Decomposition Algorithms for Training Large-scale Semiparametric Support Vector Machines

Sangkyun Lee and Stephen J. Wright

Computer Sciences Department, University of Wisconsin-Madison,
1210 W. Dayton St., Madison, WI 53706, USA
{sklee,swright}@cs.wisc.edu

Abstract. We describe a method for solving large-scale semiparametric support vector machines (SVMs) for regression problems. Most of the approaches proposed to date for large-scale SVMs cannot accommodate the multiple equality constraints that appear in semiparametric problems. Our approach uses a decomposition framework, with a primal-dual algorithm to find an approximate saddle point for the min-max formulation of each subproblem. We compare our method with algorithms previously proposed for semiparametric SVMs, and show that it scales well as the number of training examples grows.

Key words: semiparametric SVM, regression, decomposition, primal-dual gradient projection

1 Introduction

Support Vector Machines (SVMs) are the most widely used nonparametric methods in machine learning, which aims to find a function that performs well in classifying or fitting given data. The power of SVM lies in the fact that it does not require the user to define the class of functions from which the observations might have been generated. In a sense, this is also a weakness, in that prior knowledge of the function class is often available for use. *Semiparametric* SVM formulations introduce parametric components into the model of the classifying / regression function, alongside the nonparametric contribution. The basis functions in the parametric part of the model can be chosen to embed prior knowledge and can be used for analyzing the effects of certain covariates, thus giving semiparametric SVM the potential advantages of both parametric and nonparametric methods.

Despite the benefits, semiparametric models have not drawn much attention from the machine learning community, possibly in part because the optimization problems arising from semiparametric SVMs are harder to solve than those generated by standard SVMs. This paper describes an efficient approach for finding solutions to large-scale semiparametric SVM problems. We focus on the formulation of semiparametric SVM regression first introduced in [1], which gives rise to

a dual problem which is a convex quadratic program (QP) with several equality constraints as well as bound constraints.

To motivate our description of solvers for semiparametric SVMs, we discuss first the state of the art for solvers that tackle the standard SVM dual formulation, which is

$$\min_{\mathbf{x}} \frac{1}{2}\mathbf{x}^T\boldsymbol{Q}\mathbf{x} + \boldsymbol{p}^T\mathbf{x} \qquad \text{s.t.} \quad \mathbf{y}^T\mathbf{x} = 0, \qquad \mathbf{0} \le \mathbf{x} \le C\mathbf{1} \ , \tag{1}$$

where $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{1} := (1, 1, \ldots, 1)$ are column vectors of length n. Many effective algorithms for this problem solve a sequence of subproblems, each of which updates some subvector of $\mathbf{x}$ while leaving the remaining elements unchanged. These algorithms can be categorized into two distinct groups. In the first group, the subvector is very short, typically containing just two components. Since the subproblem can be solved analytically for such a small number of variables, no numerical solver is needed. The subproblems are inexpensive, but many iterations are usually needed to reach a solution with acceptable quality. Sequential Minimal Optimization (SMO) [2] and its variants such as LIBSVM [3] fall into this category. In the second group of solvers, the subvectors are longer, requiring the subproblems to be solved with a QP solver that exploits the structure of the application. Although we face the burden of designing an efficient, robust QP solver, methods in the second group often show faster convergence than those in the first group. Successful instances of methods in the second group include SVM^{light}[4] and GPDT [5, 6]. The QP solvers used in the second group can be applied to the full problem, thus solving it in one "outer" iteration, though this approach is not usually effective for large data sets.

