Bilevel Optimization and Machine Learning

Kristin P. Bennett[1], Gautam Kunapuli[1], Jing Hu[1], and Jong-Shi Pang[2]

[1] Dept. of Mathematical Sciences, Rensselaer Polytechnic Institute*, Troy, NY, USA
{bennek, kunapg, huj}@rpi.edu
[2] Dept. of Industrial and Enterprise Systems Engineering, University of Illinois at Urbana Champaign, Urbana Champaign, IL, USA
jspang@uiuc.edu

Abstract. We examine the interplay of optimization and machine learning. Great progress has been made in machine learning by cleverly reducing machine learning problems to convex optimization problems with one or more hyper-parameters. The availability of powerful convex-programming theory and algorithms has enabled a flood of new research in machine learning models and methods. But many of the steps necessary for successful machine learning models fall outside of the convex machine learning paradigm. Thus we now propose framing machine learning problems as Stackelberg games. The resulting bilevel optimization problem allows for efficient systematic search of large numbers of hyper-parameters. We discuss recent progress in solving these bilevel problems and the many interesting optimization challenges that remain. Finally, we investigate the intriguing possibility of novel machine learning models enabled by bilevel programming.

1 Introduction

Convex optimization now forms a core tool in state-of-the-art machine learning. Convex optimization methods such as Support Vector Machines (SVM) and kernel methods have been applied with great success. For a learning task, such as regression, classification, ranking, and novelty detection, the modeler selects a convex loss and regularization functions suitable for the given task and optimizes for a given data set using powerful robust convex programming methods such as linear, quadratic, or semi-definite programming. But the many papers reporting the success of such methods frequently gloss over the critical choices that go into making a successful model. For example, as part of model selection, the modeler must select which variables to include, which data points to use, and how to set the possibly many model parameters. The machine learning problem is reduced to a convex optimization problem only because the boundary of what is considered to be part of the method is drawn very narrowly. Our goal here is to expand the mathematical programming models to more fully incorporate the

* This work was supported in part by the Office of Naval Research under grant no. N00014-06-1-0014. The authors are grateful to Professor Olvi Mangasarian for his suggestions on the penalty approach.

J.M. Zurada et al. (Eds.): WCCI 2008 Plenary/Invited Lectures, LNCS 5050, pp. 25–47, 2008.

entire machine learning process. Here we will examine the bilevel approach first developed in [1].

Consider support vector regression (SVR). In SVR, we wish to compute a linear regression function that maps the input, $\mathbf{x} \in R^n$, to a response $y \in R$, given a training set of (input, output) pairs, $\{(\mathbf{x}_i, y_i), i = 1 \ldots \ell\}$. To accomplish this, we must select hyper-parameters including the two parameters in the SVR objective and the features or variables that should be used in the model. Once these are selected, the learned function corresponds to the optimal solution of a quadratic program. The most commonly used and widely accepted method for selecting these hyper-parameters is still cross validation (CV).

In CV, the hyper-parameters are selected to minimize some estimate of the out-of-sample generalization error. A typical method would define a grid over the hyper-parameters of interest, and then do 10-fold cross validation for each of the grid values. The inefficiencies and expense of such a grid-search cross-validation approach effectively limit the desirable number of hyper-parameters in a model, due to the combinatorial explosion of grid points in high dimensions.

Here, we examine how model selection using out-of-sample testing can be treated as a Stackelberg game in which the leader sets the parameters to minimize the out-of-sample error, and the followers optimize the in-sample errors for each fold given the parameters. Model selection using out-of-sample testing can then be posed as an optimization problem, albeit with an "inner" and an "outer" objective. The main idea of the approach is as follows: the data is partitioned or bootstrapped into training and test sets. We seek a set of hyper-parameters such that when the optimal training problem is solved for each training set, the loss over the test sets is minimized. The resulting optimization problem is a bilevel program. Each learning function is optimized in its corresponding training problem with fixed hyper-parameters—this is the inner (or lower-level) optimization problem. The overall testing objective is minimized—this is the outer (or upper-level) optimization problem.

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

Our successful bilevel programming approaches offer several fundamental advantages over prior approaches. First, recent advances in bilevel programming in

the optimization community permit the systematic treatment of models based on popular loss functions used for SVM and kernel methods with many hyperparameters. In addition to the ability to simultaneously optimize many hyperparameters, the bilevel programming approach offers a broad framework in which a wide class of novel machine learning algorithms can be developed. Amenable problems with many parameters are pervasive in data analysis. The bilevel programming approach can be used to address feature selection, kernel construction, semi-supervised learning, and models with missing data.

This paper illustrates the bilevel approach applied to support vector regression. Additional information can be found in [1]. Discussion of the extensions of the bilevel model selection method to other problems can be found in [2].

2 Bilevel Optimization

First, we briefly review bilevel optimization. Bilevel optimization problems are a class of constrained optimization problems whose constraints contain a *lower-level* optimization problem that is parameterized by a multi-dimensional design variable. In operations research literature, the class of bilevel optimization problems was introduced in the early 1970s by Bracken and McGill [3]. These problems are closely related to the economic problem of the Stackelberg game, whose origin predates the work of Bracken and McGill. In the late 1980s, bilevel programming was given a renewed study in the extended framework of a *mathematical program with equilibrium constraints* (MPEC) in [4], which is an extension of a bilevel program with the optimization constraint replaced by a finite-dimensional variational inequality [5].

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 by Luo, Pang and Ralph [4], which was followed by two related monographs [6,7]. During the past decade, there has been an explosion of research on these optimization problems. See the annotated bibliography [8], which contains many references. In general, bilevel programs/MPECs provide a powerful computational framework for dealing with parameter identification problems in an optimization setting. As such, they offer a novel paradigm for dealing with the model selection problem described in the last section. Instead of describing a bilevel optimization problem in its full generality, we focus our discussion on its application to CV for model selection.

3 A Bilevel Support-Vector Regression Model

We focus on a bilevel support-vector regression (SVR) problem and use it to illustrate the kind of problems that the bilevel approach can treat. Specifically, suppose that the regression data are described by the ℓ points $\Omega := \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell)\}$ in the Euclidean space $\Re^{n+1}$ for some positive integers

ℓ and n. Consider the regression problem of finding a function $f^* : \Re^n \rightarrow \Re$ among a given class that minimizes the regularized risk functional

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

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

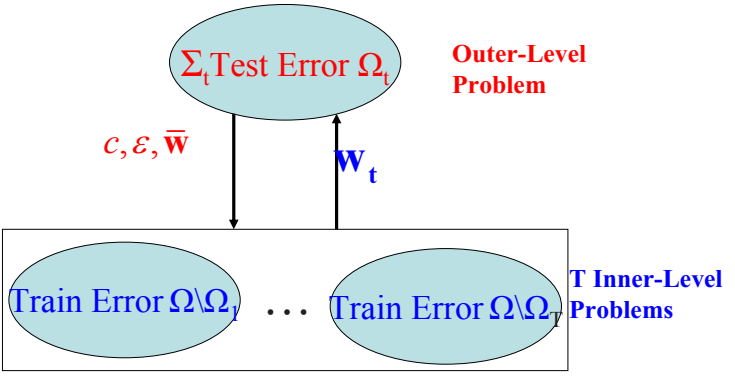

Fig. 1. SVR Model Selection as a Stackelberg Game

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

is to find the parameters ε, C, and $\mathbf{w}^t$ for $t = 1, \cdots, T$, and also the bounds $\underline{\mathbf{w}}$ and $\overline{\mathbf{w}}$ in order to

