2009 IEEE International Conference on Data Mining Workshops

Nonsmooth Bilevel Programming for Hyperparameter Selection

Gregory M. Moore, Charles Bergeron, Kristin P. Bennett
Department of Mathematical Sciences
Rensselaer Polytechnic Institute, Troy, NY, USA
{mooregm,chbergeron}@gmail.com, bennek@rpi.edu

Abstract

We propose a nonsmooth bilevel programming method for training linear learning models with hyperparameters optimized via T-fold cross-validation (CV). This algorithm scales well in the sample size. The method handles loss functions with embedded maxima such as in support vector machines. Current practice constructs models over a predefined grid of hyperparameter combinations and selects the best one, an inefficient heuristic. Innovating over previous bilevel CV approaches, this paper represents an advance towards the goal of self-tuning supervised data mining as well as a significant innovation in scalable bilevel programming algorithms. Using the bilevel CV formulation, the lower-level problems are treated as unconstrained optimization problems and are replaced with their optimality conditions. The resulting nonlinear program is nonsmooth and nonconvex. We develop a novel bilevel programming algorithm to solve this class of problems, and apply it to linear least-squares support vector regression having hyperparameters C (tradeoff) and ϵ (loss insensitivity). This new approach outperforms grid search and prior smooth bilevel CV methods in terms of modeling performance. Increased speed foresees modeling with an increased number of hyperparameters.

1 Introduction

Regularized linear modeling methods such as support vector machines (SVM) [19] are powerful and practical frameworks for a wide variety of learning problems. The one downside is that hyperparameters must be selected, and model accuracy and robustness can be highly sensitive to these choices. For example, in linear support vector regression (SVR), the hyperparameters are the tradeoff C (that determines the relative weighting of empirical and structural risk terms) and the insensitivity ϵ (radius of loss-insensitivity tube) [7]. When modelling with kernel functions, additional hyperparameters are added (such as the width, σ, for the Gaussian kernel). Typically model selection (selecting hyperparameters) is left to the user and a policy-based approach or cross-validation strategy is used.

Cross-validation (CV) is the classic approach for estimating out-of-sample generalization error in modeling. The method involves breaking up the modelling set into T partitions. $T-1$ partitions are used to train the model, and the remaining partition is used to validate the model. This is repeated T times, so that each partition is used in validation once. The average validation error across the folds is returned. Cross-validation can be applied to arbitrary data mining problems, producing a good estimate of generalization error on new data. The model's generalization is reported using new data disjoint from the modelling set.

To perform model selection, the hyperparameter space is searched to optimize the validation error. A standard algorithm to solve this problem is to define a discrete grid over a sampling of hyperparameter values and calculate a model for each point on the grid. The grid point that gives the best average validation error is chosen to be the *best* model. This method has obvious theoretical weaknesses (working from a discrete sampling of possible hyperparameters, the *best* model found is undoubtedly not optimal) and computational inefficiencies (high accuracy requires a fine grid). Moreover, its computational complexity grows exponentially in the number of hyperparameters, rendering this method intractable for more complex modeling tasks. Despite its heuristic nature and high computational complexity, it remains widely used due to its simplicity.

Other learning methods use an intelligent search over the grid hyperparameters which typically alternate between solving a problem with fixed hyperparameters, taking a hyperparameter step in a direction of decrease, and then resolving the problem using the new fixed hyperparameters. These methods treat the model weights as an implicit function of the hyperparameters and then optimize the hyperparameters using gradient [6, 12] or gradient-free techniques [10, 15]. These alternating methods have the fundamental drawback that requires optimization of all T-folds at each step. Intuitively, strategies that simultaneously solve

978-0-7695-3902-7/09 $26.00 © 2009 IEEE
DOI 10.1109/ICDMW.2009.74

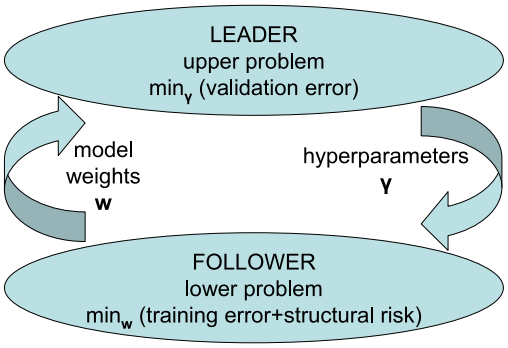

Figure 1. Bilevel cross-validation problem

for the learning models and hyperparameters are much more appealing than grid or other implicit function strategies because there is no need to fully solve the T-fold CV problems when the hyperparameters are far from optimal.

The idea of simultaneously solving for model hyperparameters and the problem parameters, formulated as a bilevel program [4, 8], has been gaining ground [1, 9, 13]. A bilevel program is a mathematical program that optimizes an objective subject to constraints which are also mathematical programs. The overall problem is called the *upper-level problem* whereas the mathematical programs within the constraints are referred to as the *lower-level problems*.

Cross-validation is naturally formulated as a bilevel program, as shown in Figure 1. In the Bilevel CV Problem (BCP), the lower-level problems are each of the T-fold CV problems which must be optimal for the hyperparameters. The objective of the upper-level is the validation error. The advantage is that the bilevel CV problem simultaneously solves for the hyperparameters in the upper-level problem and the problem parameters in the lower-level problems. This should allow faster convergence as the T-fold training problems are not solved to optimality when the upper-level problem is far from optimal.