In general, the methods in both groups discussed above are specialized to handle the single equality constraint in (1) along with the bound constraints. The analytic subproblem solution in SMO can be acquired only when the subproblem has up to one (or two in case of the modified SMO [7]) equality constraint. The subproblem selection algorithm of SVM^{light} strongly depends upon the existence of a single equality constraint; the same is true of GPDT, which uses a projection algorithm from [8]. Semiparametric SVMs, however, require solution of the following generalization of (1):

$$\min_{\mathbf{x}} F(\mathbf{x}) := \frac{1}{2}\mathbf{x}^T\boldsymbol{Q}\mathbf{x} + \boldsymbol{p}^T\mathbf{x} \qquad \text{s.t.} \quad \boldsymbol{A}\mathbf{x} = \boldsymbol{b}, \qquad \mathbf{0} \le \mathbf{x} \le C\mathbf{1}, \tag{2}$$

where $\boldsymbol{A} \in \mathbb{R}^{K\times n}$ and $\boldsymbol{b} \in \mathbb{R}^K$, where $K \ge 1$ is the number of parametric basis functions that we wish to include in the model. For semiparametric SVM regression, Smola, Frieß, and Schölkopf [1] proposed to apply a primal-dual interior point method based on the code LOQO. The size of problems that can be handled is thus limited by the need to perform a full evaluation of the matrix $\boldsymbol{Q}$ and the need for repeated factorizations of matrices of about this size. (The approach could however be used as the inner loop of a decomposition method in the second group discussed above.) Kienzle and Schölkopf [9] suggested a *Minimal Primal Dual* (MPD) algorithm. This algorithm use a variant of the method of

multipliers to formulate a sequence of convex quadratic programs of dimension n with bound constraints only (no equalities), which are solved by a method that selects a single component for updating at each iteration. (In this sense, it is akin to the methods in the first group described above.) We give further details on MPD as we introduce our methods below. This approach does not scale well as the size n of the problem grows, but its performance can be improved by embedding it in a decomposition framework, as described below. We include both MPD and its decomposition variants in our computational tests of Sect. 5.

In this paper, we propose an approach that is related to MPD but that differs in several ways. First, it is a primal-dual approach; we alternate between steps in a subvector of $\mathbf{x}$ and steps in the Lagrange multipliers for the constraints $\boldsymbol{A}\mathbf{x} = \boldsymbol{b}$. Second, subvectors of $\mathbf{x}$ with more than 1 element are allowed. Third, two-metric gradient projection techniques are used in taking steps in the $\mathbf{x}$ components. Throughout, we take account of the fact that n may be very large, that $\boldsymbol{Q}$ cannot practically be computed and stored in its entirety, and that operations involving even modest-sized submatrices of $\boldsymbol{Q}$ are expensive.

We compare our approach computationally with MPD as stand-alone solvers, and also in a decomposition framework.

The remainder of the paper is structured as follows. In the next section, we define the semiparametric SVM regression problem and show that its dual has the form (2). Section 3 outlines the decomposition framework, while Sect. 4 describes the primal-dual method that we propose for solving the subproblems that arise from decomposition. Section 5 presents some computational results.

2 Semiparametric SVM Regression

We consider a regression problem for data $\{(\boldsymbol{t}_i, \boldsymbol{y}_i)\}_{i=1}^M$ where $\boldsymbol{t}_i \in \mathbb{R}^N$ are feature vectors and $\boldsymbol{y}_i \in \mathbb{R}$ are outcomes. We wish to find a function h that minimizes ϵ-insensitive loss function $\ell_\epsilon(h; \boldsymbol{t}, \boldsymbol{y}) := \max\{0, |\boldsymbol{y} - h(\boldsymbol{t})| - \epsilon\}$, while maximizing the margin as in [10]. Following [1, 9], we formulate the semiparametric SVM regression problem as follows:

$$\min_{\boldsymbol{w},\boldsymbol{\beta},\boldsymbol{\xi},\boldsymbol{\xi}^*} \quad \frac{1}{2}\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{w} + C\sum_{i=1}^{M}(\boldsymbol{\xi}_i + \boldsymbol{\xi}_i^*) \tag{3a}$$

