w}}}{2}\|\mathbf{w}-\mathbf{w}^0\|_2^2+\frac{\lambda_b}{2}(b-b_0)^2\end{array}\right\}.
\tag{7.1}
$$

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 (7.1) with its corresponding KKT conditions:

$$
\begin{aligned}
0 &= (1+\lambda_{\mathbf{w}})\mathbf{w}^t - \lambda_w \mathbf{w}^0 + \sum_{i\in\overline{\mathcal{N}}_t} (\alpha_i^{t,+} - \alpha_i^{t,-})\mathbf{x}_i, \\
0 &= \lambda_b(b_t - b_0) + \sum_{i\in\overline{\mathcal{N}}_t} (\alpha_i^{t,+} - \alpha_i^{t,-}), \\
&\left.\begin{aligned}
&0 \le \xi_i^t \perp \frac{C}{|\overline{\mathcal{N}}_t|} - \alpha_i^{t,+} - \alpha_i^{t,-} \ge 0, \\
&0 \le \alpha_i^{t,+} \perp \xi_i^t + \varepsilon - \mathbf{x}_i'\mathbf{w}^t + b_t + y_i \ge 0, \\
&0 \le \alpha_i^{t,-} \perp \xi_i^t + \varepsilon + \mathbf{x}_i'\mathbf{w}^t - b_t - y_i \ge 0,
\end{aligned}\right\} \quad \forall i \in \overline{\mathcal{N}}_t.
\end{aligned} \tag{7.2}
$$

From (7), we deduce

$$
\begin{aligned}
\mathbf{w}^t &= \frac{1}{1+\lambda_{\mathbf{w}}} \left[\lambda_{\mathbf{w}} \mathbf{w}^0 - \sum_{i\in\overline{\mathcal{N}}_t} (\alpha_i^{t,+} - \alpha_i^{t,-})\mathbf{x}_i \right], \\
b_t &= b_0 - \frac{1}{\lambda_b} \sum_{i\in\overline{\mathcal{N}}_t} (\alpha_i^{t,+} - \alpha_i^{t,-}),
\end{aligned} \tag{7.3}
$$

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

$$
\sum_{i\in\overline{\mathcal{N}}_t} (\alpha_i^{t,+} - \alpha_i^{t,-}) = 0, \tag{7.4}
$$

which does not involve b_t. In the interest of kernelizing (7), we postulate that

$$
\mathbf{w}^0 \equiv \sum_{j\in\mathcal{N}} \beta_j \mathbf{x}^j, \tag{7.5}
$$

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

$$
\mathbf{w}^t \equiv \frac{1}{1+\lambda_{\mathbf{w}}} \left[\lambda_{\mathbf{w}} \sum_{j\in\mathcal{N}} \beta_j \mathbf{x}^j - \sum_{j\in\overline{\mathcal{N}}_t} (\alpha_j^{t,+} - \alpha_j^{t,-})\mathbf{x}^j \right]. \tag{7.6}
$$

This last expression can be substituted into the complementarities in (7) to give

$$
\begin{aligned}
&0 \le \xi_i^t \perp \frac{C}{|\overline{\mathcal{N}}_t|} - \alpha_i^{t,+} - \alpha_i^{t,-} \ge 0, \\
&0 \le \alpha_i^{t,+} \perp \xi_i^t + \varepsilon - \frac{1}{1+\lambda_{\mathbf{w}}} \left[\lambda_{\mathbf{w}} \sum_{j\in\mathcal{N}} \beta_j \mathbf{x}_i'\mathbf{x}_j - \sum_{j\in\overline{\mathcal{N}}_t} (\alpha_j^{t,+} - \alpha_j^{t,-})\mathbf{x}_i'\mathbf{x}_j \right] \\
&\qquad\qquad + b_t + y_i \ge 0, \\
&0 \le \alpha_i^{t,-} \perp \xi_i^t + \varepsilon + \frac{1}{1+\lambda_{\mathbf{w}}} \left[\lambda_{\mathbf{w}} \sum_{j\in\mathcal{N}} \beta_j \mathbf{x}_i'\mathbf{x}_j - \sum_{j\in\overline{\mathcal{N}}_t} (\alpha_j^{t,+} - \alpha_j^{t,-})\mathbf{x}_i'\mathbf{x}_j \right] \\
&\qquad\qquad - b_t - y_i \ge 0.
\end{aligned} \tag{7.7}
$$

The "kernel trick" can now be applied to (7.7); see Section 5 for details.

