Model Selection via Bilevel Optimization

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

***Abstract*— A key step in many statistical learning methods used in machine learning involves solving a convex optimization problem containing one or more hyper-parameters that must be selected by the users. While cross validation is a commonly employed and widely accepted method for selecting these parameters, its implementation by a grid-search procedure in the parameter space effectively limits the desirable number of hyper-parameters in a model, due to the combinatorial explosion of grid points in high dimensions. This paper proposes a novel *bilevel optimization* approach to cross validation that provides a systematic search of the hyper-parameters. The bilevel approach enables the use of the state-of-the-art optimization methods and their well-supported softwares. After introducing the bilevel programming approach, we discuss computational methods for solving a bilevel cross-validation program, and present numerical results to substantiate the viability of this novel approach as a promising computational tool for model selection in machine learning.**

I. Introduction

Support Vector Machines (SVM), [5], [26], kernel methods, [24], and other statistical learning methods have been applied with great success. But the many papers reporting the success of such methods frequently gloss over an important issue: model selection. For example, consider kernel methods. In kernel methods, the learning task, such as regression, classification, ranking, and novelty detection [25], is to construct a linear function that minimizes a regularized convex loss function. Nonlinear functions can be constructed using the so-called "kernel trick". The resulting optimization problem is convex (thus, is generally not difficult to deal with, both theoretically and computationally), but typically it contains *hyper-parameters* that must be selected by the users. For example in SVM, the appropriate kernel function and tradeoff parameter between error and regularization must both be selected. While there have been many interesting attempts to use bounds or other techniques to pick these hyper-parameters [6], [9], the most commonly used and widely accepted method for selecting these hyper-parameters is still cross validation (CV).

In cross validation, 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 [21]. 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. Problems with many parameters are pervasive in data analysis, e.g., they arise frequently in feature selection [16], [2], kernel construction [19], [22], and multitask learning [4], [10]. 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 fails to take into account the fact that the model parameters are continuous. Recent work on determining the full regularization path of support vector machines underscores the fact that regularization parameter is continuous. In particular, the paper [18] 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*.

The novelty of this research is to directly tackle the model selection using out-of-sample testing 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—the inner (or lower-level) optimization problem, while the overall testing objective is minimized—the outer (or upper-level) optimization problem. Prior bilevel approaches have been developed and successfully used for lower-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 [15]. But these approaches are limited to a single hyper-parameter and lower-level function with a closed-form solution.

The authors are with the Department of Mathematical Sciences, Rensselaer Polytechnic Institute, 110 8th Street, Troy, New York 12180-3590, U.S.A. Email: (bennek,huj,jix, kunapg,pangj)@rpi.edu.

The proposed bilevel programming approaches offer several fundamental advantages over prior approaches. First, recent advances in bilevel programming in the optimization community permit the systematic treatment of models based on the popular loss functions used for SVM and kernel methods with many hyper-parameters; see Section III for a brief summary of such advances. In addition to the ability to simultaneously optimize many hyper-parameters, the bilevel programming approach offers a broad framework in which novel regularization methods can be developed, valid bounds on the test set errors can be obtained, and most significantly, improved model selection can be produced.

II. Challenges in Model Selection

At one level, the model selection problem is easy. Pick a parametric family of models, and an appropriate training loss function, such that the model performs well according to some estimate of the generalization error based on the given training data. Yet, every stage of the process can introduce errors that can degrade the quality of the resulting inductive functions. We highlight three sources of such errors. The first source of error is the fact that the underlying true function and error distribution is unknown; thus any choice of data representation, model family and loss functions may not be suitable for the problem and thus introduce inappropriate bias. The second source of error stems from the fact that only finite amount of (possibly noisy) data is available. Thus even if we pick appropriate loss functions, models, and out-of-sample estimates, the method may still yield inappropriate results. The third source of error stems from the difficulty of the nonconvex optimization problem that underlies the model selection problem. Support vector and other kernel methods are usually regarded as convex optimization problems, but in fact they are not; they are made convex when the parameters are fixed. Ideally, both the variables and parameters used in these models must be selected. As mentioned above, the commonly used grid-search approach and other heuristic methods are highly deficient for dealing with problems with many parameters. Our proposal of the bilevel programming approach places this parameter selection problem on a firm ground, enabling the employment of the state-of-the-art nonlinear programming (NLP) methodology, including many highly effective algorithmic solvers that are freely available on the NEOS website, http://www-neos.mcs.anl.gov/neos/solvers/, to tackle this third challenge of model selection. In addition to such a rich resource, on-going research is underway to improve the effectiveness of these solvers applied to this class of optimization problems; these details are beyond the scope of this paper.