The basic strategy behind solving a bilevel program involves replacing the lower-level problems with their optimality conditions, so as to reformulate it as a nonlinear program. Prior methods require the optimality conditions to be smooth so that the problem is reduced to a nonconvex nonlinear program (NLP). We introduce a new class of algorithms that permit nonsmooth optimality conditions. This is important because SVM-type losses with maximum functions embedded in the loss, such as the hinge loss for classification, ϵ-insensitive for regression, and ranking loss, are inherently nonsmooth. Typically, these losses are smoothed by adding constraints and slack variables for each data point (as done for standard SVM). The optimality or Karush-Kuhn-Tucker (KKT) conditions for the smoothed lower-level problems require the introduction of dual variables and associated primal, dual, and complementarity constraints. When the lower-level problems are replaced with their optimality conditions, the number of variables and constraints in the resulting NLP grows polynomially with the number of data points. The resulting NLPs can be solved successfully using general purpose and specialized solvers that produce solutions with good generalization but the the algorithms have not proven to be scalable to large problems to date [1, 2, 13].

For primal SVM problems, faster algorithms can be produced by directly handling the underlying SVM problems without reformulating them to remove any maximum functions in the losses. The fastest linearly scalable training methods are based on cutting plane algorithms that directly tackle the unconstrained but possibly nonsmooth learning problem without introducing slacks and constraints [11]. Similarly, no dual variables or constraints are introduced into the bilevel CV problem. Under bilevel formulations, each lower-level problem is an unconstrained optimization problem involving just the weights of the function. Because of the presence of the maximum function in the loss, the optimality conditions of each of the lower-level problems are nonsmooth and nonconvex equality constraints.

The most obvious advantage of the bilevel model selection is the ability to handle multi-parametric model selection in a continuous space. Bilevel optimization also allows for the systematic treatment of current and new models based on different loss and regularization functions. Most significantly, these advantages allow for improved model selection which ultimately leads to better generalization. The proposed bilevel programming algorithm works well on data mining datasets. The problems attempted are massive compared to those typically attempted in bilevel programming so this work represents significant advances in both data mining and in bilevel programming.

Section 3 reviews the formulation of CV in the bilevel setting. In Section 4, we show how the problem can be reformulated as nonsmooth nonlinear program and develop a penalty method for solving it. Section 5 describes a novel algorithm for polyhedral constrained nonconvex nonsmooth optimization which exploits the properties of the loss function. Least-squares SVR is explored in Section 6. Section 7 details the experimental design and results. Section 8 concludes with some highlights of our method and suggests further avenues of investigation.

2 Notation and definitions

Let $\mathbb{R}$ denote the set of real numbers, with $\mathbf{x} \in \mathbb{R}^n$ a vector and $\mathbf{x}'$ its transpose. The family of vector q-norms is defined as $\|\mathbf{x}\|_q = \sqrt[q]{\sum_{i=1}^n |\mathbf{x}_i|^q}$ for $q \geq 1$. When unspecified, $q = 2$ is assumed. Define $x_+ = \max(0, x)$. The convex hull written over a vector set $\mathcal{X}$ is denoted

$\mathrm{conv}(\mathcal{X})$. Consider a nonsmooth locally Lipschitz function $f : \mathbb{R}^n \to \mathbb{R}$. The subdifferential of f at $\bar{\mathbf{x}}$ is a set defined by $\partial f(\bar{\mathbf{x}}) = \mathrm{conv}(\mathbf{g} \in \mathbb{R}^n | \nabla f(\mathbf{x}) \to \mathbf{g}, \mathbf{x} \to \bar{\mathbf{x}})$ [17]. A subgradient $\mathbf{g}$ is any element of $\partial f(\bar{\mathbf{x}})$. The set of solutions of a minimization problem is denoted by argmin.

3 Bilevel cross-validation problem

A bilevel program is a mathematical program that optimizes an objective subject to constraints which are also mathematical programs. These constraints are referred to as the *lower-level problems* whereas the overall problem is called the *upper-level problem*. Bilevel optimization problems were first studied by Bracken and McGill [4]. A survey appears in [8].

Model selection via T-fold cross-validation can be written compactly as single bilevel program (BP) [1, 13]. The lower-level problems optimize the regularized training error to determine the best function for the given hyperparameters for each fold. The hyperparameters are the upper-level control variables. The upper-level objective is the validation error based on the optimal functions returned for each fold.

Formally, let $\Omega := \{\mathbf{x}_j, y_j\}_{j=1}^{\ell} \in \mathbb{R}^{n+1}$ denote a given labeled data set where $\mathbf{x}_j$ is a column vector. Set Ω is broken up into T equally sized divisions to perform T-fold cross-validation. For fold $t = 1, \ldots, T$, one of the divisions is used for the validation set, Ω_t, and the remaining $T-1$ divisions are put into the training set, $\overline{\Omega}_t$. Given some linear learning function, $\mathbf{x}'\mathbf{w} + b$, let $\mathbf{x}'\mathbf{w}_t + b_t$ be the learning function trained within the t-th fold, $\gamma \in \Gamma$ be the set of model hyperparameters, $\mathcal{S}$ be the regularization operator, $\mathcal{L}_{trn}$ be the lower-level training loss function, and $\mathcal{L}_{val}$ be the upper-level validation loss function. The bilevel program for T-fold cross-validation is:

$$\begin{aligned}
\min_{\mathbf{w}_1,\ldots,\mathbf{w}_T,b_1,\ldots,b_T,\gamma} \quad & \sum_{t=1}^{T} \frac{1}{|\Omega_t|} \sum_{(\mathbf{x}_j,y_j)\in\Omega_t} \mathcal{L}_{val}\left(y_j, \mathbf{x}_j'\mathbf{w}_t - b_t; \gamma\right) \\
& \qquad \textbf{(upper level)} \\
\text{subject to} \quad & \gamma \in \Gamma \\
& \text{for } t = 1, \ldots, T : [\mathbf{w}_t, b_t] \in \ldots \\
\underset{\mathbf{w},b}{\mathrm{argmin}} & \left\{ \mathcal{S}(\mathbf{w}, \gamma) + \sum_{(\mathbf{x}_j,y_j)\in\overline{\Omega}_t} \mathcal{L}_{trn}\left(y_j, \mathbf{x}_j'\mathbf{w} - b; \gamma\right) \right\}. \\
& \qquad \textbf{(lower level)}
\end{aligned} \tag{1}$$

The bilevel program is challenging to solve in this form. As discussed in the introduction, the BCP it is typically reformulated into some sort of nonlinear programming problem. Sections 4 and 5 give a novel reformulation of the model and a novel algorithm for solving the reformulation.