8. Optimization Methods for Bilevel Models

The bilevel and multilevel 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.

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 [11]. Furthermore, the complementarity constraints cause the feasible region of a bilevel program to lack closedness and convexity or, even possibly, be disjoint [40]. 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 [41] which can be useful, especially if it is hard to construct globally optimal solutions

8.1. Stationarity and constraint qualification for MPECs. In this subsection, we introduce some standard assumptions and definitions from MPEC theory in order to better understand the properties of MPECs at optimality. We consider bilevel programs of the type shown below, which is slightly different from the Bracken and McGill formulation, (1.4),

$$\begin{aligned} &\min_{x,y} F(x,y) \\ &\text{s. t. } G(x,y) \geq 0, \\ &\qquad y \in \left\{ \begin{array}{l} \arg\min_{y} f(x,y) \\ \text{s. t. } g_i(x,y) \geq 0, \quad \forall i = 1,\ldots,m \end{array} \right\}. \end{aligned} \tag{8.1}$$

Introducing Lagrange multipliers, $\lambda_i \geq 0$, for the inner-level constraints, (8.1) can be rewritten using either the first-order KKT conditions or a variational inequality

as follows:

$$
\begin{aligned}
&\min_{x,y} F(x,y) \\
&\text{s. t. } G(x,y) \geq 0, \\
&\quad \nabla f(x,y) - \sum_{i=1}^{m} \lambda_i \nabla g_i(x,y) = 0, \\
&\quad 0 \leq \lambda_i \perp g_i(x,y) \geq 0, \ \forall i = 1,\ldots,m
\end{aligned}
\tag{8.2}
$$

$$
\iff
\begin{aligned}
&\min_{x,y} F(x,y) \\
&\text{s. t. } G(x,y) \geq 0, \\
&\quad (u-y)'\nabla f(x,y) \geq 0, \text{ for some } y, \\
&\quad u \in \{y \mid g_i(x,y) \geq 0, \ \forall i = 1,\ldots,m\}.
\end{aligned}
$$

The two formulations above are equivalent nonlinear programs; we shall use the one with the inner-level KKT conditions. As noted above, LICQ or MFCQ, which are necessary to guarantee the existence of the multipliers, λ_i, at stationarity, fail to hold for (8.2) 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 (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 (8.1) and (8.2) 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, (8.2), we can relate it to the tightened and relaxed non-linear programs, where the first-order equality constraints have been collected into $H(x,y,\lambda)$,

$$
\begin{aligned}
&\min_{x,y} F(x,y) \text{ (tightened)} && \min_{x,y} F(x,y) \text{ (relaxed)} \\
&\text{s. t. } G(x,y) \geq 0, \ H(x,y,\lambda) = 0, && \text{s. t. } G(x,y) \geq 0, \ H(x,y,\lambda) = 0, \\
&\quad \lambda_i = 0, \qquad \forall i \in \mathcal{I}_\alpha, && \quad \lambda_i = 0, \qquad \forall i \in \mathcal{I}_\alpha, \\
&\quad g_i(x,y) = 0, \qquad \forall i \in \mathcal{I}_\gamma, && \quad \lambda_i = 0, \qquad \forall i \in \mathcal{I}_\alpha, \\
&\quad \lambda_i = 0, \ g_i(x,y) = 0, \quad \forall i \in \mathcal{I}_\beta. && \quad \lambda_i \geq 0, \ g_i(x,y) \geq 0, \quad \forall i \in \mathcal{I}_\beta.
\end{aligned}
\tag{8.3}
$$