Since our focus is the third challenge, we assume that the model family and the loss function for the training data, and the method for estimating the generalization error are given. While the bilevel programming approach is broadly applicable to many function classes and learning methods, we focus in this paper on support vector based training. Specifically, we optimize the model with respect to the ε-insensitive loss function with 2-norm regularization and with a trade-off parameter C [26]. Further we assume the data has irrelevant variables; to help identify them, we introduce a symmetric box constraint on the support vector[1]. This introduces an additional nonnegative vector of parameters, denoted $\overline{\mathbf{w}}(= -\underline{\mathbf{w}})$, each of whose components is the size of the box for an individual dimension of the data. Thus the linear regression problem for n-dimensional data has $n + 2$ parameters, C, ε, and $\overline{\mathbf{w}}$, which are chosen such that the k-fold least-absolute deviation cross-validation error is minimized. Note that the linear model for each fold is a convex quadratic program that is easy to solve to global optimality for each choice of parameters. Among many questions raised by this formulation, we focus on two: Can the bilevel k-fold CV optimization problem be solved efficiently? Do we get improved generalization errors? Results are reported in Section VI.

III. Bilevel Optimization

Since bilevel optimization is a novel tool in the machine learning community, we introduce this methodology by way of a brief historical perspective. In a nutshell, 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 the 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 Stackelberg game, whose origin predates the work of Bracken and McGill. In the late 1980s, the bilevel program was given a renewed study in the extended framework of a *mathematical program with equilibrium constraints* (MPEC) [20], which is an extension of a bilevel program with the optimization constraint replaced by a finite-dimensional variational inequality [11].

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 [20], which is followed by two related monographs [23] and [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.

IV. A Bilevel Support-Vector Regression Model

As mentioned before, 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,

[1]This idea was suggested by Stan Uryasev at the University of Florida, Gainesville in a private communication to the last author of this paper in March 2004.

suppose that the regression data are described by the ℓ points $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell)\}$ in the Euclidean space $\Re^{n+1}$ for some positive integers ℓ and n. Consider the regression problem of finding a function $f^* : \Re^n \to \Re$ 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., $P[f] \equiv \|\mathbf{w}\|_2^2 = \sum_{i=1}^{n} w_i^2$.

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 distinct partitions, Ω_t for $t = 1, \ldots, T$, such that $\bigcup_{t=1}^{T} \Omega_t = \{1, \ldots, \ell\}$. Let $\overline{\Omega}_t \equiv \{1, \ldots, \ell\} \setminus \Omega_t$ be the subset of the data other than those in group Ω_t. 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, λ, and $\mathbf{w}_t$ for $t = 0, 1, \cdots, T$, and also the bounds $\underline{\mathbf{w}}$ and $\overline{\mathbf{w}}$ in order to

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

$$\begin{aligned} \mathbf{w}^t \in \underset{\underline{\mathbf{w}} \leq \mathbf{w} \leq \overline{\mathbf{w}}}{\arg\min} \Bigg\{ & C \sum_{j \in \overline{\Omega}_t} \max(|\mathbf{x}_j'\mathbf{w} - y_j| - \varepsilon, 0) \\ & + \frac{1}{2} \|\,\mathbf{w}\,\|_2^2 + \frac{\lambda}{2} \|\,\mathbf{w} - \mathbf{w}_0\,\|_2^2 \Bigg\}, \end{aligned} \tag{2}$$

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, λ, $\mathbf{w}_0$, $\underline{\mathbf{w}}$, and $\overline{\mathbf{w}}$. Problem 1 is called the first-level or outer problem. Problem (2) is referred to as the the second-level or inner problem. 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 parameter λ is inspired by current research in multitask learning [4], [10], which in turn is motivated by statistical learning theory. Since we have set $\lambda = 0$ in our experiments reported in Section VI, we will not discuss this parameter further. We included λ in the formulation (1) in order to make the point that the bilevel optimization methodology is applicable to multitask learning. 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 will force $\mathbf{w}$ to be sparse. The box constraints will ensure that a consistent but not necessarily identical set 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 VI 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. In general, each such variation of the basic formulation (1) will have interesting modeling implications; details of these are beyond the scope of this first paper on the subject and are presently being further investigated.