The general approach developed in this paper is applicable to general convex loss functions including the nonsmooth ones used in SVM. However, for a simpler exposition, we focus on lower-level regularization functions and training losses that are differentiable but not necessarily twice continuously differentiable. For example, the least-squares SVR with empirical risk terms $\frac{1}{2}\max\left(|\mathbf{x}'\mathbf{w} - y| - \epsilon, 0\right)^2$ and structural risk $\frac{1}{2}\mathbf{w}'\mathbf{w}$ is addressed in this paper. This restriction includes many well-known linear machine learning algorithms. Future investigations will be made into handling other losses such as absolute value training errors and 1-norm regularization.

4 Nonsmooth penalty bilevel problem

In this section, we introduce the nonsmooth penalty bilevel problem (PBP) specifically designed to exploit the structure of the CV BP. We assume that the lower-level problems are continuous and convex, and that each has a lower-bound and a minimum that is attained for all choices of hyperparameters. Each lower-level problem is unconstrained and has an objective that is once differentiable but not necessarily twice differentiable. For simplicity of presentation we drop the bias term b but all results include a bias term and full details of the algorithm with the bias term can be found at `http://www.rpi.edu/~bennek/bilevel/`.

Let L_{val}^t be a convex subdifferentiable function of $(\mathbf{w}_t, \gamma)$. Let L_{trn}^t (assume now for compactness that L_{trn}^t is the lower level objective) be differentiable functions of $(\mathbf{w}_t, \gamma)$ that are convex with respect to $\mathbf{w}_t$ for fixed γ. Further let the function $\partial_{\mathbf{w}_t} L_{trn}^t(\mathbf{w}_t, \gamma)$ be subdifferentiable with respect to $(\mathbf{w}_t, \gamma)$. Let $\Gamma := \{\gamma | A\gamma \le c\}$ be a polyhedral set. The problem becomes

$$\begin{aligned}
\min_{\mathbf{w}_1,\ldots,\mathbf{w}_T,\gamma} \quad & \sum_{t=1}^{T} L_{val}^t(\mathbf{w_t}, \gamma) \\
\text{s.t.} \quad & A\gamma \le c \\
& \mathbf{w_t} \in \underset{\mathbf{w}}{\mathrm{argmin}}\ L_{trn}^t(\mathbf{w}, \gamma) \quad \forall\, t = 1, \ldots T
\end{aligned} \tag{2}$$

We can further transform the problem into a nonlinear program with nonconvex constraints. A necessary and sufficient condition of the convex lower-level problem is that its gradient is zero at $\mathbf{w_t}$. Thus we can replace each lower-level problem with an equality constraint representing the optimality condition, yielding

$$\begin{aligned}
\min_{\mathbf{w}_1,\ldots,\mathbf{w}_T,\gamma} \quad & \sum_{t=1}^{T} L_{val}^t(\mathbf{w_t}, \gamma) \\
\text{s.t.} \quad & A\gamma \le c \\
& 0 = \partial_{\mathbf{w}_t} L_{trn}^t(\mathbf{w}_t, \gamma) \text{ for } t = 1, \ldots, T
\end{aligned} \tag{3}$$

Algorithm 1 Penalized bilevel problem algorithm

Choose an appropriate penalty parameters $\beta^t,\ t = 1..T$.
repeat
  Solve problem (4) using Algorithm 2.
  if $\|\partial_{\mathbf{w}_t} L^t_{trn}(\mathbf{w}_t, \gamma)\|^q_q \leq tol$ for all t **then**
    Exit, solution optimal for original problem (2).
  else
    for $t = 1 \ldots T$ **do**
      if $\|\partial_{\mathbf{w}_t} L^t_{trn}(\mathbf{w_t}, \gamma)\|^q_q \geq tol$ **then**
        Set $\beta^t = 2\beta^t$.
      end if
    end for
  end if
until algorithm exits

The constrained problem (3) is equivalent to (2), but the nonlinear constraints present a challenge. This difficulty is handled through conversion to the following penalty form.

$$\begin{array}{ll} \min\limits_{\mathbf{w}_1,\ldots,\mathbf{w}_T,\gamma} & \sum\limits_{t=1}^{T} L^t_{val}(\mathbf{w_t}, \gamma) + \sum\limits_{t=1}^{T} \dfrac{\beta^t}{q} \|\partial_{\mathbf{w}_t} L^t_{trn}(\mathbf{w_t}, \gamma)\|^q_q \\ \text{s.t.} & A\gamma \leq c. \end{array} \tag{4}$$

For $q = 2$, the problem uses a 2-norm penalty. For q=1, the problem uses a 1-norm penalty. In this paper we restrict ourselves to the 2-norm case for simplicity of presentation. For this case, as the penalty parameters, $\beta^t > 0$, approach infinity, the solution of the penalty problem (4) approaches the solution of the bilevel CV problem (2). To avoid numerical instability, the penalty parameters are progressive increased until a solution of the original bilevel CV problem is achieved with desired numeric precision. Note that we only assume that $\partial_{\mathbf{w}_t} L^t_{trn}(\mathbf{w}_t, \gamma)$ is subdifferentiable with respect $\mathbf{w}_t$ and γ, so the bilevel penalty problem is a nonsmooth, nonconvex program with polyhedral constraints.