and with the Lagrangian function,

$$
\mathcal{L}(x,y,\lambda_i,\mu,\nu,u,v) = F(x,y) - \mu G(x,y) - \nu H(x,y,\lambda) - \sum_{i=1}^{m} u_i\lambda_i - \sum_{i=1}^{m} v_i g_i(x,y),
\tag{8.4}
$$

where

$$\begin{aligned} \mathcal{I}_\alpha &:= \{i \mid \lambda_i = 0, g_i(x,y) > 0\}, \\ \mathcal{I}_\beta &:= \{i \mid \lambda_i = 0, g_i(x,y) = 0\}, \\ \mathcal{I}_\gamma &:= \{i \mid \lambda_i > 0, g_i(x,y) = 0\}. \end{aligned} \tag{8.5}$$

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 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 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

$$\begin{gathered} \nabla \mathcal{L}(x, y, \lambda_i, \mu, \nu, u, v) = 0, \\ \mu \geq 0, \qquad u_i = 0, \quad \forall i \in \mathcal{I}_\gamma, \qquad v_i = 0, \quad \forall i \in \mathcal{I}_\alpha. \end{gathered} \tag{8.6}$$

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 4 (S-stationarity). A feasible point (x^*, y^*, λ^*) is said to be *strongly or S-stationary* if the W-stationarity conditions, (8.6), 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 [50]. This discussion can be easily extended to the case where the outer-level problem may have equality constraints.

We now discuss some approaches to solving MPECs.

[2]W-stationarity concepts can be strengthened by enforcing additional constraints on the multipliers in (8.4). For example, replacing $\lambda_i g_i(x,y) = 0$ with $\min\big(\lambda_i, g_i(x,y)\big) = 0$ in (8.2) 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 [55] for more details.

8.2. 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

$$\begin{aligned} &\min_{x,y} c'x + d'y \\ &\text{s. t. } 0 \le y \perp w = Nx + My + q \ge 0, \\ &\qquad Ax + By + p \ge 0, \\ &\qquad Gx + Hy + f = 0. \end{aligned} \tag{8.7}$$

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 component wise plus function applied to some vector $\mathbf{r} \ge 0$.

8.2.1. *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

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

with relaxed, "soft" complementarity constraints of the form

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

where 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 [24, 25] or SNOPT [27], which are freely available on the NEOS server [13]. 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 was used to solve bilevel cross validation for support vector regression, (2.5), in [3] and support vector classification, (4.1), in [36] using FILTER. 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 favourably 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.

8.2.2. *Smooth approximations.* The condition, $\min(y, Nx + My + q) = 0$, can be replaced by a function $\phi(y, w)$, possibly non-smooth, such that $\phi(y, w) = 0 \equiv 0 \le y \perp w \ge 0$. The Fischer-Burmeister function [23], $\phi(y, w) = y + w - \sqrt{y^2 + w^2}$, is a non-smooth example of such a function. This function is smoothed using a parameter ε to give the smoothed Fischer-Burmeister function, $\phi(y, w) = y + w - \sqrt{y^2 + w^2 + \varepsilon^2}$. The smoothed function is everywhere differentiable and yields the

following approximation of (8.7):

$$\begin{aligned} &\min_{x,y,w} c'x + d'y \\ &\text{s. t. } w = Nx + My + q \geq 0,\ y \geq 0, \\ &\qquad Ax + By + p \geq 0, \\ &\qquad Gx + Hy + f = 0 \\ &\qquad y_i + w_i - \sqrt{y_i^2 + w_i^2 + \varepsilon_k^2} = 0, \quad \forall i = 1, \ldots, m. \end{aligned} \tag{8.8}$$

Pang and Fukushima [26] showed that for decreasing values of ε_k, the sequence of stationary points to the nonlinear program (8.8), (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 (8.8); for example, the sequential quadratic programming (SQP) algorithm [35].