A. More about the bilevel problem

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

$$\frac{1}{T} \sum_{t=1}^{T} \frac{1}{|\,\Omega_t\,|} \sum_{i \in \Omega_t} |\,\mathbf{x}_i'\mathbf{w}^t - y_i\,|,$$

subject to the simple restrictions on these parameters, and most importantly, to the additional lower-level optimality requirement of each $\mathbf{w}^t$ for $t = 1, \ldots, T$. To solve (1), we rewrite the lower-level optimization problem (2) as follows:

for given ε, C, λ, $\mathbf{w}_0$, $\underline{\mathbf{w}}$, and $\overline{\mathbf{w}}$,

$$\begin{array}{ll} \text{minimize} & C\displaystyle\sum_{j\in\overline{\Omega}_t} e_j + \frac{1}{2}\,\|\,\mathbf{w}\,\|_2^2 + \frac{\lambda}{2}\,\|\,\mathbf{w}-\mathbf{w}_0\,\|_2^2 \\ \text{subject to} & \underline{\mathbf{w}} \le \mathbf{w} \le \overline{\mathbf{w}} \\ \text{and} & \\ & \left\{\begin{array}{l} e_j \ge \mathbf{x}_j'\mathbf{w} - y_j - \varepsilon \\ e_j \ge -\mathbf{x}_j'\mathbf{w} + y_j - \varepsilon \\ e_j \ge 0 \end{array}\right\} \quad j \in \overline{\Omega}_t, \end{array} \tag{3}$$

which is easily seen to be a convex quadratic program in the variables $\mathbf{w}$ and $\{e_j\}_{j\in\overline{\Omega}_t}$. By letting $\boldsymbol{\gamma}^{t,\pm}$ be the multipliers of the bound constraints: $\underline{\mathbf{w}} \le \mathbf{w} \le \overline{\mathbf{w}}$, respectively, and $\eta_j^{t,\pm}$ be the multipliers of the constraints $e_j \ge \mathbf{x}_j'\mathbf{w} - y_j - \varepsilon$ and $e_j \ge -\mathbf{x}_j'\mathbf{w} + y_j - \varepsilon$, respectively, we obtain the Karush-Tucker-Tucker optimality conditions of (3) as the following linear complementarity problem in the variables $\mathbf{w}^t$, $\boldsymbol{\gamma}^{t,\pm}$, $\eta_j^{t,\pm}$, and $e_j^{t,\pm}$:

$$\begin{array}{l} 0 \le \boldsymbol{\gamma}^{t,-} \perp \mathbf{w}^t - \underline{\mathbf{w}} \ge 0 \\ 0 \le \boldsymbol{\gamma}^{t,+} \perp \overline{\mathbf{w}} - \mathbf{w}^t \ge 0 \\ \left.\begin{array}{l} 0 \le \eta_j^{t,-} \perp \mathbf{x}_j'\mathbf{w}^t - y_j + \varepsilon + e_j^t \ge 0 \\ 0 \le \eta_j^{t,+} \perp y_j - \mathbf{x}_i'\mathbf{w}^t + \varepsilon + e_j^t \ge 0 \\ 0 \le e_j^t \perp C - \eta_j^{t,+} - \eta_j^{t,-} \ge 0 \end{array}\right\} \forall j \in \overline{\Omega}_t \\ 0 = \mathbf{w}^t + \lambda(\mathbf{w}^t - \mathbf{w}_0) + \displaystyle\sum_{j\in\overline{\Omega}_t} (\eta_j^{t,+} - \eta_j^{t,-})\mathbf{x}_j + \boldsymbol{\gamma}^{t,+} - \boldsymbol{\gamma}^{t,-}, \end{array} \tag{4}$$

where $a \perp b$ means $a'b = 0$. The orthogonality conditions in (4) express the well-known complementary slackness properties in the optimality conditions of the lower-level (parametric) quadratic program. The overall two-level regression problem is therefore