$$\begin{array}{ll} \underset{C,\varepsilon,\mathbf{w}^t,\underline{\mathbf{w}},\overline{\mathbf{w}}}{\text{minimize}} & \frac{1}{T}\sum_{t=1}^{T}\frac{1}{\mid \mathcal{N}_t \mid}\sum_{i\in\mathcal{N}_t} \mid \mathbf{x}_i'\mathbf{w}^t - y_i \mid \\ \text{subject to} & \varepsilon, C, \geq 0, \quad \underline{\mathbf{w}} \leq \overline{\mathbf{w}}, \end{array} \tag{1}$$

$$\text{and for} \quad t = 1, \ldots, T, \quad \mathbf{w}^t \in \underset{\underline{\mathbf{w}} \leq \mathbf{w} \leq \overline{\mathbf{w}}}{\arg\min} \left\{ C \sum_{j\in\overline{\mathcal{N}}_t} \max(|\mathbf{x}_j'\mathbf{w} - y_j| - \varepsilon, 0) + \frac{1}{2} \parallel \mathbf{w} \parallel_2^2 \right\}, \tag{2}$$

where the argmin in the last constraint denotes the set of optimal solutions to the convex optimization problem (2) in the variable $\mathbf{w}$ for given hyper-parameters ε, C, $\mathbf{w}_0$, $\underline{\mathbf{w}}$, and $\overline{\mathbf{w}}$. Problem 1 is called the first-level or outer-level problem. Problem (2) is referred to as the the second-level or inner-level problem. The bilevel program is equivalent to the Stackelberg game shown in figure 1. The bilevel programming approach has no difficulty handling the additional hyper-parameters and other convex constraints (such as prescribed upper bounds on these parameters) because it is based on constrained optimization methodology.

The parameters, $\underline{\mathbf{w}}$ and $\overline{\mathbf{w}}$, are related to feature selection and regularization. The bound constraints $\underline{\mathbf{w}} \leq \mathbf{w} \leq \overline{\mathbf{w}}$ enforce the fact that the weights on each descriptor must fall in a range for all of the cross-validated solutions. This effectively constrains the capacity of each of the functions, leading to an increased likelihood of improving the generalization performance. It also forces all the subsets to use the same descriptors, a form of variable selection. This effect can be enhanced by adopting the one-norm, which forces $\mathbf{w}$ to be sparse. The box constraints will ensure that consistent but not necessarily identical sets will be used across the folds. This represents a fundamentally new way to do feature selection, embedding it within cross validation for model selection.

Note that the loss functions used in the first level and second level—to measure errors—need not match. For the inner-level optimization, we adopt the ε-insensitive loss function because it produces robust solutions that are sparse in the dual space. But typically, ε-insensitive loss functions are not employed in the outer cross-validation objective; so here we use mean absolute deviation (as an example). Variations of the bilevel program (1) abound, and these can all be treated by the general technique described next, suitably extended/modified/specialized to handle the particular formulations. For instance, we may want to impose some restrictions on the bounds $\underline{\mathbf{w}}$ and $\overline{\mathbf{w}}$ to reflect some *a priori* knowledge on the desired support vector $\mathbf{w}$. In particular, we use $-\underline{\mathbf{w}} = \overline{\mathbf{w}} \geq 0$ in Section 5 to restrict the search for the weights to square boxes that are symmetric with respect to the origin. Similarly, to facilitate comparison with grid search, we restrict C and ε to be within prescribed upper bounds.

3.1 Bilevel Problems as MPECs

The bilevel optimization problem (1) determines all of the model parameters via the minimization of the outer objective function. Collecting all the weight

vectors across the folds, $\mathbf{w}^t$, column-wise into the matrix W for compactness, the cross-validation error measured as mean average deviation across all the folds is

$$\Theta(W) = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{|\,\mathcal{N}_t\,|}\sum_{i\in\mathcal{N}_t}|\,\mathbf{x}_i'\mathbf{w}^t - y_i\,|, \tag{3}$$

and is subject to the simple restrictions on these parameters, and most importantly, to the additional inner-level optimality requirement of each $\mathbf{w}^t$ for $t = 1,\ldots,T$. To solve (1), we rewrite the inner-level optimization problem (2) by introducing additional slack variables, $\boldsymbol{\xi}^t \geq 0$ within the t-th fold as follows: for given ε, C, $\underline{\mathbf{w}}$, and $\overline{\mathbf{w}}$,

$$\begin{array}{ll}\underset{\mathbf{w}^t,\,\boldsymbol{\xi}^t}{\text{minimize}} & C\displaystyle\sum_{j\in\overline{\mathcal{N}}_t}\xi_j^t + \frac{1}{2}\,\|\,\mathbf{w}^t\,\|_2^2 \\ \text{subject to} & \underline{\mathbf{w}} \leq \mathbf{w}^t \leq \overline{\mathbf{w}}, \\ & \left.\begin{array}{l}\xi_j^t \geq \mathbf{x}_j'\mathbf{w}^t - y_j - \varepsilon \\ \xi_j^t \geq y_j - \mathbf{x}_j'\mathbf{w}^t - \varepsilon \\ \xi_j^t \geq 0\end{array}\right\} \quad j \in \overline{\mathcal{N}}_t,\end{array} \tag{4}$$

which is easily seen to be a convex quadratic program in the variables $\mathbf{w}^t$ and $\boldsymbol{\xi}^t$. By letting $\boldsymbol{\gamma}^{t,\pm}$ be the multipliers of the bound constraints, $\underline{\mathbf{w}} \leq \mathbf{w} \leq \overline{\mathbf{w}}$, respectively, and $\alpha_j^{t,\pm}$ be the multipliers of the constraints $\xi_j^t \geq \mathbf{x}_j'\mathbf{w}^t - y_j - \varepsilon$ and $\xi_j^t \geq y_j - \mathbf{x}_j'\mathbf{w}^t - \varepsilon$, respectively, we obtain the Karush-Tucker-Tucker optimality conditions of (4) as the following linear complementarity problem in the variables $\mathbf{w}^t$, $\boldsymbol{\gamma}^{t,\pm}$, $\boldsymbol{\alpha}_j^{t,\pm}$, and $\boldsymbol{\xi}_j^t$:

$$\begin{array}{l}0 \leq \boldsymbol{\gamma}^{t,-} \perp \mathbf{w}^t - \underline{\mathbf{w}} \geq 0, \\ 0 \leq \boldsymbol{\gamma}^{t,+} \perp \overline{\mathbf{w}} - \mathbf{w}^t \geq 0, \\ \left.\begin{array}{ll}0 \leq \alpha_j^{t,-} \perp \mathbf{x}_j'\mathbf{w}^t - y_j + \varepsilon + \xi_j^t \geq 0 \\ 0 \leq \alpha_j^{t,+} \perp y_j - \mathbf{x}_j'\mathbf{w}^t + \varepsilon + \xi_j^t \geq 0 \\ 0 \leq \xi_j^t \quad \perp C - \alpha_j^{t,+} - \alpha_j^{t,-} \geq 0\end{array}\right\} \forall j \in \overline{\mathcal{N}}_t, \\ 0 = \mathbf{w}^t + \displaystyle\sum_{j\in\overline{\mathcal{N}}_t}(\alpha_j^{t,+} - \alpha_j^{t,-})\mathbf{x}_j + \boldsymbol{\gamma}^{t,+} - \boldsymbol{\gamma}^{t,-},\end{array} \tag{5}$$

where $a \perp b$ means $a'b = 0$. The orthogonality conditions in (5) express the well-known complementary slackness properties in the optimality conditions of the inner-level (parametric) quadratic program. All the conditions (5) represent the Karush-Kuhn-Tucker conditions. The overall two-level regression problem is therefore