The algorithm for finding locally optimal solutions of problem (2) by solving a sequence of penalty bilevel problems (4) is given in Algorithm 1.

5 Proximity control approximation algorithm

We exploit the special structure of the cross-validation PBP (4) in the design of Algorithm 2. This algorithm builds on the very general algorithm for unconstrained nonsmooth nonconvex optimization in [16], adding an additional subgradient optimization step that is essential to handling the constraints that are present. The inner loop attempt to identify a descent step $\mathbf{d}$ using increasingly large values of the proximity control parameter τ. Once found, point $(\mathbf{w}^i, \gamma^i)$ is updated, and a new $\mathbf{d}$ is sought, until local optimality is reached.

Algorithm 2 Proximity control approximation algorithm

Choose feasible $(\mathbf{w}^i, \gamma^i)$, starting τ and tolerance tol.
repeat
  Solve (9) for $\mathbf{g}, \lambda$.
  repeat
    Build approximation function $\hat{f}$ as in (6) about $(\mathbf{w}^i, \gamma^i)$.
    Solve subproblem (8) for direction $\mathbf{d}$.
    Expand proximity control parameter $\tau = 2\tau$.
  until $f(\mathbf{w}^i, \gamma^i) \leq f((\mathbf{w}^i, \gamma^i) + \mathbf{d})$
  Update $(\mathbf{w}^{i+1}, \gamma^{i+1}) = (\mathbf{w}^i, \gamma^i) + \mathbf{d},\ \ i = i + 1$.
until Problem (9) objective $< tol$.

The convex program uses an approximation of the objective plus a penalty term for proximity. Recall that the objective of PBP (4) (which we now denote as f), is decomposed into a sum of convex functions and a sum of squared normed nonconvex nonsmooth subdifferentiable functions $\|\partial_{\mathbf{w}_t} L^t_{trn}(\mathbf{w}_t, \gamma)\|^2$ (which we now call $\|H^t(\mathbf{w}_t, \gamma)\|^2$):

$$f(\mathbf{w}_t, \gamma) := \sum_{t=1}^{T} L^t_{val}(\mathbf{w}_t, \gamma) + \sum_{t=1}^{T} \frac{\beta^t}{2} \|H^t(\mathbf{w}_t, \gamma)\|^2 \tag{5}$$

At iteration i, the algorithm uses a convex approximation of f at the current iterate, $(\mathbf{w}^i, \gamma^i)$, based on linearizing the inner nonconvex functions H^t using their subgradients:

$$\begin{array}{l} \hat{f}(\mathbf{w}_t, \gamma) := \sum_{t=1}^{T} L^t_{val}(\mathbf{w}_t, \gamma) \ldots \\ + \sum_{t=1}^{T} \frac{\beta^t}{2} \|H^t(\mathbf{w}^i, \gamma^i) + \partial_{\mathbf{w}_t} H^t(\mathbf{w}_t, \gamma)'(\mathbf{w}_t - \mathbf{w}^i_t) \ldots \\ + \partial_\gamma H^t(\mathbf{w}_t, \gamma)'(\gamma - \gamma^i)\|^2. \end{array} \tag{6}$$

Note that when H is nonsmooth, then the subgradient is not unique. The best subgradient would be the one that leads to the steepest feasible descent direction. The selection of this subgradient is discussed in detail later within this section.

The convex subproblem has the form

$$\begin{array}{ll} \min_{\mathbf{w},\gamma} & \hat{f}(\mathbf{w}, \gamma; \mathbf{w}^i, \gamma^i) + \frac{\tau}{2}\|\mathbf{w} - \mathbf{w}^i\|^2 + \frac{\tau}{2}||\gamma - \gamma^i\|^2 \\ \text{s.t.} & A\gamma \leq c. \end{array} \tag{7}$$

or equivalently after a change of variables

$$\begin{array}{ll} \min_{\mathbf{d}} & \hat{f}([\mathbf{w}^i, \gamma^i] + \mathbf{d}) + \frac{\tau}{2}\|\mathbf{d}\|^2 \\ \text{s.t.} & [0\ \ A]\mathbf{d} \leq \left[\begin{array}{c} 0 \\ c \end{array} \right]. \end{array} \tag{8}$$

The approximation $\hat{f}$ is only locally valid and thus proximity control is required to tradeoff approximation accuracy versus step size. For a major iterate and associated approximation function, the algorithm loops on increasing values of the proximity control parameter (τ). Increasing values of τ decreases $\mathbf{d}$ and thus the approximation function will better

approximate the true function. When a decrease in the original function is found, then $(\mathbf{w}^i, \gamma^i)$ and the approximation function are updated and τ is reset.

For the case of nonsmooth f, two issues of the algorithm remain: selecting the best subgradient for the approximation and detecting necessary conditions for optimality. We prove that these both reduce to solving the same optimality problem for a fixed $\mathbf{w}, \gamma$:

$$\begin{array}{ll} \min_{\mathbf{g},\lambda} & \|\mathbf{g} + [0 \quad A_I]'\lambda\|^2 \\ \text{s.t.} & \lambda \geq 0 \\ & \mathbf{g} \in \partial f(\mathbf{w}, \gamma) \end{array} \tag{9}$$

where $I = \{i | A_i\gamma = c_i\}$ is the index set of active constraints and A_I is the corresponding submatrix.