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 parametrized 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

$$p(z, \alpha) = z + \frac{1}{\alpha} \log\left(1 + e^{-\alpha z}\right), \quad \alpha > 0, \tag{8.9}$$

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

8.2.3. *Exact penalty methods.* Penalty and augmented Lagrangian methods have been widely applied to solving LPECs and MPECs [33]. These methods typically require solving an unconstrained optimization problem. In contrast, exact penalty methods penalize only the complementarity constraints in the objective:

$$\begin{aligned} &\min_{x,y,w} c'x + d'y + \mu\phi(y, w) \\ &\text{s. t. } w = Nx + My + q \geq 0,\ y \geq 0, \\ &\qquad Ax + By + p \geq 0, \\ &\qquad Gx + Hy + f = 0. \end{aligned} \tag{8.10}$$

One approach to solving exact penalty formulations like (8.10) is the successive linearization algorithm, where a sequence of problems with a linearized objective,

$$c'(x - x^k) + d'(y - y^k) + \mu\big(\partial_x\phi(y^k, w^k)(x - x^k) + \partial_y\phi(y^k, w^k)(y - y^k)\big) \tag{8.11}$$

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 [41].

Alternately, the concave penalty function, $\min(y, w)$, 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 [43],

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

$$\partial_y \phi = \sum_{j=1}^{m} \begin{cases} I_j, & \text{if } y_j < w_j, \\ (1-\lambda_j) I_j + \lambda_j M_j, & \text{if } y_j = w_j, \\ M_j, & \text{if } y_j > w_j. \end{cases} \tag{8.12}$$

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

$$\min(r, s) = \arg\min_{\rho,\sigma} \{\rho r + \sigma s \mid \rho, \sigma \geq 0, \rho + \sigma = 1\}. \tag{8.13}$$

This gives a separable bilinear program [42],

$$\begin{aligned} &\min_{x,y,w} c'x + d'y + \rho' r + \sigma' s \\ &\text{s. t.} w = Nx + My + q \geq 0,\ y \geq 0, \\ &\qquad Ax + By + p \geq 0, \\ &\qquad Gx + Hy + f = 0. \end{aligned} \tag{8.14}$$

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,

$$m(z, \alpha) = -\frac{1}{\alpha} \log(1 + e^{-\alpha z}), \quad \alpha > 0. \tag{8.15}$$

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

8.3. Integer programming approaches. The connections between bilevel programs, MPECs and mixed integer programs (MIPs) are well known. It was shown in [1] that there exists a polynomial time reformulation to convert a mixed integer program to a bilevel program. Also demonstrated in [1] was an implicit reformulation of a bilevel program as a mixed integer program via MPECs. Specifically, a program with equilibrium constraints, such as (8.7), can be converted to a MIP by splitting the complementarity constraints through the introduction of integer variables, z, and a large finite constant θ.

$$\begin{aligned} &\min_{x,y,z} c'x + d'y \\ &\text{s. t. } 0 \leq Nx + My + q \leq \theta(1 - z), \\ &\qquad 0 \leq y \leq \theta z, \quad z \in \{0, 1\}^m, \\ &\qquad Ax + By + p \geq 0, \\ &\qquad Gx + Hy + f = 0. \end{aligned} \tag{8.16}$$

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 (8.16) 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 [32], wherein a parameter-free dual program of (8.16) 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.

8.4. 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 [38], interior point methods [40, 39], implicit programming [14, 40] and non-smooth methods [49].

9. Conclusions

We showed how various important machine learning problems can be cast as bilevel programs. This includes cross validation for support vector regression and classification in order to perform parameter and feature selection, kernel methods, semi-supervised learning and multi-task learning. This is certainly not an exhaustive list of machine learning problems to which bilevel programming can be applied. Noting that all the methods proposed here were nonparametric methods, an interesting avenue of further research with regard to modelling is the incorporation of parametric or generative methods based on probability models into the bilevel framework.

In our approach, the inner-level problems in the bilevel programs are replaced by their first-order KKT conditions. This yields mathematical programs with equilibrium constraints, a class of non-convex, nonlinear, and generally hard problems. Under some mild conditions, the equivalence of bilevel programs and their corresponding MPECs can be guaranteed. However, the presence of complementarity constraints in MPECs is a major theoretical and computational challenge, as the principles of nonlinear programming theory cannot be directly extended to MPECs. However, additional stationarity and constraint qualification concepts enable one to guarantee the existence of solutions for these problems. In the machine learning context, many of the theoretical considerations are not a source of difficulty as machine learning applications generally yield reasonably well-posed and well-behaved MPECs.