$$
\begin{aligned}
\text{minimize} \quad & \frac{1}{T}\sum_{t=1}^{T}\frac{1}{|\mathcal{N}_t|}\sum_{i\in\mathcal{N}_t} z_i^t \\
\text{subject to } & \varepsilon, C, \geq 0, \quad \underline{\mathbf{w}} \leq \overline{\mathbf{w}}, \\
\text{and} \quad & \text{for all } t = 1, \ldots, T \\
& -z_i^t \leq \mathbf{x}_i'\mathbf{w}^t - y_i \leq z_i^t, \qquad \forall\, i \in \mathcal{N}_t, \\
& \left.\begin{aligned}
& 0 \leq \alpha_j^{t,-} \perp \mathbf{x}_j'\mathbf{w}^t - y_j + \varepsilon + \xi_j^t \geq 0 \\
& 0 \leq \alpha_j^{t,+} \perp y_j - \mathbf{x}_j'\mathbf{w}^t + \varepsilon + \xi_j^t \geq 0 \\
& 0 \leq \xi_j^t \quad \perp C - \alpha_j^{t,+} - \alpha_j^{t,-} \geq 0
\end{aligned}\right\} \forall j \in \overline{\mathcal{N}}_t, \\
& 0 \leq \boldsymbol{\gamma}^{t,-} \perp \mathbf{w}^t - \underline{\mathbf{w}} \geq 0, \\
& 0 \leq \boldsymbol{\gamma}^{t,+} \perp \overline{\mathbf{w}} - \mathbf{w}^t \geq 0, \\
& 0 = \mathbf{w}^t + \sum_{j\in\overline{\mathcal{N}}_t} (\alpha_j^{t,+} - \alpha_j^{t,-})\mathbf{x}_j + \boldsymbol{\gamma}^{t,+} - \boldsymbol{\gamma}^{t,-}.
\end{aligned}
\tag{6}
$$

The most noteworthy feature of the above optimization problem is the complementarity conditions in the constraints, making the problem an instance of a linear program with linear complementarity constraints (sometimes called an LPEC). The discussion in the remainder of this paper focuses on this case.

4 Alternative Bilevel Optimization Methods

The bilevel cross-validation model described above searches the continuous domain of hyper-parameters as opposed to classical cross validation via grid search, which relies on the discretization of the domain. In this section, we describe two alternative methods for solving the model. We also describe the details of the classical grid search approach.

The difficulty in solving the LPEC reformulation (6) of the bilevel optimization problem (1) stems from the linear complementarity constraints formed from the optimality conditions of the inner problem (5); all of the other constraints and the objective are linear. It is well recognized that a straightforward solution using the LPEC formulation is not appropriate because of the complementarity constraints, which give rise to both theoretical and computational anomalies that require special attention. Among various proposals to deal with these constraints, two are particularly effective for finding a local solution: one is to relax the complementarity constraints and retain the relaxations in the constraints. The other proposal is via a penalty approach that allows the violation of these constraints but penalizes the violation by adding a penalty term in the objective function of (6). There are extensive studies of both treatments, including detailed convergence analyses and numerical experiments on realistic applications and random problems. Some references are [9,10,11,6] and [4]. In this work, we experiment with both approaches.

4.1 A Relaxed NLP Reformulation

Exploiting the LPEC structure, the first solution method that is implemented in our experiments for solving (6) employs a relaxation of the complementarity constraint. In the relaxed complementarity formulation, we let $\mathbf{tol} > 0$ be a prescribed tolerance of the complementarity conditions. Consider the relaxed formulation of (6):

$$
\begin{aligned}
&\text{minimize} \quad \frac{1}{T}\sum_{t=1}^{T}\frac{1}{|\,\mathcal{N}_t\,|}\sum_{i\in\Omega_t} z_i^t \\
&\text{subject to } \varepsilon, C \geq 0, \quad \underline{\mathbf{w}} \leq \overline{\mathbf{w}}, \\
&\text{and} \qquad \text{for all } t = 1,\ldots,T \\
&\qquad -z_i^t \leq \mathbf{x}_i'\mathbf{w}^t - y_i \leq z_i^t, \qquad\qquad \forall\, i \in \mathcal{N}_t \\
&\qquad \left.\begin{array}{l} 0 \leq \alpha_j^{t,-} \perp_{\mathbf{tol}} \mathbf{x}_j'\mathbf{w}^t - y_j + \varepsilon + \xi_j^t \geq 0 \\ 0 \leq \alpha_j^{t,+} \perp_{\mathbf{tol}} y_j - \mathbf{x}_j'\mathbf{w}^t + \varepsilon + \xi_j^t \geq 0 \\ 0 \leq \xi_j^t \quad \perp_{\mathbf{tol}} C - \alpha_j^{t,+} - \alpha_j^{t,-} \geq 0 \end{array}\right\} \forall j \in \overline{\mathcal{N}}_t \\
&\qquad 0 \leq \boldsymbol{\gamma}^{t,-} \perp_{\mathbf{tol}} \mathbf{w}^t - \underline{\mathbf{w}} \geq 0, \\
&\qquad 0 \leq \boldsymbol{\gamma}^{t,+} \perp_{\mathbf{tol}} \overline{\mathbf{w}} - \mathbf{w}^t \geq 0, \\
&\qquad 0 = \mathbf{w}^t + \sum_{j\in\overline{\mathcal{N}}_t} (\alpha_j^{t,+} - \alpha_j^{t,-})\mathbf{x}_j + \boldsymbol{\gamma}^{t,+} - \boldsymbol{\gamma}^{t,-},
\end{aligned}
\tag{7}
$$

where $a \perp_{\mathbf{tol}} b$ means $a'b \leq \mathbf{tol}$. The latter formulation constitutes the *relaxed bilevel support-vector regression problem* that we employ to determine the hyper-parameters C, ε, $\underline{\mathbf{w}}$ and $\overline{\mathbf{w}}$; the computed parameters are then used to define the desired support-vector model for data analysis.

The relaxed complementary slackness is a novel feature that aims at enlarging the search region of the desired regression model; the relaxation corresponds to *inexact cross validation* whose accuracy is dictated by the prescribed scalar, **tol**. This reaffirms an advantage of the bilevel approach mentioned earlier, namely, it adds flexibility to the model selection process by allowing early termination of cross validation, and yet not sacrificing the quality of the out-of-sample errors.

The above NLP remains a non-convex optimization problem; thus, finding a global optimal solution is hard, but the state-of-the-art general-purpose NLP solvers such as FILTER (see [12] and [13]) and SNOPT (see [14]) are capable of computing good-quality feasible solutions. These solvers are available on the NEOS server – an internet server that allows remote users to utilize professionally implemented state-of-the-art optimization algorithms. To solve a given problem, the user first specifies the problem in an algebraic language, such as AMPL or GAMS, and submits the code as a job to NEOS. Upon receipt, NEOS assigns a number and password to the job, and places it in a queue. The remote solver unpacks, processes the problem, and sends the results back to the user.

The nonlinear programming solver, FILTER, was chosen to solve our problems. We also experimented with SNOPT but as reported in [1], we found FILTER to work better overall. FILTER is a sequential quadratic programming (SQP) based method, which is a Newton-type method for solving problems with nonlinear objectives and nonlinear constraints. The method solves a sequence of approximate convex quadratic programming subproblems. FILTER implements a SQP algorithm using a trust-region approach with a "filter" to enforce global convergence [12]. It terminates either when a Karush-Kuhn-Tucker point is found within a specified tolerance or no further step can be processed (possibly due to the infeasibility of a subproblem).

4.2 Penalty Reformulation

Another approach to solving the problem (6) is the penalty reformulation. Penalty and augmented Lagrangian methods have been widely applied to solving LPECs and MPECs, for instance, by [15]. These methods typically require solving an unconstrained optimization problem. In contrast, penalty methods penalize only the complementarity constraints in the objective by means of a penalty function.