Theorem 1 (Necessary Optimality Condition of PBP)**.** *Let $\mathbf{w}^*, \gamma^*$ be a local minimum of problem (4) and let $\mathbf{g}^*, \lambda^*$ solve problem (9), then $0 = \mathbf{g}^* + [0 \quad A_I]'\lambda^*$.*

Proof. Specializing Proposition 4.7.3 in [3] to the normal cone of polyhedral constraints yields the necessary optimality condition: $0 \in \{\partial f(\mathbf{z}*) + A_I'\lambda | \lambda \geq 0\}$. Thus the lower bound of problem (9) is feasible and thus optimal. □

This problem also returns the direction of maximum decrease.

Theorem 2 (Greatest Feasible Descent Direction)**.** *Let $\mathbf{g}^*, \lambda^*$ be a nondegenerate solution of problem (9). If $0 \neq \mathbf{g}^* + [\mathbf{0} \; A_I]']\lambda^*$, then $\mathbf{g}^*, \mathbf{d}$ solves the optimal feasible descent direction problem:*

$$\begin{array}{lll} \min_{g \in \partial f} & \min_{\mathbf{d}} & \mathbf{g}'\mathbf{d} \\ & \text{s.t.} & [0 \;\; A_I]\mathbf{d} \leq 0 \\ & & \|\mathbf{d}\| \leq 1. \end{array} \tag{10}$$

The details of the proof is found at `http://www.rpi.edu/~bennek/bilevel/`.

6 Least-squares support vector regression with multiple groups

Here we show how the method is applied to least-squares support vector regression (LS-SVR) [18]. Least-squares SVMs typically have maximization functions embedded in the loss function. For LS-SVM, the ϵ-insensitive regression loss is $\frac{1}{2}\max(|\mathbf{x}'\mathbf{w} - y| - \epsilon, 0)^2$. This loss is differentiable but not twice differentiable. The resulting penalized bilevel program has a nonsmooth nonconvex objective that is only subdifferentiable.

Since BP methods permit many hyperparameters, we examine a variant of LS-SVR with multiple groups of data requiring different hyperparameters. Datasets containing K subsets ($\nu_k \in \Omega, \; k = 1..K$) of the data which are known to be of varying error distributions should not be treated as a homogeneous set while training. For example, groups of differing quality within a same dataset can arise for a number of reasons: data may be collected from different sources, following different data collection techniques, or at different times, thus affecting the data matrix and/or the response vector.

Our multigroup LS-SVR loss function allows different weights C_k and insensitivities ϵ_k to be used for each different set. For example, if the data can be separated into varying degrees of *goodness*, multiple hyperparameters should upweight higher quality groups and downweight lower quality groups when calculating a model. LS-SVR uses a structural risk of $\frac{1}{2}\mathbf{w}_t'\mathbf{w}_t$ and the following validation and training loss functions for each fold $t = 1..T$:

$$\mathcal{L}_{val}\left(y_j, \mathbf{x}_j, \mathbf{w}_t; C, \epsilon\right) = \sum_{k=1}^{K} \frac{1}{2|\Omega_t \cap \nu_k|} \sum_{(\mathbf{x}_j, y_j) \in (\Omega_t \cap \nu_k)} \left(\mathbf{x}_j'\mathbf{w}_t - y_j\right)^2 \tag{11}$$

$$\mathcal{L}_{trn}\left(y_j, \mathbf{x}_j, \mathbf{w}_t; C, \epsilon\right) = \frac{1}{2}\sum_{k=1}^{K} \frac{C_k}{|\bar{\Omega}_t \cap \nu_k|} \sum_{(\mathbf{x}_j, y_i) \in (\bar{\Omega}_t \cap \nu_k)} \left(|\mathbf{x}_j'\mathbf{w} - y_j| - \epsilon_k\right)_+^2. \tag{12}$$

The optimality condition to be penalized for each lower-level problem, $t = 1 \ldots T$, is:

$$0 \in \left\{ \mathbf{w}_t + \sum_{k=1}^{K} \frac{C_k}{|\bar{\Omega}_t \cap \nu_k|} \sum_{(\mathbf{x}_j, y_j) \in (\bar{\Omega}_t \cap \nu_k)} \mathbf{x}_j' Q_{t,k}^j \right\} \tag{13}$$

and $Q_{t,k}^j = \Big(\left(\mathbf{x}_j'\mathbf{w}_t - y_j - \epsilon_k\right)_+ - \left(-\mathbf{x}_j'\mathbf{w}_t + y_j - \epsilon_k\right)_+\Big)$. Using this optimality condition gives a nonsmooth nonconvex NLP with bound constraints:

$$\begin{array}{ll} f(\mathbf{w}_t, C, \epsilon) = & \frac{1}{T}\sum_{t=1}^{T}\sum_{k=1}^{K} \frac{1}{|\Omega_t \cap \nu_k|} \sum_{(\mathbf{x}_j, y_j) \in (\Omega_t \cap \nu_k)} \left(\mathbf{x}_j'\mathbf{w}_t - y_j\right)^2 \\ & + \sum_{t=1}^{T} \beta^t \left\| \mathbf{w}_t + \sum_{k=1}^{K} \frac{C_k^i}{|\bar{\Omega}_t \cap \nu_k|} \sum_{(\mathbf{x}_j, y_j) \in (\bar{\Omega}_t \cap \nu_k)} \mathbf{x}_j' Q_{t,k}^j \right\|_2^2 \\ \text{s.t.} & C_{LB} \leq C \leq C_{UB} \\ & \epsilon_{LB} \leq \epsilon \leq \epsilon_{UB} \end{array} \tag{14}$$