$$\begin{array}{ll} \text{minimize} & \displaystyle\frac{1}{T}\sum_{t=1}^{T} \frac{1}{|\,\Omega_t\,|} \sum_{i\in\Omega_t} z_i^t \\ \text{subject to} & \varepsilon, C, \lambda \ge 0, \quad \underline{\mathbf{w}} \le \overline{\mathbf{w}}, \\ \text{and} & \text{for all } t = 1, \ldots, T \\ & \left\{\begin{array}{l} -z_i^t \le \mathbf{x}_i'\mathbf{w}^t - y_i \le z_i^t, \quad \forall\, i \in \Omega_t \\ \text{plus all conditions in (4)} \end{array}\right\}. \end{array} \tag{5}$$

The most noteworthy feature of the above optimization problem is the complementarity conditions in the constraints, making the problem an instance of an MPEC. A particular case of the problem is of further interest; namely, when $\lambda = 0$; i.e., when the variance of the support vector $\mathbf{w}$ is not of concern in the model selection. In this case, (5) is a linear program with linear complementarity constraints (sometimes called an LPEC). The discussion in the remainder of this paper focuses on this case.

V. A RELAXED NLP REFORMULATION

Exploiting the LPEC structure, the solution method that is implemented in our experiments for solving (5) with $\lambda = 0$ employs a relaxation of the complementarity constraint. Whereas (5) is by itself a nonlinearly constrained optimization problem, it is well recognized that a straightforward solution using this 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: one 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 (5). The other proposal is to relax the complementarity constraints and retain the relaxations in the constraints. There are extensive studies of both treatments, including detailed convergence analyses and numerical experimentations on realistic applications and random problems. For our purpose, we choose the relaxed complementarity formulation. Specifically, let tol > 0 be a prescribed tolerance of the complementarity conditions. Consider the relaxed formulation of (5), with $\lambda = 0$,

$$\begin{array}{ll} \text{minimize} & \displaystyle\frac{1}{T}\sum_{t=1}^{T} \frac{1}{|\,\Omega_t\,|} \sum_{i\in\Omega_t} z_i^t \\ \text{subject to} & \varepsilon, C \ge 0, \quad \underline{\mathbf{w}} \le \overline{\mathbf{w}}, \\ \text{and} & \text{for all } t = 1, \ldots, T \\ & \left.\begin{array}{l} -z_i^t \le \mathbf{x}_i'\mathbf{w}^t - y_i \le z_i^t, \quad \forall\, i \in \Omega_t \\ 0 \le \boldsymbol{\gamma}^{t,-} \perp_{\text{tol}} \mathbf{w}^t - \underline{\mathbf{w}} \ge 0 \\ 0 \le \boldsymbol{\gamma}^{t,+} \perp_{\text{tol}} \overline{\mathbf{w}} - \mathbf{w}^t \ge 0 \\ \left.\begin{array}{l} 0 \le \eta_j^{t,-} \perp_{\text{tol}} \mathbf{x}_j'\mathbf{w}^t - y_j + \varepsilon + e_j^t \ge 0 \\ 0 \le \eta_j^{t,+} \perp_{\text{tol}} y_j - \mathbf{x}_i'\mathbf{w}^t + \varepsilon + e_j^t \ge 0 \\ 0 \le e_j^t \perp_{\text{tol}} C - \eta_j^{t,+} - \eta_j^{t,-} \ge 0 \end{array}\right\} \forall j \in \overline{\Omega}_t \\ 0 = \mathbf{w}^t + \displaystyle\sum_{j\in\overline{\Omega}_t} (\eta_j^{t,+} - \eta_j^{t,-})\mathbf{x}_j + \boldsymbol{\gamma}^{t,+} - \boldsymbol{\gamma}^{t,-}, \end{array}\right\} \end{array}$$