Consider the LPEC, (6), resulting from the reformulation of the bilevel regression problem. Define S_t, for $t = 1, \ldots, T$, to be the constraint set within the t-th fold, without the complementarity constraints:

$$S_t := \left\{ z^t, \boldsymbol{\alpha}^{t,\pm}, \boldsymbol{\xi}^t, \boldsymbol{\gamma}^{t,\pm}, \mathbf{r}^t, s^t \;\middle|\; \begin{array}{l} -z_i^t \le \mathbf{x}_i'\mathbf{w}^t - y_i \le z_i^t, \quad \forall\, i \in \mathcal{N}_t, \\ \left.\begin{array}{l} \mathbf{x}_j'\mathbf{w}^t - y_j + \varepsilon + \xi_j^t \ge 0 \\ y_j - \mathbf{x}_j'\mathbf{w}^t + \varepsilon + \xi_j^t \ge 0 \\ C - \alpha_j^{t,+} - \alpha_j^{t,-} \ge 0 \end{array}\right\} \forall j \in \overline{\mathcal{N}}_t, \\ \underline{\mathbf{w}} \le \mathbf{w}^t \le \overline{\mathbf{w}}, \\ 0 = \mathbf{w}^t + \sum_{j \in \overline{\mathcal{N}}_t} (\alpha_j^{t,+} - \alpha_j^{t,-})\mathbf{x}_j + \boldsymbol{\gamma}^{t,+} - \boldsymbol{\gamma}^{t,-}, \\ \mathbf{w}^t = \mathbf{r}^t - \mathbb{1}\, s^t, \\ z^t, \boldsymbol{\alpha}^{t,\pm}, \boldsymbol{\xi}^t, \boldsymbol{\gamma}^{t,\pm}, \mathbf{r}^t, s^t \ge 0. \end{array} \right\}, \tag{8}$$

where we rewrite the weight vector, $\mathbf{w}^t$, within each fold as $\mathbf{w}^t = \mathbf{r}^t - \mathbb{1}\, s^t$, with $\mathbf{r}^t$, $s^t \ge 0$ and $\mathbb{1}$ denotes a vector of ones of appropriate dimension. Also, let S_0 be defined as the set of constraints on the outer-level variables:

$$S_0 := \left\{ \begin{array}{c} C, \varepsilon, \\ \overline{\mathbf{w}}, \underline{\mathbf{w}}, \mathbf{w}_0 \end{array} \;\middle|\; \begin{array}{c} C, \varepsilon, \mathbf{w}_0, \overline{\mathbf{w}}, \underline{\mathbf{w}} \ge 0 \\ \underline{\mathbf{w}} \le \overline{\mathbf{w}} \end{array} \right\}. \tag{9}$$

Then, the overall constraint set for the LPEC (6), without the complementarity constraints is defined as $S_{\mathsf{LP}} := \bigcup_{t=0}^{T} S_t$. Let all the variables in (8) and (9) be collected into the vector $\boldsymbol{\zeta} \ge 0$.

In the penalty reformulation, all the complementarity constraints of the form $a \perp b$ in (6) are moved into the objective via the penalty function, $\phi(a, b)$. This effectively converts the LPEC (6) into a penalty problem of minimizing some, possibly non-smooth, objective function on a *polyhedral set*. Typical penalty functions include the differentiable quadratic penalty term, $\phi(a, b) = a'b$, and the non-smooth piecewise-linear penalty term, $\phi(a, b) = \min(a, b)$. In this paper, we consider the quadratic penalty. The penalty term, which is a product of the complementarity terms is

$$\phi(\boldsymbol{\zeta}) = \sum_{t=1}^{T} \left(\begin{array}{l} \overbrace{\frac{1}{2} \| \mathbf{w}^t \|_2^2 + C \sum_{j \in \overline{\mathcal{N}}_t} \xi_j^t}^{\Theta_p^t} \\ \quad + \frac{1}{2} \sum_{i \in \overline{\mathcal{N}}_t} \sum_{j \in \overline{\mathcal{N}}_t} (\alpha_i^{t,+} - \alpha_i^{t,-})(\alpha_j^{t,+} - \alpha_j^{t,-}) \mathbf{x}_i' \mathbf{x}_j \\ \quad + \varepsilon \sum_{j \in \overline{\mathcal{N}}_t} (\alpha_j^{t,+} + \alpha_j^{t,-}) + \sum_{j \in \overline{\mathcal{N}}_t} y_j \, (\alpha_j^{t,+} - \alpha_j^{t,-}) \\ \quad - \overline{\mathbf{w}}' \boldsymbol{\gamma}^{t,+} + \underline{\mathbf{w}}' \boldsymbol{\gamma}^{t,-} \\ \underbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}_{-\Theta_d^t} \end{array} \right). \tag{10}$$

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

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

where $\boldsymbol{\zeta}_p^t \equiv (\mathbf{w}^t, \boldsymbol{\xi}^t)$, the vector of primal variables in the t-th primal problem and $\boldsymbol{\zeta}_d^t \equiv (\boldsymbol{\alpha}^{t,\pm}, \boldsymbol{\gamma}^{t,\pm})$, the vector of dual variables in the t-th dual problem. However, the penalty function also contains the hyper-parameters, C, ε and $\overline{\mathbf{w}}$ as variables, rendering $\phi(\boldsymbol{\zeta})$ non-convex. Recalling that the linear cross-validation objective was denoted by Θ, we define the penalized objective: $P(\boldsymbol{\zeta}; \mu) = \Theta(\boldsymbol{\zeta}) + \mu \, \phi(\boldsymbol{\zeta})$, and the penalized problem, $PF(\mu)$, is

$$\begin{array}{ll} \min\limits_{\boldsymbol{\zeta}} & P(\boldsymbol{\zeta}; \mu) \\ \text{subject to } \boldsymbol{\zeta} \in S_{\mathsf{LP}}. & \end{array} \tag{11}$$

This penalized problem has some very nice properties that have been extensively studied. First, we know that finite values of μ can be used, since local solutions of LPEC, as defined by strong stationarity, correspond to stationarity points of $PF(\mu)$. The point $\boldsymbol{\zeta}^*$ is a stationary point of $PF(\mu)$ if and only if there exists a Lagrangian multiplier vector $\boldsymbol{\rho}^*$, such that $(\boldsymbol{\zeta}^*, \boldsymbol{\rho}^*)$ is a KKT point of $PF(\mu)$. In general, KKT points do not exist for LPECs. An alternative local

optimality condition, strong stationarity of the LPEC, means that $\boldsymbol{\zeta}^*$ solves an LP formed by fixing the LPEC complementarity conditions appropriately. See Definition 2.2. [9] for precise details on strong stationarity. Finiteness ensures that the penalty parameter can be set to reasonable values, contrasting with other approaches in which the penalty problem only solve the original problem in the limit.

Theorem 1 (Finite penalty parameter). *[[9], Theorem 5.2] Suppose that $\boldsymbol{\zeta}^*$ is a strongly stationary point of (6), then for all μ sufficiently large, there exists a Lagrangian multiplier vector $\boldsymbol{\rho}^*$, such that $(\boldsymbol{\zeta}^*, \boldsymbol{\rho}^*)$ is a KKT point of $PF(\mu)$ (11).*

It is perhaps not surprising to note that the zero penalty corresponds to a point where the primal and dual objectives are equal in (4.2). These strongly stationary solutions correspond to solutions of (11) with $\phi(\boldsymbol{\zeta}) = 0$, i.e., a zero penalty. The quadratic program, $PF(\mu)$, is non-convex, since the penalty term is not positive definite. Continuous optimization algorithms will not necessarily find a global solution of $PF(\mu)$. But we do know know that local solutions of $PF(\mu)$ that are feasible for the LPEC are also local optimal for the LPEC.