The nonsmooth nonconvex portions of the penalty terms are approximated with their first order linear approximations. Thus assuming a current point $(\mathbf{w}^i, C^i, \epsilon^i)$ and choosing a subgradient $P_{t,k}^{j,i} \in \partial_{\mathbf{w}_t, \epsilon}\left((\mathbf{x}_j'\mathbf{w}_t^i - y_j - \epsilon_k)_+ - (-\mathbf{x}_j'\mathbf{w}_t^i + y_j - \epsilon_k)_+\right)$, the approximation problem with proximity control is to

minimize $\hat{f}(\mathbf{w}_t,\ C,\ \epsilon; \mathbf{w}_t^i,\ C^i,\ \epsilon^i) =$

$$\begin{aligned}
&\tfrac{1}{T}\sum_{t=1}^{T}\sum_{k=1}^{K}\frac{1}{|\Omega_t \cap \nu_k|}\sum_{(\mathbf{x}_j,y_j)\in(\Omega_t\cap\nu_k}\left(\mathbf{x}_j'\mathbf{w}_t - y_j\right)^2 \dots \\
&\quad +\sum_{t=1}^{T}\beta^t\left\|\mathbf{w}_t + \sum_{k=1}^{K}\frac{1}{|\bar{\Omega}_t \cap \nu_k|}\sum_{(\mathbf{x}_j,y_j)\in(\bar{\Omega}_t\cap\nu_k)} C_k\mathbf{x}_j' Q_{t,k}^{j,i}\right. \\
&\quad \left. +C_k^i\mathbf{x}^j\left(P_{t,k}^{j,i}\left([\mathbf{w}_t,\epsilon]-[\mathbf{w}_t^i,\epsilon^i]\right)\right)\right\| \\
&\quad +\tau\left\|[\mathbf{w},C,\epsilon]-\left[\mathbf{w}^i,C^i,\epsilon^i\right]\right\|_2^2
\end{aligned} \tag{15}$$

subject to $C_{LB} \le C \le C_{UB}$, $\epsilon_{LB} \le \epsilon \le \epsilon_{UB}$ and $Q_{t,k}^{j,i} = \left(\left(\mathbf{x}_j'\mathbf{w}_t^i - y_j - \epsilon_k^i\right)_+ - \left(-\mathbf{x}_j'\mathbf{w}_t^i + y_j - \epsilon_k^i\right)_+\right)$. Problem 15 is convex and smooth for fixed $(\mathbf{w}^i, C^i, \epsilon^i)$ and choice of P, but it has freedom in choosing which subgradient to use in the Taylor series approximation. In order to achieve a robust algorithm, problem (9) must be solved to select the proper value of $P_{t,k}^{j,i}$ by finding the subgradient leading to the direction of greatest feasible descent. Further detail are given for choosing P at `http://www.rpi.edu/~bennek/bilevel/`.

7 Experimental results

We report two sets of experiments, applying PBP against various state-of-the-art approaches.

All results were computed using Matlab 7.6 (`http://www.mathworks.com/`) on a 2.4 GHz quad-core computer running Ubuntu Linux, except as noted. The quadratic solver used throughout the PBP algorithm is Adrian Wills' `qpas` (QPC project, `http://sigpromu.org/quadprog/`). Parameters used throughout the algorithm were $tol = 10^{-3}$, $\tau_0 = 10$ and $\beta_t = 1$.

7.1 Pyruvate kinase dataset

The first set of experiments considers a larger cheminformatics problem. This dataset is related to drug design, that of identifying compounds that make suitable medicines. The effect of 51441 small drug-like compounds on pyruvate kinase was assayed using the quantitative high-throughput screening (qHTS) [14] technique. Pyruvate kinase is an interesting enzyme to study as it is a frequent target in cancer therapy. The experimental data is distributed through PubChem (`http://pubchem.ncbi.nlm.nih.gov/`) under BioAssay identifier (AID) 361. The resulting dose-response information was used to compute for each compound the half-maximal activity concentration (AC_{50}), which is the concentration at which 50% of the pyruvate kinase is inhibited. The reliability in calculating AC_{50} depends on the extent to which the dose-response captures the inhibition process.

A total of 4766 compounds are categorized as follows: 15% are *reliable* (having activity at the highest tested concentration less than -70%), 40% are *intermediate* (with highest tested concentration activity between -70 and -30%), and 45% are *uncertain* (with highest tested concentration activity between -30 and -20%). Each category is assigned a distinct C and ϵ, resulting in a model possessing 6 hyperparameters. Compounds are separated into a modelling set (varying from 100 to 3000 samples) and test set (1766 samples). Both MOE (implemented within the Molecular Operating Environment software, Chemical Computing Group, `http://www.chemcomp.com/`) and RECON [5] features were computed. 851 features describe chemical properties for each compound. The features were median-centered, absolute deviation-scaled, and thresholded to range $[-5, 5]$. Principal components analysis was used to eliminate redundant features and reduce the dimensionality to 100.

Grid values for single C, ϵ were $\{0.0001, 0.001, 0.01, 0.1, 1, 10, 100\}$ and $\{0.001, 0.01, 0.1, 1\}$, respectively. The grid search over the single C and ϵ was run using Matlab 7.1 on a 2.33 GHz Windows XP system. The grid search with multiple Cs and ϵs used $C \in \{0.0001, 0.01, 1\}$ and $\epsilon \in \{0.01, 0.1, 1\}$ and was run on the previously mentioned linux computer. The training objective (average $\mathcal{L}_{\text{val}}$) used 5-fold CV. The results are the average of 10 different modeling and testing runs.