where $a \perp_{\text{tol}} b$ means $ab \le$ 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 nonconvex optimization problem; thus finding a global optimal solution is hard, but the state-of-the-art general-purpose NLP solvers such as FILTER [12], [13] and SNOPT [14] that are available on the NEOS server are capable of computing good-quality feasible solutions. This internet server offers administration tools to allow 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, or via the NEOS Application Interface, and then 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 solvers, SNOPT and FILTER, were chosen to solve our problems. Both are Sequential Quadratic Programming (SQP) based methods, which are Newton-type methods for solving problems with nonlinear objectives and nonlinear constraints. These methods solve a sequence of approximate convex quadratic programming subproblems. A recent algorithm, 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 with a specified tolerance or no further step can be processed (possibly due to the infeasibility of a subproblem). In the experiment, we used FILTER with the option mxws=60000 to increase the storage. In some cases with Gaussian noise (see next section), like "10D and 90 points", "10D and 60 points", "15D and 90 points" and "15D and 90 points", we added the option rho=15 to increase the robustness of FILTER. The code SNOPT uses SQP as well. From a starting point, the algorithm sets up a quadratic subproblem to search the direction for the next subproblem. At each iteration, instead of using a trust-region formulation, an augmented Lagrangian merit function is minimized along the derived direction to induce global convergence [14]. In the experiment, we used SNOPT with the option: scale=2 and option=2. This setting yields a solution in all cases except a few, where SNOPT doesn't deliver a result. The reported computational results exclude these cases.

VI. Experimental Design

Our preliminary experiments aim to address two issues: how effective is the NLP relaxation of the bilevel CV problem and how the results compare with alternative approaches. We use the slightly modified version of the problem (1) where we set $-\underline{\mathbf{w}} = \overline{\mathbf{w}}$ and $\lambda = 0$. Randomly generated data with various dimensions were used, with the number of CV folds being $T = 3$. We set the complementarity tol=10^{-6} throughout except in some cases like 15D data with Laplacian noise, we used 10^{-4} for tol in order to increase the robustness of the NLP solvers. We compare the bilevel formulation with classical cross-validation methods on both synthetic and real data problems. The following sections explain the experimental setup and the implementation details of grid search.

A. Synthetic Data

Data sets of different dimensionalities, training sizes and noise models were generated. The dimensionalities i.e., number of features considered were $n = 5$, 10, and 15, among which, only $n_r = 3$, 7, and 10 features respectively, were relevant. One goal of the experiments is to determine how effective the bound $\overline{\mathbf{w}}$ is in identifying the irrelevant features. We trained on sets of $\ell = 15$, 30, 60 and 90 points 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.

B. Real-world QSAR data

We examined four real-world regression cheminformatics data sets: Aquasol, Blood/Brain Barrier (BBB), Cancer, and Cholecystokinin (CCK), previously studied in [1]. The goal is to create Quantitative Structure Activity Relationship (QSAR) models to predict bioactivites typically using the supplied descriptors as part of a drug design process. The data is scaled and preprocessed to reduce the dimensionality. As was done in [1], we standardize the data at each dimension and eliminate the uninformative variables that have values outside of ± 4 standard deviations range. Next, we do PCA(Principle Component Analysis), and use the top 25 principal components as descriptors. The FILTER solver on NEOS cannot currently reliably solve our bilevel programs resulting from data with more than 25 dimensions.

TABLE I
4 QSAR DATA SETS

Data set	No of Obs.	No of Vars.	No of Vars.
Aquasol	197	640	149
B/B Barrier (BBB)	62	694	569
Cancer	46	769	362
Cholecystokinin (CCK)	66	626	350

C. 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).
- We further restrict to a maximum of $n = 10$ features. For data sets with more features, *recursive feature elimination* [17] is used to rank the features and the 10 largest features are chosen.

D. Post-processing

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

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

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

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

VII. Computational Results: synthetic data

There are in total 12 sets of problems being solved; each set corresponds to a given dimensionality (n = 5, 10, and 15) and a number of training points (ℓ = 15, 30, 60, and 90). For each set of problems, 4 methods (Unc.Grid, Con.Grid, FILTER and SNOPT) were employed. For each method, 10 random instances of the same problem are solved, 5 with Gaussian noise and 5 with Laplacian noise. The results of these 10 runs are averaged and reported in Tables II, III and IV. Each table shows the results for increasing sizes of the training sets for a fixed dimensionality. We report Obj, the bilevel objective value found (for Grid search, this is simply the outer objective value produced by the folds); Time, the computation time as reported by MATLAB for Grid Search and by AMPL for the NLP results (note that the NEOS times include the total time for submission, waiting, and solution); MAD, mean absolute deviation on the 1000 point testing set; and MSE, mean square error on the 1000 point testing set. Standard deviations are provided for Obj and Time to indicate variability. 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.