Theorem 2 (Complementary $PF(\mu)$ solution solves LPEC). *[[9], Theorem 5.2] Suppose $\boldsymbol{\zeta}^*$ is a stationary point of $PF(\mu)$ (11) and $\phi(\boldsymbol{\zeta}^*) = 0$. Then $\boldsymbol{\zeta}^*$ is a strongly stationary for (6).*

One approach to solving exact penalty formulations like (11) is the successive linearization algorithm, where a sequence of problems with a linearized objective,

$$\Theta(\boldsymbol{\zeta} - \boldsymbol{\zeta}^k) + \mu \nabla\phi(\boldsymbol{\zeta}^k)'(\boldsymbol{\zeta} - \boldsymbol{\zeta}^k), \tag{12}$$

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

4.3 Successive Linearization Algorithm for Model Selection

The QP, (11), can be solved using the Frank-Wolfe method of [10] which simply involves solving a sequence of LPs until either a global minimum or some locally stationary solution of (6) is reached. In practice, a sufficiently large value of μ will lead to the penalty term vanishing from the penalized objective, $P(\boldsymbol{\zeta}^*; \mu)$. In such cases, the locally optimal solution to (11) will also be feasible and locally optimal to the LPEC (6).

Algorithm 1 gives the details of SLAMS. In Step 2, the notation *arg vertex min* indicates that $\boldsymbol{\zeta}^k$ is a vertex solution of the LP in Step 2. The step size in Step 4 has a simple closed form solution since a quadratic objective subject to bounds constraints is minimized. The objective has the form $f(\lambda) = a\lambda^2 + b\lambda$, so the optimal solution is either 0, 1 or $\frac{-b}{2a}$, depending on which value yields the smallest objective. SLAMS converges to a solution of the finite penalty problem (11). SLAMS is a special case of the Frank-Wolfe algorithm and a convergence

Algorithm 1. Successive linearization algorithm for model selection

Fix $\mu > 0$.

1. *Initialization*:
   Start with an initial point, $\boldsymbol{\zeta}^0 \in S_{\mathsf{LP}}$.
2. *Solve Linearized Problem*:
   Generate an intermediate iterate, $\bar{\boldsymbol{\zeta}}^k$, from the previous iterate, $\boldsymbol{\zeta}^k$, by solving the linearized penalty problem, $\bar{\boldsymbol{\zeta}}^k \in \arg \underset{\boldsymbol{\zeta} \in S_{\mathsf{LP}}}{\text{vertex min}} \nabla_{\boldsymbol{\zeta}} P(\boldsymbol{\zeta}^k; \mu)' (\boldsymbol{\zeta} - \boldsymbol{\zeta}^k)$.
3. *Termination Condition*:
   Stop if the minimum principle holds, i.e., if $\nabla_{\boldsymbol{\zeta}} P(\boldsymbol{\zeta}^k; \mu)' (\bar{\boldsymbol{\zeta}}^k - \boldsymbol{\zeta}^k) = 0$.
4. *Compute Step Size*:
   Compute step length $\lambda \in \arg \underset{0 \le \lambda \le 1}{\min} P\left((1-\lambda)\, \boldsymbol{\zeta}^k + \lambda\, \bar{\boldsymbol{\zeta}}^k; \mu \right)$, and get the next iterate, $\boldsymbol{\zeta}^{k+1} = (1-\lambda)\, \boldsymbol{\zeta}^k + \lambda\, \bar{\boldsymbol{\zeta}}^k$.

proof of the Frank-Wolfe algorithm with no assumptions on the convexity of $P(\boldsymbol{\zeta}^j, \mu)$ can be found in [11], thus we offer the convergence result without proof.

Theorem 3 (Convergence of SLAMS). *[[11]] Algorithm 1 terminates at $\boldsymbol{\zeta}^k$ that satisfies the minimum principle necessary optimality condition of $PF(\mu)$: $\nabla_{\boldsymbol{\zeta}} P(\boldsymbol{\zeta}^k; \mu)'(\boldsymbol{\zeta} - \boldsymbol{\zeta}^k) \ge 0$ for all $\boldsymbol{\zeta} \in S_{LP}$, or each accumulation $\bar{\boldsymbol{\zeta}}$ of the sequence $\{\boldsymbol{\zeta}^k\}$ satisfies the minimum principle.*

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

Theorem 4 (SLAMS solves LPEC). *Let $\boldsymbol{\zeta}^k$ be the sequence generated by SLAMS that accumulates to $\bar{\boldsymbol{\zeta}}$. If $\phi(\bar{\boldsymbol{\zeta}}) = 0$, then $\boldsymbol{\zeta}$ is strongly stationary for LPEC (6).*

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

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

We know that $\bar{\boldsymbol{\zeta}}$ satisfies $\mathbf{A}\bar{\boldsymbol{\zeta}} \ge \mathbf{b}$ since $\boldsymbol{\zeta}^k$ is feasible at the k-th iteration. By Theorem 3 above, $\bar{\boldsymbol{\zeta}}$ satisfies the minimum principle; thus, we know the system of equations

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

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

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

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

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

Thus $(\bar{\boldsymbol{\zeta}}, \bar{\mathbf{u}})$ is a KKT point of $PF(\mu)$ and $\bar{\boldsymbol{\zeta}}$ is a stationary point of $PF(\mu)$. By Theorem 2, $\bar{\boldsymbol{\zeta}}$ is also a strongly stationary point of LPEC (6).

4.4 Early Stopping

Typically, in many machine learning applications, emphasis is placed on generalization and scalability. Consequently, inexact solutions are preferred to globally optimal solutions as they can be obtained cheaply and tend to perform reasonably well. Noting that, at each iteration, the algorithm is working to minimize the LPEC objective as well as the complementarity penalty, one alternative to speeding up termination at the expense of the objective is to stop as soon as *complementarity is reached*. Thus, as soon as an iterate produces a solution that is feasible to the LPEC, (6), the algorithm is terminated. We call this approach *Successive Linearization Algorithm for Model Selection with Early Stopping* (EZ-SLAMS). This is similar to the well-known machine learning concept of early stopping, except that the criterion used for termination is based on the status of the complementarity constraints i.e., feasibility to the LPEC. We adapt the finite termination result in [11] to prove that EZ-SLAMS terminates finitely for the case when complementary solutions exist, which is precisely the case of interest here. Note that the proof relies upon the fact that S_{LP} is polyhedral with no straight lines going to infinity in both directions.

Theorem 5 (Finite termination of EZ-SLAMS). *Let $\boldsymbol{\zeta}^k$ be the sequence generated by SLAMS that accumulates to $\bar{\boldsymbol{\zeta}}$. If $\phi(\bar{\boldsymbol{\zeta}}) = 0$, then EZ-SLAM terminates at an LPEC (6) feasible solution $\boldsymbol{\zeta}^k$ in finitely many iterations.*

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

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

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

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

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

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

Thus, $\mathbf{v}$ is feasible for LPEC (6).

The results comparing SLAMS to EZ-SLAMS are reported in Sections 6 and 7. It is interesting to note that there is always a significant decrease in running time with typically no significant degradation in generalization performance when early stopping is employed.

4.5 Grid Search

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

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

- Perform a two-dimensional grid search on the unconstrained (classic) SVR problem to determine C and ε. We call this the *unconstrained grid search* (Unc. Grid). A coarse grid with values of 0.1, 1 and 10 for C, and 0.01, 0.1 and 1 for ε was chosen.
- Perform an n-dimensional grid search to determine the features of $\overline{\mathbf{w}}$ using C and ε obtained from the previous step. Only two distinct choices for each feature of $\overline{\mathbf{w}}$ are considered: 0, to test if the feature is redundant, and some large value that would not impede the choice of an appropriate feature weight, otherwise. Cross validation under these settings would involve solving roughly $O(3.2^N)$ problems; this number is already impractical and necessitates the heuristic. We label this step the *constrained grid search* (Con. Grid).
- For data sets with more than 10 features, *recursive feature elimination* [16] is used to rank the features and the 10 largest features are chosen, then constrained grid search is performed.