Table 1 demonstrates how PBP methods outperformed grid search on pyruvate kinase in every respect. PBP achieved a smaller generalization error in every case using little computational time. The PBP generalization error, objective function, and timing results were quite consistent as the training set size was increased from 100 to 3000 points. In contrast, both the grid generalization errors and objective functions ($\mathcal{L}_{\text{val}}$) are worse for smaller training set sizes. This indicates that the hyperparameter problem is critical and challenging on smaller training set sizes where regularization is important. The multiple C and ϵ problems converge more quickly for both grid and PBP but they don't always result in better generalizations and training objectives. This indicates that the underlying problems are easier but perhaps more prone to local minima since the validation error is slightly worse than the single C and ϵ case.

The PBP with single C and ϵ achieved the best generalization and but was more computationally challenging for PBP with a few slower converging runs raising the computational times. Since PBP is a subgradient method, it is sensitive to conditioning.

Method	Hold-out testing error	Average $\mathcal{L}_{\text{val}}$	Time (seconds)
100 pts			
GRID C ϵ	14.57 ± 9.40	1.81 ± 0.34	256±38.2
GRID CS ϵS	11.40 ± 10.31	10.31 ± 1.08	599±855
PBP C	0.65 ± 0.11	0.34 ± 0.09	7.2±9.5
PBP C ϵ	$\mathbf{0.52 \pm 0.03}$	0.34 ± 0.10	10.3±10.3
PBP CS ϵS	0.55 ± 0.07	0.34 ± 0.09	2.3±0.7
1000 pts			
GRID C ϵ	6.61 ± 9.50	1.41 ± 0.11	234±21.9
GRID CS ϵS	7.97 ± 1.01	7.04 ± 0.81	164±149
PBP C	0.60 ± 0.04	0.35 ± 0.03	2.8±0.2
PBP C ϵ	$\mathbf{0.51 \pm 0.02}$	0.34 ± 0.03	42.3±16.2
PBP CS ϵS	0.53 ± 0.05	0.35 ± 0.03	4.0±0.4
2000 pts			
GRID C ϵ	2.69 ± 6.21	1.33 ± 0.09	225±33.3
GRID CS ϵS	7.61 ± 1.78	6.51 ± 0.91	127±6.9
PBP C	0.56 ± 0.04	0.36 ± 0.02	2.9±0.2
PBP C ϵ	$\mathbf{0.51 \pm 0.02}$	0.34 ± 0.02	186±243
PBP CS ϵS	0.55 ± 0.04	0.36 ± 0.02	5.7±0.1
3000 pts			
GRID C ϵ	0.78 ± 0.14	1.26 ± 0.06	224±30.9
GRID CS ϵS	7.38 ± 0.59	6.21 ± 0.46	145±6.0
PBP C	0.56 ± 0.02	0.35 ± 0.02	3.4±0.2
PBP C ϵ	$\mathbf{0.51 \pm 0.01}$	0.33 ± 0.02	130±50.8
PBP CS ϵS	0.56 ± 0.05	0.35 ± 0.02	7.7±0.1

Table 1. Table reports results of pyruvate kinase data with 5-fold cross-validation average over 10 trials. Average and standard deviations of the hold-out testing error and the optimization objectives are reported.

Table 2. Hold-out testing results for QSAR data under 5-fold cross-validation averaged over 20 splits. Times are in seconds and are for runs on different computers.

Method	Time (sec.)	Hold-out testing error
Aquasol		
PBP C	1.9± 1.5	**0.86**
PBP C ϵ	152.8±438.6	0.86
GRID	17.1± 0.4	0.91
FILTER (SQP)	1253.0±533.7	0.97
SLAMS	137.8± 52.0	0.91
EZ-SLAMS	19.1± 3.3	0.91
Blood/Brain Barrier		
PBP C	0.4± 0.3	**0.18**
PBP C ϵ	147.2±364.7	0.19
GRID	13.4± 1.9	0.23
FILTER (SQP)	572.7±339.5	0.21
SLAMS	17.1± 9.8	0.23
EZ-SLAMS	8.0± 1.6	0.24
Cancer		
PBP C	0.8± 0.9	**0.26**
PBP C ϵ	437.4±371.5	0.30
GRID	10.3± 0.9	0.47
FILTER (SQP)	180.8± 64.3	0.34
SLAMS	25.5± 9.2	0.34
EZ-SLAMS	5.2± 1.1	0.34
Cholecystokinin		
PBP C	0.3± 0.3	1.53
PBP C ϵ	392.1±623.5	1.53
GRID	12.0± 0.4	1.63
FILTER (SQP)	542.3±211.5	**1.52**
SLAMS	35.1± 20.4	2.58
EZ-SLAMS	9.1± 1.3	2.57

7.2 QSAR datasets

These datasets construct quantitative structure activity relationship (QSAR) models that predict the bioactivity of small molecules. The four QSAR datasets considered are Aquasol, Blood/Brain Barrier, Cancer, and Cholecystokinin [2]. To facilitate direct comparison with existing approaches, we reproduced the experimental design of [2]. 5-fold cross-validation was repeated 20 times. For each split, Aquasol has 100 modelling examples and 97 testing examples; Blood/Brain Barrier has 60 modelling examples and 2 testing examples; Cancer has 40 modelling examples and 6 testing examples; and finally Cholecystokinin has 60 modelling examples and 6 testing examples. Table 2 reports the results for 6 different bilevel cross-validation algorithms. The average mean square test set error obtained on the 20 partitions are reported as well as the wall clock computation time. Note that the non-PBP results are from [2], and thus times are on different computers and only provide a sense of the algorithms' computational cost. All of the methods used 5-fold CV.