A major outstanding open question is the development of efficient algorithms for bilevel programs. It should also be noted that machine learning problems, particularly support vector machines, yield elegant, convex, sparse and highly structured problems; it should not be surprising that a lot of these desirable characteristics get carried over to their bilevel counterparts. While the MPECs resulting from the bilevel programs are non-convex, they are certainly very sparse and highly structured.

The structure inherent in SVMs and kernel methods makes them an attractive target for decomposition methods. Well-known machine learning methods such as

sequential minimal optimization (SMO) take this to the extreme by decomposing the problem to consider pairs of points. More recently, the method [32] has taken an important step in this direction of algorithmic development. Mangasarian showed that variables in MPECs such as misclassification minimization problems [41] are typically uncoupled, allowing the problem to be decomposed into smaller linear programs. Many of the algorithms presented in this monograph have similar structures. It has already been demonstrated that bilevel machine learning problems perform fairly well with regards to generalization error [3, 36] when solved using SQP-based methods such as FILTER. It should be noted that since the ultimate goal is to produce good generalization, the solutions found need not necessarily be highly accurate or global optimal. However, as the number of complementarities grows rapidly with increasing data set sizes, problem size becomes all but intractable for these general purpose solvers. An urgent need is to specifically exploit the structure and properties of machine learning problems to yield algorithms that are efficient and scalable. We present these models as challenges to mathematical programming researchers.

References

1. C. Audet, P. Hansen, B. Jaumard, and G. Savard, *Links between linear bilevel and mixed* $0-1$ *programming problems*, J. Optim. Theory Appl. **93** (1997), no. 2, 273–300.
2. K. Bennett and E. Bredensteiner, *Duality and geometry in* SVM *classifiers*, Proc. 7th Internat. Conf. on Machine Learning (P. Langley, ed.), Morgan Kaufmann, San Francisco, 2000, pp. 57–64.
3. K. Bennett, X. Ji, J. Hu, G. Kunapuli, and J.-S. Pang, *Model selection via bilevel optimization*, Proc. IEEE World Congress on Computational Intelligence (2006), PUBL ???, PUBL ADDR ???, Vancouver, BC, Canada, 2006, pp. 1922–1929.
4. K. Bennett and E. Parrado-Hernández *The interplay of optimization and machine learning research*, J. Mach. Learn. Res. **7** (2006), 1265–1281.
5. J. Bi, K. Bennett, M. Embrechts, C. Breneman, and M. Song, *Dimensionality reduction via sparse support vector machines*, J. Mach. Learn. Res. **3** (2003), 1229–1243.
6. J. Bracken and J. McGill, *Mathematical programs with optimization problems in the constraints*, Oper. Res. **21** (1973), 37–44.
7. P. Bradley and O. Mangasarian, *Feature selection via concave minimization and support vector machines*, Machine Learning Proc. 5th Internat. Conf. (ICML 1998) (J. Shavlik, ed.), Morgan Kaufmann, San Francisco, CA, 1998, pp. 82–90.
8. ———, *Optimization methods in massive data sets*, Handbook of Massive Datasets (J. Abello, P. M. Pardalos, and M. G. C. Resende, eds.), vol. 20, Kluwer Acad. Publ., Dordrecht, 1999.
9. R. Caruana, *Multitask learning*, Machine Learning **28** (1997), 41–75.
10. O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee, *Choosing multiple parameters for support vector machines*, Machine Learning **46** (2002), 131–159.
11. S. Chung, NP-*completeness of the linear complementarity problem*, J. Optim. Theory Appl. **60** (1989), 393–399.
12. C. Cortes and V. Vapnik, *Support-vector networks*, Machine Learning **20** (1995), 273–297.
13. J. Czyzyk, M. Mesnier, and J. More, *The* NEOS *server*, IEEE J. Comput. Sci. Engr. **5** (1998), 68–75.
14. X. Chen, and M. Fukushima, *A smoothing method for a mathematical program with* P*-matrix linear complementarity constraints*, Comput. Optim. Appl. **27** (2004), no. 3, 223–246.
15. C. Chen and O. Mangasarian, *A class of smoothing functions for nonlinear and mixed complementarity problems*, Comput. Optim. Appl. **5** (1996), 97–138.
16. A. Demiriz and K. Bennett, *Optimization approaches to semi-supervised learning*, Complementarity: Applications, Algorithms and extensions (Madison, 1999), Appl. Optim., vol. 50, Kluwer Acad. Publ., Dordrecht, 2001, pp. 121–141.