5 Experimental Design

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

5.1 Synthetic Data

Data sets of different dimensionalities, training sizes and noise models were generated. The dimensionalities i.e., number of features considered were $n = 10$, 15 and 25, among which, only $n_r = 7$, 10 and 16 features respectively, were relevant. We trained on sets of $\ell = 30$, 60, 90, 120 and 150 points using 3-fold cross

Table 1. The Chemoinformatics (QSAR) data sets

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

validation and tested on a hold-out set of a further $1,000$ points. Two different noise models were considered: Laplacian and Gaussian. For each combination of feature size, training set size and noise model, 5 trials were conducted and the test errors were averaged. In this subsection, we assume the following notation: $U(a,b)$ represents the uniform distribution on $[a,b]$, $N(\mu,\sigma)$ represents the normal distribution with probability density function $\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$, and $L(\mu,b)$ represents the Laplacian distribution with the probability density function $\frac{1}{2b}\exp\left(-\frac{|x-\mu|}{b}\right)$.

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

5.2 Real-World QSAR Data

We examined four real-world regression chemoinformatics data sets: Aquasol, Blood/Brain Barrier (BBB), Cancer, and Cholecystokinin (CCK), previously studied in [17]. The goal is to create Quantitative Structure Activity Relationship (QSAR) models to predict bioactivities typically using the supplied descriptors as part of a drug design process. The data is scaled and preprocessed to reduce the dimensionality. As was done in [17], we standardize the data at each dimension and eliminate the uninformative variables that have values outside of ± 4 standard deviations range. Next, we perform principle components analysis (PCA), and use the top 25 principal components as descriptors. The training and hold out set sizes and the dimensionalities of the final data sets are shown in Table 1. For each of the training sets, 5-fold cross validation is optimized using bilevel programming. The results are averaged over 20 runs.

The LPs within each iterate in both SLA approaches were solved with CPLEX. The penalty parameter was uniformly set to $\mu = 10^3$ and never resulted in

complementarity failure at termination. The hyper-parameters were bounded as $0.1 \leq C \leq 10$ and $0.01 \leq \varepsilon \leq 1$ so as to be consistent with the hyper-parameter ranges used in grid search. All computational times are reported in seconds.

5.3 Post-processing

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

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

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

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

and

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

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

6 Computational Results: Synthetic Data

In the following sections, constrained (abbreviated con.) methods refer to the bilevel models that have the box constraint $-\overline{\mathbf{w}} \leq \mathbf{w} \leq \overline{\mathbf{w}}$, while unconstrained (abbreviated unc.) methods refer to the bilevel models without the box constraint. In this section, we compare the performance of several different methods on synthetic data sets.

Five methods are compared: unconstrained and constrained grid search (Unc. Grid and Con. Grid), constrained SLAMS (SLAMS), constrained SLAMS with early stopping (EZ-SLAMS) and constrained FILTER based sequential quadratic programming (Filter SQP).

There are in total 15 sets of problems being solved; each set corresponds to a given dimensionality ($n = 10, 15$ or 25) and a number of training points ($\ell = 30, 60, \ldots 150$). For each set of problems, 5 methods (as described above) were employed. For each method, 10 random instances of the same problem are solved, 5 with Gaussian noise and 5 with Laplacian noise. The averaged results for the 10, 15, and 25-d data sets are shown in Tables 2, 3 and 4 respectively. Each table shows the results for increasing sizes of the training sets for a fixed

Table 2. 10-d synthetic data with Laplacian and Gaussian noise under 3-fold cross validation. Results that are significantly better or worse are tagged ✓ or ✗ respectively.

Method	Objective	Time (sec.)	MAD	MSE
30 pts				
UNC. GRID	1.385 ± 0.323	5.2± 0.5	1.376	2.973
CON. GRID	1.220 ± 0.276	635.3±59.1	1.391	3.044
FILTER (SQP)	1.065 ± 0.265	18.7± 2.2	**1.284** ✓	**2.583** ✓
SLAMS	1.183 ± 0.217	2.4± 0.9	1.320	2.746
EZ-SLAMS	1.418 ± 0.291	0.6± 0.1	1.308	2.684
60 pts				
UNC. GRID	1.200 ± 0.254	5.9± 0.5	1.208	2.324
CON. GRID	1.143 ± 0.245	709.2±55.5	1.232	2.418
FILTER (SQP)	1.099 ± 0.181	23.3± 4.4	1.213	2.328
SLAMS	1.191 ± 0.206	3.7± 2.4	1.186	2.239
EZ-SLAMS	1.232 ± 0.208	1.3± 0.3	1.186	2.238
90 pts				
UNC. GRID	1.151 ± 0.195	7.2± 0.5	1.180	2.215
CON. GRID	1.108 ± 0.192	789.8±51.7	**1.163** ✓	**2.154** ✓
FILTER (SQP)	1.069 ± 0.182	39.6±14.1	**1.155** ✓	**2.129** ✓
SLAMS	1.188 ± 0.190	5.8± 2.6	**1.158** ✓	**2.140** ✓
EZ-SLAMS	1.206 ± 0.197	2.7± 0.8	**1.159** ✓	**2.139** ✓
120 pts				
UNC. GRID	1.124 ± 0.193	7.0± 0.1	1.144	2.087
CON. GRID	1.095 ± 0.199	704.3±15.6	1.144	2.085
FILTER (SQP)	1.037 ± 0.187	30.2± 7.8	1.161	2.152
SLAMS	1.116 ± 0.193	15.6±15.3	1.141	2.082
EZ-SLAMS	1.137 ± 0.191	4.2± 1.1	1.143	2.089
150 pts				
UNC. GRID	1.091 ± 0.161	8.2± 0.3	1.147	2.098
CON. GRID	1.068 ± 0.154	725.1± 2.7	1.142	2.081
FILTER (SQP)	1.029 ± 0.171	40.6± 5.9	1.150	2.110
SLAMS	1.103 ± 0.173	20.1± 5.5	1.136	2.063
EZ-SLAMS	1.110 ± 0.172	7.4± 1.1	1.136	2.062

dimensionality. The criteria used for comparing the various methods are validation error (cross-validation objective), test error (generalization error measured as MAD or MSE on the 1000-point hold-out test set) and computation time (in seconds). For MAD and MSE, the results in bold refer to those that are significantly different than those of the unconstrained grid as measured by a two-sided t-test with significance of 0.1. The results that are significantly better and worse are tagged with a check (✓) or a cross (✗) respectively.