$$\text{s.t.} \quad \boldsymbol{y}_i - \langle \boldsymbol{w}, \phi(\boldsymbol{t}_i)\rangle - \sum_{j=1}^{K}\boldsymbol{\beta}_j\psi_j(\boldsymbol{t}_i) \le \epsilon + \boldsymbol{\xi}_i \qquad \text{for } i = 1, \ldots, M \tag{3b}$$

$$\langle \boldsymbol{w}, \phi(\boldsymbol{t}_i)\rangle + \sum_{j=1}^{K}\boldsymbol{\beta}_j\psi_j(\boldsymbol{t}_i) - \boldsymbol{y}_i \le \epsilon + \boldsymbol{\xi}_i^* \qquad \text{for } i = 1, \ldots, M \tag{3c}$$

$$\boldsymbol{\xi} \ge \mathbf{0},\ \boldsymbol{\xi}^* \ge \mathbf{0}\ . \tag{3d}$$

where ϕ is a feature mapping function which defines a positive semidefinite kernel $\kappa(\boldsymbol{t}_i, \boldsymbol{t}_j) := \langle \phi(\boldsymbol{t}_i), \phi(\boldsymbol{t}_j)\rangle$, for all $i, j \in \{1, \ldots, M\}$, while $\{\psi_j\}_{j=1}^K$ are the basis functions for the parametric part of the model function. The model function is

defined as an extended linear model of parametric and nonparametric parts, that is, $h(\boldsymbol{t}) = \langle \boldsymbol{w}, \phi(\boldsymbol{t}) \rangle + \sum_{j=1}^{K} \boldsymbol{\beta}_j \psi_j(\boldsymbol{t})$. We typically have $K \ll M$. If $K = 1$ and ψ_1 is a constant function, we recover the standard SVM regression problem.

The Wolfe-dual of (3) has the form (2), where

$$
\begin{aligned}
\mathbf{x} &= \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\alpha}^* \end{bmatrix} \in \mathbb{R}^{2M} \text{ for the dual vectors } \boldsymbol{\alpha} \text{ and } \boldsymbol{\alpha}^* \text{ of (3b) and (3c), resp.,} \\
\boldsymbol{p} &= [\epsilon - \boldsymbol{y}_1, \ldots, \epsilon - \boldsymbol{y}_M, \epsilon + \boldsymbol{y}_1, \ldots, \epsilon + \boldsymbol{y}_M]^T \in \mathbb{R}^{2M} \ , \\
\boldsymbol{Q}_{ij} &= \begin{cases} \boldsymbol{y}_i \boldsymbol{y}_j \kappa(\boldsymbol{t}_i, \boldsymbol{t}_j) & \text{if } 1 \le i, j \le M, \text{ or } M+1 \le i, j \le 2M \\ -\boldsymbol{y}_i \boldsymbol{y}_j \kappa(\boldsymbol{t}_i, \boldsymbol{t}_j) & \text{otherwise} \end{cases} , \\
\boldsymbol{b} &= \mathbf{0} \ ,
\end{aligned}
$$

and

$$
\boldsymbol{A} = \begin{bmatrix} \psi_1(\boldsymbol{t}_1) & \cdots & \psi_1(\boldsymbol{t}_M) & -\psi_1(\boldsymbol{t}_1) & \cdots & -\psi_1(\boldsymbol{t}_M) \\ \psi_2(\boldsymbol{t}_1) & \cdots & \psi_2(\boldsymbol{t}_M) & -\psi_2(\boldsymbol{t}_1) & \cdots & -\psi_2(\boldsymbol{t}_M) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \psi_K(\boldsymbol{t}_1) & \cdots & \psi_K(\boldsymbol{t}_M) & -\psi_K(\boldsymbol{t}_1) & \cdots & -\psi_K(\boldsymbol{t}_M) \end{bmatrix} \in \mathbb{R}^{K \times 2M} \ .
$$