17. A. Demiriz, K. Bennett, C. Brenman, and M. Embrechts, *Support vector regression methods in cheminformatics*, Comput. Sci. Statist. **33** (2001), pp. ???–???; `http://www.galaxy.gmu.edu/interface/I01/I2001HTML`.
18. S. Dempe, *Foundations of bilevel programming*, Kluwer Acad. Publ., Dordrecht, 2002.
19. ———, *Annotated bibliography on bilevel programming and mathematical programs with equilibrium constraints*, Optimization **52** (2003), 333–359.
20. K. Duan, S. Keerthi, and A. Poo, *Evaluation of simple performance measures for tuning* SVM *hyperparameters*, Neurocomputing *51* (2003), 41–59.
21. T. Evgeniou and M. Pontil, *Regularized multi–task learning*, Proc. Internat. Conf. on Knowledge Discovery and Data Mining, Seattle, USA, 2004, pp. 109–117.
22. F. Facchinei and J.-S. Pang, *Finite-dimensional variational inequalities and complementarity problems*, Springer-Verlag, New York, 2003.
23. A. Fischer, *A special Newton-type optimization method*, Optimization **24** (1992), 269–282.
24. R. Fletcher and S. Leyffer, *Nonlinear programming without a penalty function*, Math. Programming **91** (2002), 239–270.
25. ———, *User manual for filter* SQP, University of Dundee, 1999; `http://www-unix.mcs.anl.gov/leyffer/papers/SQP_manual.pdf`.
26. M. Fukushima, Z. Luo, and J.-S. Pang, *A globally convergent sequential quadratic programming algorithm for mathematical programs with linear complementarity constraints*, Comput. Optim. Appl. **10** (1998), 5–34.
27. P. Gill, W. Murray, and M. Saunders, *User's guide for* SNOPT *version 6*: *A Fortran package for large-scale nonlinear programming*, Systems Optimization Laboratory, Stanford University, 2002; `http://www.cam.ucsd.edu/ peg/papers/sndoc6.pdf`.
28. G. Golub, M. Heath, and G. Wahba, *Generalised cross validation as a method for choosing a good ridge parameter*, Technometrics **21** (1979), 215–223.
29. I. Guyon and A. Elisseeff, *An introduction to variable and feature selection*, J. Mach. Learn. Res. **3** (2003), no. 3, 1157–1182.
30. I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, *Gene selection for cancer classification using support vector machines*, Machine Learning **46** (2002), 389–422.
31. T. Hastie and S. Rossett, *The entire regularization path for the support vector machine*, J. Mach. Learn. Res. **5** (2004), 1391–1415.
32. J. Hu, J. Mitchell, J.-S. Pang, K. Bennett, and G. Kunapuli, *On the global solution of linear programs with complementarity constraints*, Manuscript, Rensselaer Polytechnic Institute, Troy, NY, 2007.
33. X. Huang, X. Yang, and K. Teo, *Partial augmented Lagrangian method and mathematical programs with complementarity constraints*, J. Global Optim. **35** (2006), no. 2, 235–254.
34. X. Ji, *A bilevel programming approach to semi-supervised learning*, Private Communication.
35. H. Jiang and D. Ralph, *Smooth* SQP *methods for mathematical programs with nonlinear complementarity constraints*, SIAM J. Optim. **10** (2000), no. 3, 779–808.
36. G. Kunapuli, K. Bennett, J. Hu, and J.-S. Pang, *Classification model selection via bilevel programming*, Optim. Methods Softw., 2007 (to appear).
37. G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M. Jordan, *Learning the kernel matrix with semidefinite programming*, J. Mach. Learn. Res. **5** (2004), 27–72.
38. G. Lin and M. Fukushima, *Hybrid algorithms with active set identification for mathematical programs with complementarity constraints*, Technical report 2002-008, Kyoto, Japan, 2002.
39. X. Liu and J. Sun, *Generalized stationary points and a robust interior point method for mathematical programs with equilibrium constraints*, Math. Programming **101** (2004), 231–261.
40. Z. Luo, J.-S. Pang, and D. Ralph, *Mathematical programs with equilibrium constraints*, Cambridge University Press, Cambridge, England, 1996.
41. O. Mangasarian, *Misclassification minimization*, J. Global Optim. **5** (1994), 309–323.
42. ———, *The linear complementarity problem as a separable bilinear program*, J. Global Optim. **6** (1995), 153–161.
43. ———, *Solution of general linear complementarity problems via nondifferentiable concave minimization*, Acta Math. Vietnam. **22** (1997), no. 1, 199–205.
44. ———, *Successive overrelaxation for support vector machines*, IEEE Trans. Neural Networks **10** (1999), 1032–1037.