For each dataset in Table 2, *PBP C* optimizes a single C, thus using a simple least squares loss, *PBP C* ϵ optimizes C and ϵ using the LS-SVR loss. Polyhedral constraints were $10^{-3} \leq C \leq 10^{3}$ and $3^{-3} \leq \epsilon \leq \text{std}(y)$. The remaining four rows reprint results published in [2]. All four optimized both C and ϵ, and chose the least absolute deviation as the upper-level validation error and the ϵ-insensitive loss with 2-norm regularization for the lower-level problems. A grid search over the parameters C and ϵ is labelled *Grid*. *FILTER (SQP)* is a general purpose sequential quadratic programming algorithm. Two variants of successive linearization algorithms for bilevel programming are *SLAMS* and *EZ-SLAMS*.

The generalization errors in Table 2 report that PBP finds better models. The *PBP C* method that selects a single C takes negligible computing time compared to both grid and the prior bilevel methods on these small problems. Both PBP approaches returned at-par generalization errors, but optimizing the hyperparameter ϵ takes more computational time.

8 Conclusion

This work demonstrates a new method for solving the model selection problem without requiring extraneous variables and constraints. We have developed a nonsmooth bilevel programming approach for hyperparameter selection via cross-validation and demonstrated that it quickly achieves better generalization results than grid search and prior bilevel programming methodologies on two sets of cheminformatics problem solved using LS-SVR. Our novel general purpose nonsmooth bilevel programming algorithm helps bring the recent advances in nonsmooth optimization that have proven so successful for primal SVM problems to the model selection problem. The proposed nonsmooth methods scale well in the size of the data because they do not have to resort to smoothing techniques commonly used in SVM that add additional variables and constraints. The algorithm presented can be used with any subdifferentiable validation loss and once differentiable (but not necessarily twice differentiable) training loss. With modification of the subgradient search problem, the approach can be extended to all the subdifferentiable losses used in SVM, but this is left to future papers.

This work represents a preliminary study into the capabilities of bilevel programming for model selection in data mining. Our goal is to fully integrate the model selection into model training. Future work includes investigating novel data mining and learning frameworks based on models with many hyperparameters including those with kernels. One could imagine addressing problems in data cleaning, multitask learning, missing/uncertain data, and model selection across different loss functions. New models are sure to create new computational challenges. Further improvements to the PBP algorithm are possible to enhance its efficiency and robustness to local minima.

Acknowledgments

This work is supported in part by the Office of Naval Research (Grant number N00014-06-1-0014). Micheal Krein generated the descriptors for the pyruvate kinase dataset, under support from the Rensselaer Exploratory Center for Cheminformatics Research (RECCR, `http://reccr.chem.rpi.edu/`).

References

[1] K. P. Bennett, J. Hu, X. Ji, G. Kunapuli, and J.-S. Pang. Model selection via bilevel optimization. *International Joint Conference on Neural Networks*, pages 1922–1929, 2006.

[2] K. P. Bennett, G. Kunapuli, J. Hu, and J.-S. Pang. Bilevel optimization and machine learning. In J. Zurada and et al, editors, *Computational Intelligence: Research Frontiers :IEEE WCCI 2008, Hong Kong, China, June 1-6, 2008 : Plenary/invited Lectures*, volume 5050 of *Lecture Notes in Computer Science*, pages 25–47. Springer, 2008.

[3] D. Bertsekas, A. Nedic, and A. Ozdaglar. *Convex analysis and optimization*. Athena, 2003.

[4] J. Bracken and J. McGill. Mathematical programs with optimization problems in the constraints. In *Operations Research*, volume 21, pages 37–44, 1973.

[5] C. Breneman and M. Rhem. A QSPR analysis of hplc column capacity factors for a set of high-energy materials using electronic van der waals surface property descriptors computed by the transferable atom equivalent method. *Journal of Computational Chemistry*, 18(2):182–197, 1997.

[6] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1–3):131–159, 2002.

[7] C. Cortes and V. Vapnik. Support-vector networks. pages 273–297, 1995.

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

[9] C. Do, C.-S., and N. Ng. Efficient multiple hyperparameter learning for log-linear models. In *Advances in Neural Information Processing Systems 20*, pages 377–384, 2008.

[10] T. Eitrich and B. Lang. Efficient optimization of support vector machine learning parameters for unbalanced datasets. *Journal of Computational and Applied Mathematics*, 196(2):425–436, 2006.

[11] T. Joachims. Training linear SVMs in linear time. In *Knowledge Discovery and Data Mining*, pages 217–226, 2006.

[12] S. Keerthi, V. Sindhwani, and O. Chapelle. An efficient method for gradient-based adaptation of hyperparameters in svm models. In *Neural Information Processing Systems 18*, pages 673–680, 2006.

[13] G. Kunapuli, K. P. Bennett, and J.-S. Pang. Bilevel model selection for support vector machines. In P. Pardalos and P. Hansen, editors, *Data Mining and Mathematical Programming*, volume 45, pages 129–158. AMS, 2008.

[14] J. Inglese, et al. Quantitative high-throughput screening: A titration-based approach that efficiently identifies biological activities in large chemical libraries. *Proceedings of the National Academy of Sciences*, 103(31):11473–11478, 2006.

[15] M. Momma and K. P. Bennett. A pattern search method for model selection of support vector regression. In *Proceedings of the Second SIAM International Conference on Data Mining*, 2002.

[16] D. Noll, O. Prot, and A. Rondepierre. A proximity control algorithm to minimize nonsmooth and nonconvex functions. *Pacific Journal on Optimization*, pages 571–604, 2008.

[17] A. Ruszczyński. *Nonlinear Optimization*. Princeton University Press, Princeton, NJ, 2006.

[18] J. Suykens, J.Brabanter, and T. V. Gestel. *Least Squares Support Vector Machines*. World Scientific, 2002.

[19] V. N. Vapnik. *The Nature of Statistical Learning Theory, Second Edition*. Springer-Verlay New York, Inc., New York, 2000.