From an optimization perspective, the bilevel programming methods significantly outperform the Grid approaches. The objective values found by the bilevel methods are much smaller than those found by the Grid methods. FILTER finds a lower objective more often than SNOPT. Of the 120 problems attempted, the objective value of FILTER was lower than that of SNOPT in 67 of them. The coarse grid size and heuristic used in the grid search cause it to find relatively poor objective values.

The reported times provide a rough idea of the computational effort of each algorithm. As noted above, the computation times for the 2 NEOS solvers include transmission, and waiting times as well as solve times. Also it is quite possible to speed up the grid searches through smart restart and parametric programming strategies. But clearly the Con.Grid algorithm becomes impractical as the problem size grows. The computation times of FILTER and SNOPT are both much less than that of Con.Grid. So as expected, the bilevel approach is much more computationally efficient than grid search on the fully parameterized problems. The results are amazingly efficient and quite acceptable when considering that they include miscellaneous times for solution by NEOS. It is reasonable to expect that a FILTER implementation on 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.

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 Unconstrained

Grid, FILTER yields solutions that are significantly better on 5 of the 12 problems and never significantly worse. In contrast, the Con.Grid only does significantly better than Unc.Grid on one of the 12 sets of problems. Between two NLP solvers, SNOPT is much slower than FILTER (though still much faster than Con.Grid) and the generalization results for FILTER are more consistent. Thus, for this set of experimented problems, FILTER seems like the preferred approach.

TABLE II
RESULTS FOR 5D DATA WITH LAPLACIAN AND GAUSSIAN NOISE

Method	Objective	Time	MAD	MSE
15 pts				
Unc. Grid	1.436 ± 0.419	5.2± 0.4	1.519	3.886
Con. Grid	1.268 ± 0.355	19.5± 1.4	1.505	3.854
Snopt (9)	1.335 ± 0.291	19.7± 3.1	1.450	3.475
Filter	1.180 ± 0.324	17.9± 2.6	1.374	3.167
30 pts				
Unc. Grid	1.345 ± 0.411	5.2± 0.4	1.269	2.733
Con. Grid	1.283 ± 0.364	19.9± 1.2	1.244	2.633
Snopt	1.223 ± 0.388	25.9± 5.4	1.273	2.733
Filter	1.174 ± 0.318	17.2± 2.7	1.251	2.620
60 pts				
Unc. Grid	1.203 ± 0.258	6.1± 0.3	1.149	2.284
Con. Grid	1.172 ± 0.250	23.0± 1.6	1.142	2.267
Snopt (9)	1.163 ± 0.244	42.5±15.8	**1.160**	**2.284**
Filter	1.144 ± 0.226	21.1± 5.0	**1.169**	**2.341**
90 pts				
Unc. Grid	1.169 ± 0.260	7.0± 0.6	1.149	2.296
Con. Grid	1.149 ± 0.243	24.7± 1.7	**1.133**	2.239
Snopt (9)	1.168 ± 0.287	59.3±22.6	1.161	2.303
Filter	1.121 ± 0.229	28.8± 7.5	1.141	2.260

TABLE III
RESULTS FOR 10D DATA WITH LAPLACIAN AND GAUSSIAN NOISE

Method	Objective	Time	MAD	MSE
15 pts				
Unc. Grid	1.896 ± 0.661	5.1± 0.6	1.812	5.336
Con. Grid	1.368 ± 0.437	604.3± 49.4	1.840	6.124
Snopt	0.807 ± 0.293	23.7± 1.7	1.806	5.737
Filter	0.803 ± 0.266	19.6± 3.8	**1.605**	**4.129**
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
Snopt	1.019 ± 0.254	42.5± 7.6	1.351	2.871
Filter	1.065 ± 0.265	18.7± 2.2	**1.284**	**2.583**
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
Snopt	1.070 ± 0.204	116.5± 70.1	1.202	2.284
Filter	1.099 ± 0.181	23.3± 4.4	1.213	2.328
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**
Snopt (9)	1.073 ± 0.211	184.9±156.7	1.160	2.144
Filter	1.069 ± 0.182	39.6± 14.1	**1.155**	**2.129**