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

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

Method	Objective	Time (sec.)	MAD	MSE
30 pts				
Unc. Grid	1.995 ± 0.421	9.1 ± 9.3	1.726	4.871
Con. Grid	1.659 ± 0.312	735.8 ± 92.5	1.854	5.828
filter (SQP)	1.116 ± 0.163	28.7 ± 7.3	1.753	5.004
Slams	1.497 ± 0.258	5.0 ± 1.3	1.675	4.596
ez-Slams	1.991 ± 0.374	0.9 ± 0.2	1.697	4.716
60 pts				
Unc. Grid	1.613 ± 0.257	7.3 ± 1.3	1.584	4.147
Con. Grid	1.520 ± 0.265	793.5 ± 83.1	1.589	4.254
filter (SQP)	1.298 ± 0.238	52.6 ± 36.4	1.511	3.874
Slams	1.565 ± 0.203	8.3 ± 3.5	1.504	3.820
ez-Slams	1.673 ± 0.224	2.3 ± 0.3	1.498	3.807
90 pts				
Unc. Grid	1.553 ± 0.261	8.2 ± 0.5	1.445	3.553
Con. Grid	1.575 ± 0.421	866.2 ± 67.0	1.551	4.124
filter (SQP)	1.333 ± 0.254	64.7 ± 12.9	**1.407** ✓	**3.398** ✓
Slams	1.476 ± 0.182	16.3 ± 6.3	**1.411** ✓	**3.398** ✓
ez-Slams	1.524 ± 0.197	3.8 ± 0.9	1.412	**3.404** ✓
120 pts				
Unc. Grid	1.481 ± 0.240	7.5 ± 0.0	1.396	3.350
Con. Grid	1.432 ± 0.171	697.9 ± 2.2	1.395	3.333
filter (SQP)	1.321 ± 0.168	57.5 ± 11.6	1.388	3.324
Slams	1.419 ± 0.166	32.6 ± 18.6	1.375	**3.273** ✓
ez-Slams	1.474 ± 0.181	6.2 ± 0.8	1.379	3.291
150 pts				
Unc. Grid	1.448 ± 0.264	8.7 ± 0.1	1.362	3.221
Con. Grid	1.408 ± 0.232	723.2 ± 2.0	1.376	3.268
filter (SQP)	1.333 ± 0.204	85.7 ± 23.6	1.371	3.240
Slams	1.436 ± 0.217	41.8 ± 17.5	1.360	3.214
ez-Slams	1.459 ± 0.216	10.1 ± 1.8	1.359	3.206

The reported times provide a rough idea of the computational effort of each algorithm. As noted above, the computation times for the neos solver, filter, includes transmission, and waiting times as well as solve times. For grid search methods, smart restart techniques were used to gain a considerable increase in speed. However, for Con. Grid, even these techniques cannot prevent the running time from becoming impractical as the problem size grows. While the computation times of filter are both much less than that of Con. Grid, it is the SLA approaches that really dominate. The efficiency of the SLA approaches is vastly computationally superior to both grid search and filter.

The bilevel approach is much more computationally efficient than grid search on the fully parameterized problems. The results, for filter, are relatively efficient and very acceptable when considering that they include miscellaneous times for solution by neos. It is reasonable to expect that a filter implementation on

Table 4. 25-d synthetic data with Laplacian and Gaussian noise under 3-fold cross validation. Results that are significantly better or worse are tagged ✓ or ✗ respectively.

Method	Objective	Time (sec.)	MAD	MSE
30 pts				
Unc. Grid	3.413 ± 0.537	5.7 ± 0.0	2.915	13.968
Con. Grid	2.636 ± 0.566	628.5 ± 0.5	**3.687** ✗	**22.065** ✗
filter (SQP)	1.087 ± 0.292	18.0 ± 2.7	2.916	13.881
Slams	1.684 ± 0.716	7.9 ± 2.4	2.962	14.607
ez-Slams	3.100 ± 0.818	1.5 ± 0.2	2.894	13.838
60 pts				
Unc. Grid	2.375 ± 0.535	6.2 ± 0.0	2.321	8.976
Con. Grid	2.751 ± 0.653	660.9 ± 1.6	**3.212** ✗	**16.734** ✗
filter (SQP)	1.467 ± 0.271	53.1 ± 11.1	2.282	8.664
Slams	2.065 ± 0.469	15.3 ± 7.1	2.305	8.855
ez-Slams	2.362 ± 0.441	3.1 ± 0.4	2.312	8.894
90 pts				
Unc. Grid	2.256 ± 0.363	7.0 ± 0.0	2.161	7.932
Con. Grid	2.927 ± 0.663	674.7 ± 1.0	**3.117** ✗	**15.863** ✗
filter (SQP)	1.641 ± 0.252	86.0 ± 15.5	**2.098** ✓	**7.528** ✓
Slams	2.149 ± 0.304	29.3 ± 12.2	2.119	7.711
ez-Slams	2.328 ± 0.400	6.3 ± 1.2	2.131	7.803
120 pts				
Unc. Grid	2.147 ± 0.343	8.4 ± 0.0	2.089	7.505
Con. Grid	2.910 ± 0.603	696.7 ± 1.5	**3.124** ✗	**15.966** ✗
Slams	2.156 ± 0.433	45.6 ± 16.4	**2.028** ✓	**7.121** ✓
ez-Slams	2.226 ± 0.461	10.3 ± 1.5	**2.034** ✓	**7.154** ✓
150 pts				
Unc. Grid	2.186 ± 0.383	9.9 ± 0.1	1.969	6.717
Con. Grid	2.759 ± 0.515	721.1 ± 1.7	**2.870** ✗	**13.771** ✗
Slams	2.069 ± 0.368	63.5 ± 30.5	1.949	6.636
ez-Slams	2.134 ± 0.380	14.2 ± 2.5	**1.947** ✓	6.619

a local machine (instead of over the internet) would require significantly less computation times, which could bring it even closer to the times of Unc. Grid or the SLA methods. The filter approach does have a drawback, in that is that it tends to struggle as the problem size increases. For the synthetic data, filter failed to solve 10 problems each from the 25d data sets with 120 and 150 points and these runs have been left out of Table 4.

Of course, in machine learning, an important measure of performance is generalization error. These problems were generated with irrelevant variables; presumably, appropriate choices of the symmetric box parameters in the bilevel problem could improve generalization. (This topic is worth further investigation but is beyond the scope of this paper.) Compared to classic SVR optimized with Unc. Grid, filter and the SLA approaches yield solutions that are better or comparable to the test problems and never significantly worse. In contrast, the generalization performance of Con. Grid steadily degrades as problem size and dimensionality grow.

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

Method	Objective	Time (sec.)	MAD	MSE
Aquasol				
Unc. Grid	0.719 ± 0.101	17.1 ± 0.4	0.644	0.912
Con. Grid	0.778 ± 0.094	1395.9 ± 3.5	**0.849** ✗	**1.605** ✗
filter (SQP)	0.574 ± 0.083	1253.0 ± 533.7	0.676	0.972
Slams	0.670 ± 0.092	137.8 ± 52.0	0.647	0.911
ez-Slams	0.710 ± 0.088	19.1 ± 3.3	0.643	0.907
Blood/Brain Barrier				
Unc. Grid	0.364 ± 0.048	13.4 ± 1.9	0.314	0.229
Con. Grid	0.463 ± 0.081	1285.7 ± 155.3	**0.733** ✗	**0.856** ✗
filter (SQP)	0.204 ± 0.043	572.7 ± 339.5	0.338	0.214
Slams	0.363 ± 0.042	17.1 ± 9.8	0.312	0.231
ez-Slams	0.370 ± 0.042	8.0 ± 1.6	0.315	0.235
Cancer				
Unc. Grid	0.489 ± 0.032	10.3 ± 0.9	0.502	0.472
Con. Grid	0.477 ± 0.065	1035.3 ± 1.5	**0.611** ✗	**0.653** ✗
filter (SQP)	0.313 ± 0.064	180.8 ± 64.3	0.454	0.340
Slams	0.476 ± 0.086	25.5 ± 9.2	0.481	**0.336** ✓
ez-Slams	0.567 ± 0.096	5.2 ± 1.1	0.483	**0.341** ✓
Cholecystokinin				
Unc. Grid	0.798 ± 0.055	12.0 ± 0.4	1.006	1.625
Con. Grid	0.783 ± 0.071	1157.6 ± 1.8	**1.280** ✗	**2.483** ✗
filter (SQP)	0.543 ± 0.063	542.3 ± 211.5	0.981	1.520
Slams	0.881 ± 0.108	35.1 ± 20.4	**1.235** ✗	**2.584** ✗
ez-Slams	0.941 ± 0.092	9.1 ± 1.3	**1.217** ✗	**2.571** ✗

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

7 Computational Results: QSAR Data

Table 5 shows the average results for the QSAR data. After the data is preprocessed, we randomly partition the data into 20 different training and testing sets. For each of the training sets, 5-fold cross validation is optimized using bilevel programming. The results are averaged over the 20 runs. We report results for the same 5 methods as those used for synthetic data. The parameter settings

used in the grid searches, FILTER and SLA approaches and the statistics reported are the same as those used for the synthetic data.