Introducing $\boldsymbol{\eta}$ as the Lagrange multipliers for the constraints $\boldsymbol{A}\mathbf{x} = \boldsymbol{b}$ in (2), the Karush-Kuhn-Tucker (KKT) optimality conditions for (2), stated here for later reference, are as follows:

$$
\begin{aligned}
\left(\boldsymbol{Q}\mathbf{x} + \boldsymbol{p} + \boldsymbol{A}^T\boldsymbol{\eta}\right)_i &\ge 0 && \text{if } \mathbf{x}_i = 0 && \text{(4a)} \\
\left(\boldsymbol{Q}\mathbf{x} + \boldsymbol{p} + \boldsymbol{A}^T\boldsymbol{\eta}\right)_i &\le 0 && \text{if } \mathbf{x}_i = C && \text{(4b)} \\
\left(\boldsymbol{Q}\mathbf{x} + \boldsymbol{p} + \boldsymbol{A}^T\boldsymbol{\eta}\right)_i &= 0 && \text{if } \mathbf{x}_i \in (0, C) && \text{(4c)} \\
\boldsymbol{A}\mathbf{x} &= \boldsymbol{b} && && \text{(4d)} \\
\mathbf{0} \le \mathbf{x} &\le C\mathbf{1} \ . && && \text{(4e)}
\end{aligned}
$$

If the kernel function κ is positive semidefinite, the Hessian matrix $\boldsymbol{Q}$ of (2) is also positive semidefinite, by definition. Therefore the objective function $F(\cdot)$ of (2) is convex, and as we only have linear constraints, the dual objective of (2) is a concave function in terms of the dual variable $\boldsymbol{\eta}$. Therefore the primal-dual pair $(\mathbf{x}, \boldsymbol{\eta})$ satisfying the conditions in (4) is the saddle point of (2). Moreover, $\boldsymbol{\eta}$ agrees with $\boldsymbol{\beta}$ in (3) since $\boldsymbol{\eta}$ is the double dual variable of $\boldsymbol{\beta}$ (refer [11] for details.) As our primal-dual solver discussed in Sect. 4 provides the optimal value of $\boldsymbol{\eta}$, there is no need to compute $\boldsymbol{\beta}$ separately.

3 Decomposition Framework

In this section we outline the decomposition strategy, giving details of two key aspects.

3.1 Subproblem Definition

The convex quadratic program (2) becomes harder to solve as the number of variables $n := 2M$ grows (where M is the number of data points), as the Hessian $\boldsymbol{Q}$ in (2) is dense and poorly conditioned for typical choices of the kernel function κ. The decomposition framework can alleviate these difficulties by working with a subset $\mathbf{x}_{\mathcal{B}}$, $\mathcal{B} \subset \{1, 2, \ldots, n\}$ of the variables at a time, fixing the other variables $\mathbf{x}_{\mathcal{N}}$, $\mathcal{N} = \{1, 2, \ldots, n\} \setminus \mathcal{B}$ at their current values. We usually choose the number of elements $n_{\mathcal{B}}$ in $\mathcal{B}$ to be much smaller than n. By partitioning the data objects $\boldsymbol{p}$, $\boldsymbol{A}$, and $\boldsymbol{Q}$ in the obvious way, we obtain the following subproblem at outer iteration k:

$$\begin{aligned} \min_{\mathbf{x}_{\mathcal{B}}} \quad & f(\mathbf{x}_{\mathcal{B}}) := \frac{1}{2}\mathbf{x}_{\mathcal{B}}^T \boldsymbol{Q}_{\mathcal{B}\mathcal{B}} \mathbf{x}_{\mathcal{B}} + (\boldsymbol{Q}_{\mathcal{B}\mathcal{N}} \mathbf{x}_{\mathcal{N}}^k + \boldsymbol{p}_{\mathcal{B}})^T \mathbf{x}_{\mathcal{B}} \\ \text{s.t.} \quad & \boldsymbol{A}_{\mathcal{B}} \mathbf{x}_{\mathcal{B}} = -\boldsymbol{A}_{\mathcal{N}} \mathbf{x}_{\mathcal{N}}^k + \boldsymbol{b}, \qquad 0 \le \mathbf{x}_{\mathcal{B}} \le C\mathbf{1}, \end{aligned} \tag{5}$$

where $\mathbf{x}_{\mathcal{N}}^k$ contains the current values of the $\mathcal{N}$ components. This problem has the same form as (2); we discuss solution methods in Sect. 4.