VIII. COMPUTATIONAL RESULTS: QSAR DATA

Table V shows the average results for the QSAR data, on which we ran 12 experiments. After the data is preprocessed, we randomly partition the data into 20 different training and testing sets. The number of training points are 100, 60, 40, 60 for Aquasol, BBB, Cancer, CCK respectively. For each of the training sets, 5-fold cross validation is optimized using bilevel programming. The results are averaged 20 runs. We report results for Unc.Grid, , Unc.Grid, Con.Grid, and FILTER. The parameters settings used in the grid searches and FILTER and the statistics reported are the same as those used for the synthetic data. For MAD and MSE, the bold results are the ones which are significantly different from those of unconstrained grid as measured by a two-sided t-test with significance of 0.1.

TABLE IV
RESULTS FOR 15D DATA WITH LAPLACIAN AND GAUSSIAN NOISE

Method	Objective	Time	MAD	MSE
15 pts				
Unc. Grid	2.653 ± 0.576	6.1± 1.5	2.377	9.090
Con. Grid	1.790 ± 0.646	730.0±260.5	**2.726**	**11.959**
Snopt (9)	0.741 ± 0.262	37.4± 2.2	**2.714**	**11.713**
Filter	0.563 ± 0.205	23.7± 8.5	2.528	10.546
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
Snopt (8)	1.191 ± 0.173	62.6± 19.8	1.730	4.962
Filter	1.116 ± 0.163	28.7± 7.3	1.753	5.004
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
Snopt (8)	1.243 ± 0.101	232.4± 38.4	1.427	3.439
Filter	1.298 ± 0.238	52.6± 36.4	1.511	3.874
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
Snopt (7)	1.393 ± 0.236	477.1±144.6	1.456	3.527
Filter	1.333 ± 0.254	64.7± 12.9	**1.407**	**3.398**

TABLE V
AVERAGE RESULTS FOR 4 QSAR DATASETS

Method	Objective	Time	MAD	MSE
aquasol				
Unc. Grid	0.700 ± 0.103	17.2± 0.6	0.669	0.992
Con. Grid	0.754 ± 0.095	1396.5± 10.5	**0.855**	**1.634**
Filter	0.532 ± 0.059	594.0±272.8	**0.700**	1.021
BBB				
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	0.174 ± 0.011	371.3±452.4	0.336	0.185
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	0.220 ± 0.024	129.0± 64.4	**0.418**	**0.289**
CCK				
Unc. Grid	0.804 ± 0.060	12.0± 0.5	0.975	1.493
Con. Grid	0.777 ± 0.068	1173.6± 32.3	**1.234**	**2.425**
Filter	0.516 ± 0.036	217.3±101.9	1.006	1.534

Evaluating the problem from an optimization standpoint, FILTER is the preferred approach since it always find solutions with strictly lower objective values than either of the grid method. The difficulty of the underlying bilevel optimization problem is underscored by the fact that the greedy Con.Grid search in Section VI.C 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 significantly outperforms

Unc.Grid on the cancer data and does as well on the remaining three datasets. Thus the results suggest that bilevel programming approach is resistant to overfitting and can lead to significantly improved results on some but not all data sets. As expected, the computational times for FILTER are between those of Unc.Grid and Con.Grid.

IX. Discussion

Cross validation is an optimization problem of some sort that aims to identify the hyper-parameters of the model such that a generalization error is minimized. Grid search tackles this problem by discretizing the parameter space and searching in a small finite set. The proposed NLP formulation relaxes the problem by, among other things, introducing a tolerance in the training step, thus allowing an enlarged parameter space to be searched more effectively.

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 the NLP methods yield comparable if not better results than current methods. The computational results presented here are very preliminary and additional testing on real and synthetic problems is in progress.

From a machine-learning perspective, the novel bilevel programming opens up many possibilities. We can extend the approach to bilevel cross validation or bootstrapping combined with any of the widely used convex optimization formulations for a wide variety of tasks including classification, regression, novelty detection, and multitask learning. The ability to optimize a large number of parameters allows one to consider new forms of models and regularization. Here we have tested two types: addition of box constraints and a relaxation of exact cross validation. We can easily envision the treatment of other additions such as a term to reduce the variance in the weights across the folds and variants of the basic model.