Again, as with the synthetic data, FILTER finds solutions with the smallest cross-validated training errors or equivalently objective value. SLAMS also finds good quality solutions except for the clearly suboptimal solution found on the Cholecystokinin data. The SLA method have very good computational times. However, computation times for FILTER are not competitive with the SLA methods or Unc. Grid. Unsurprisingly, constrained grid search has the worst computation time. The difficulty of the underlying bilevel optimization problem is underscored by the fact that the greedy Con. Grid search in Section 4.5 sometimes fails to find a better solution than the unconstrained grid search. The constrained search drops important variables that cause it to have bad generalization.

In terms of test set error, FILTER performs the best. SLA also performs quite well on all data sets except on Cholecysotkinin, where the SLA get trapped in poor local minima. However, on the remaining data sets, the SLA approaches generalize very well and tend to be competitive with Unc. Grid with regard to execution time. The best running times, however, are produced by the early stopping based SLA approaches, which SLAM the door on all other approaches computationally while maintaining as of good generalization performance as SLAMS.

8 Discussion

We showed how the widely used model selection technique of cross validation (for support vector regression) could be formulated as a bilevel programming problem; the formulation is more flexible and can deal with many more hyper-parameters than the typical grid search strategy which quickly becomes intractable as the hyper-parameter space increases. The proposed bilevel problem is converted to an instance of a linear program with equilibrium constraints (LPEC). This class of problems is difficult to solve due to the non-convexity created by the complementarity constraints introduced in the reformulation. A major outstanding question has always been the development of efficient algorithms for LPECs and bilevel programs. To this end, we proposed two approaches to solve the LPEC: a relaxed NLP-based approach which was solved using the off-the-shelf, SQP-based, NLP solver, FILTER and a exact penalty-based approach which was solved using a finite successive linearization algorithm.

Our preliminary computational results indicate that general purpose SQP solvers can tractably find high-quality solutions that generalize well. The computation times of the FILTER solver are especially impressive considering the fact that they are obtained via internet connections and shared resources. Generalization results on random data show that FILTER yields are comparable, if not better than current methods. Interestingly, SLAMS typically finds worse solutions than FILTER in terms of the objective (cross-validated error) but with very comparable generalization. The best estimate of the generalization error to be optimized in the bilevel program remains an open question. The SLAMS algorithm computationally outperforms classical grid search and the FILTER

solver especially as the number of hyper-parameters and data points grows. We have demonstrated scalability to high dimensional data sets containing up to thousands points (results not reported here).

The computational speed of NLP- or the SLA-based approaches can be improved by taking advantage of the structure inherent in bilevel problems arising from machine learning applications. Machine learning problems, especially support vector machines, are highly structured, and yield elegant and sparse solutions, a fact that several decomposition algorithms such as sequential minimal optimization target. Despite the non-convexity of the LPECs, bilevel programs for machine learning problems retain the structure inherent in the original machine learning problems. In addition, the variables in these LPECs tend to decouple, for example, in cross validation, the variables may be decoupled along the folds. This suggests that applying decomposition or cutting-plane methods to bilevel approaches can make them even more efficient. An avenue for future research is developing decomposition-based or cutting-plane algorithms that can train on data sets containing tens of thousands of points or more.

While support vector regression was chosen as the machine learning problem to demonstrate the potency of the bilevel approach, the methodology can be extended to several machine learning problems including classification, semi-supervised learning, multi-task learning, missing value imputation, and novelty detection. Some of these formulations have been presented in [2], while others remain open problems. Aside from discriminative methods, bilevel programming can also be applied to generative methods such as Bayesian techniques. Furthermore, the ability to optimize a large number of parameters allows one to consider new forms of models, loss functions and regularization.

Another pressing question, however, arises from a serious limitation of the formulation presented herein: the model can only handle linear data sets. Classical machine learning addresses this problem by means of the kernel trick. It was shown in [2] that the kernel trick can be incorporated into a generic bilevel model for cross validation. The flexibility of the bilevel approach means that one can even incorporate input-space feature selection into the kernelized bilevel model. This type of bilevel program can be reformulated as an instance of a mathematical program with equilibrium constraints (MPEC). The MPECs arising from kernelized bilevel machine learning problems tend to have several diverse sources of non-convexity because they have nonlinear complementarity constraints; this leads to very challenging mathematical programs and an equally challenging opening for future pursuits in this field.

References

1. Bennett, K., Hu, J., Ji, X., Kunapuli, G., Pang, J.: Model selection via bilevel optimization. In: International Joint Conference on Neural Networks (IJCNN 2006), pp. 1922–1929 (2006)
2. Kunapuli, G., Bennett, K., Hu, J., Pang, J.: Bilevel model selection for support vector machines. In: Hansen, P., Pardolos, P. (eds.) CRM Proceedings and Lecture Notes. American Mathematical Society (in press, 2008)

3. Bracken, J., McGill, J.: Mathematical programs with optimization problems in the constraints, vol. 21, pp. 37–44 (1973)
4. Luo, Z., Pang, J., Ralph, D.: Mathematical Programs With Equilibrium Constraints. Cambridge University Press, Cambridge (1996)
5. Facchinei, F., Pang, J.: Finite-Dimensional Variational Inequalities and Complementarity Problems. Springer, New York (2003)
6. Outrata, J., Kocvara, M., Zowe, J.: Nonsmooth Approach to Optimization Problems with Equilibrium Constraints: Theory, Applications and Numerical Results. Kluwer Academic Publishers, Dordrecht (1998)
7. Dempe, S.: Foundations of Bilevel Programming. Kluwer Academic Publishers, Dordrecht (2002)
8. Dempe, S.: Annotated bibliography on bilevel programming and mathematical programs with equilibrium constraints. Optimization 52, 333–359 (2003)
9. Ralph, D., Wright, S.: Some properties of regularization and penalization schemes for mpecs. Optimization Methods and Software 19, 527–556 (2004)
10. Mangasarian, O.: Misclassification minimization. Journal of Global Optimization 5, 309–323 (1994)
11. Bennett, K.P., Mangasarian, O.L.: Bilinear separation of two sets in n-space. Computational Optimization and Applications 2, 207–227 (1993)
12. Fletcher, R., Leyffer, S.: Nonlinear programming without a penalty function. Mathematical Programming 91, 239–269 (2002)
13. Fletcher, R., Leyffer, S.: User manual for filtersqp Tech. Report NA/181, Department of Mathematics, University of Dundee (1999), `http://www-unix.mcs.anl.gov/leyffer/papers/SQP_manual.pdf`
14. Gill, P., Murray, W., Saunders, M.: User's guide for snopt version 6: A fortran package for large-scale nonlinear programming (2002)
15. Huang, X., Yang, X., Teo, K.: Partial augmented lagrangian method and mathematical programs with complementarity constraints. Journal of Global Optimization 35, 235–254 (2006)
16. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. Journal of Machine Learning Research 3, 1157–1182 (2003)
17. Demiriz, A., Bennett, K., Breneman, C., Embrecht, M.: Support vector regression methods in cheminformatics. Computer Science and Statistics 33 (2001)