45. O. Mangasarian and D. R. Musicant, *Robust linear and support vector regression*, IEEE Trans. Pattern Analysis and Machine Intelligence **22** (2000), no. 9, 950–956.
46. M. Momma and K. Bennett, *A pattern search method for model selection of support vector regression*, Proc. 2nd SIAM Internat. Conf. on Data Mining (R. Grossman, J. Han, V. Kumar, H. Mannila, and R. Motwani, eds.), SIAM Publications, Philadelphia, PA, 2002, ppp. ???–???.
47. B. Murdukhovich, *Metric approximation and necessary optimality conditions for general classes of nonsmooth extremal problems*, Soviet Math. Dokl. **22** (1980) 526–530.
48. C. Ong, A. Smola, and R. Williamson, *Learning the kernel with hyperkernels*, J. Mach. Learn. Res. **6** (2005) 1043–1071.
49. J. Outrata, M. Kocvara, and J. Zowe, *Nonsmooth approach to optimization problems with equilibrium constraints*: *theory, applications and numerical results*, Kluwer Acad. Publ., Dordrecht, 1998.
50. H. Scheel and S. Scholtes, *Mathematical programs with complementarity conditions*: *stationarity, optimality and sensitivity*, Math. Oper. Res. **25** (2000), no. 1, 1–22.
51. J. Shawe-Taylor and N. Cristianini, *Kernel methods for pattern analysis*, Cambridge University Press, Cambridge, 2004.
52. B. Schölkopf, R. Williamson, A. Smola, J. Shawe-Taylor, and J. Platt, *Support vector method for novelty detection*, Adv. Neural Inf. Process. Syst. **12** (2000), 582–588.
53. H. Van Stackelberg, *The theory of market economy*, Oxford University Press, Oxford, 1952.
54. V. Vapnik, *The nature of statistical learning theory*, 2nd ed., Springer-Verlag, New York, 2000.
55. J. Ye, *Necessary and sufficient optimality conditions for mathematical programs with equilibrium conditions*, J. Math. Anal. Appl. **307** (2005), 350–369.
56. H. Zou and T. Hastie, *Regularization and variable selection via the elastic net*, J. Roy. Statist. Soc. Ser. B **67**, no. 2 (2005), 301–320.
57. H. Zou and M. Yuan, *The F_∞-norm support vector machine*, Statist. Sinica (to appear); `http://www2.isye.gatech.edu/ myuan/papers/fsvm.pdf`.

Department of Mathematical Sciences, Rensselaer Polytechnic Institute, 110 8th Street Troy NY 12180, USA.

E-mail address, G. Kunapuli: `kunapg@rpi.edu`
E-mail address, K. P. Bennett: `bennek@rpi.edu`
E-mail address, J. Hu: `huj@rpi.edu`
E-mail address, J.-S. Pang: `pangj@rpi.edu`