From an optimization perspective, the novel formulation introduces many challenges but offers great promises. The improvement of the optimization methods for solving the bilevel machine learning problems is presently under investigation. Future results in this area will be reported elsewhere.

Acknowledgment

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 Stan Uryasev for sharing his suggestion of a bounded-variable approach to regression.

References

[1] A. Demiriz, K. Bennett, C. Brenman, and M. Embrechts. Support Vector Regression Methods in Cheminformatics. *Computer Science and Statistics* 33 (2001) available at http://www.galaxy.gmu.edu/interface/I01/I2001HTML.

[2] J. Bi, K. Bennett, M. Embrechts, C. Breneman, and M. Song. Dimensionality reduction via sparse support vector machines. *Journal on Machine Learning Research* 3 (2003) 1229-1243.

[3] J. Bracken and J. McGill. Mathematical programs with optimization problems in the constraints. *Operations Research* 21 (1973) 37-44.

[4] R. Caruana. Multitask learning. *Machine Learning* 28 (1997) 41–75.

[5] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning* 20 (1995) 273–297.

[6] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning* 46 (2002) 131–159.

[7] S. Dempe. *Foundations of Bilevel Programming*. Kluwer Academic Publishers (Dordrecht 2002).

[8] S. Dempe. Annotated bibliography on bilevel programming and mathematical programs with equilibrium constraints. *Optimization* 52 (2003) 333–359.

[9] K. Duan, S. Keerthi, and A. Poo. Evaluation of simple performance measures for tuning SVM hyperparameters. *Neurocomputing* 51 (2003) 41–59.

[10] T. Evgeniou and M. Pontil. Regularized multi–task learning. In *Proceedings of the 2004 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2004) pp. 109–117.

[11] F. Facchinei and J.S. Pang. *Finite-Dimensional Variational Inequalities and Complementarity Problems*, Springer-Verlag (New York 2003).

[12] R. Fletcher and S. Leyffer. Nonlinear programming without a penalty function. *Mathematical Programming* 91 (2002) 239-270.

[13] R. Fletcher and S. Leyffer. User manual for FilterSQP. Department of Mathematics, University of Dundee (Updated March 1999) [http://www-unix.mcs.anl.gov/leyffer/papers/SQP_manual.pdf].

[14] P.E. Gill, W. Murray and M.A. Saunders. *User's guide for SNOPT Version 6: A Fortran package for large-scale nonlinear programming*, Systems Optimization Laboratory, Stanford University (December 2002) [http://www.cam.ucsd.edu/ peg/papers/sndoc6.pdf].

[15] G. Golub, M. Heath, and G. Wahba. Generalised cross - validation as a method for choosing a good ridge parameter, *Technometrics* 21 (1979) 215–223.

[16] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning* 46 (2002) 389–422.

[17] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research* 3 (2003) 1157–1182.

[18] T. Hastie and S. Rossett. The entire regularization path for the support vector machine. *Journal of Machine Learning Research* 5 (2004) 1391-1415.

[19] G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, M. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Maching Learning Research* 5 (2004) 27–72.

[20] Z.Q. Luo, J.S. Pang, and D. Ralph. *Mathematical Programs With Equilibrium Constraints*, Cambridge University Press, Cambridge, England (1996).

[21] M. Momma and K. Bennett. A pattern search method for model selection of support vector regression. In *Proceedings of the Second SIAM International Conference on Data Mining*. R. Grossman, J. Han, V. Kumar, H. Mannila, and R. Motwani, Editors, SIAM Publications (2002).

[22] C. Ong, A. Smola, R. Williamson. Learning the kernel with hyperkernels. *Journal of Machine Learning Research* 6 (2005) 1043–1071.

[23] J. Outrata, M. Kocvara, and J. Zowe. *Nonsmooth Approach to Optimization Problems with Equilibrium Constraints: Theory, Applications and Numerical Results*. Kluwer Academic Publishers (Dordrecht 1998).

[24] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press (Cambridge 2004).

[25] B. Schölkopf, R. Williamson, A. Smola, J. Shawe-Taylor, and J. Platt. Support vector method for novelty detection. *Advances in Neural Information Processing Systems* 12 (2000) 582–588.

[26] V.N. Vapnik. *The Nature of Statistical Learning Theory*. Second Edition. Springer-Verlag (New York 2000).