Since our emphasis in this paper is computational, we leave a convergence theory for this decomposition framework for future work. Suffice for the present to make a few remarks. If $\mathcal{B}$ is chosen so that the columns of $\boldsymbol{A}_{\mathcal{B}}$ corresponding to components of $\mathbf{x}_{\mathcal{B}}$ that are away from their bounds in (5) form a full-row-rank matrix, and if appropriate two-sided projections of $\boldsymbol{Q}_{\mathcal{B}\mathcal{B}}$ are positive definite, then (5) has a primal-dual solution $(\mathbf{x}_{\mathcal{B}}^*, \boldsymbol{\eta}^*)$ that corresponds to a solution $(\mathbf{x}^*, \boldsymbol{\eta}^*) = (\mathbf{x}_{\mathcal{B}}^*, \mathbf{x}_{\mathcal{N}}^*, \boldsymbol{\eta}^*)$ of (2), when $\mathbf{x}_{\mathcal{N}}^k = \mathbf{x}_{\mathcal{N}}^*$. Perturbation results can be used to derive a local convergence theory, and it may be possible to derive a global theory from appropriate generalizations of the results in [12].

3.2 Working Set Selection

The selection of working set $\mathcal{B}$ at each outer iteration is inspired by the approach of Joachims [4], later improved by Serafini and Zanni [6]. The size of the working set is fixed at some value $n_{\mathcal{B}}$, of which up to n_{c} are allowed to be "fresh" indices while the remainder are carried over from the current working set. Given the current primal-dual iterate $(\mathbf{x}^{k+1}, \boldsymbol{\eta}^{k+1})$, we find the indices corresponding to the nonzero components $\boldsymbol{d}_i$ obtained from the following problem:

$$\begin{aligned} \min_{\boldsymbol{d}} \quad & \left(\nabla F(\mathbf{x}^{k+1}) + (\boldsymbol{\eta}^{k+1})^T \boldsymbol{A}\right)^T \boldsymbol{d} \\ \text{s.t.} \quad & \begin{aligned} 0 \le \boldsymbol{d}_i \le 1 \quad & \text{if } \mathbf{x}_i^{k+1} = 0, \\ -1 \le \boldsymbol{d}_i \le 0 \quad & \text{if } \mathbf{x}_i^{k+1} = C, \\ -1 \le \boldsymbol{d}_i \le 1 \quad & \text{if } \mathbf{x}_i^{k+1} \in (0, C), \\ \#\{\boldsymbol{d}_i | \boldsymbol{d}_i \ne 0\} \le n_{\mathrm{c}}. & \end{aligned} \end{aligned} \tag{6}$$

Note that the objective function of (6) is a linearization of the Lagrangian function of F at the current primal-dual pair $(\mathbf{x}^{k+1}, \boldsymbol{\eta}^{k+1})$. Our approach is motivated by the KKT conditions (4), and indeed can be solved by simply sorting

the violations of these conditions. It contrasts with previous methods [4, 6, 12], in which the equality constraints are enforced explicitly in the working set selection subproblem. Our approach has no requirements on the size of n_c, yet it is still effective when $\boldsymbol{\eta}^{k+1}$ is close to the optimal value $\boldsymbol{\eta}^*$.

Earlier analysis of decomposition algorithms based on working set selection schemes has been performed by Lin [13], who shows linear convergence for the case of a single constraint, under positive definiteness assumptions on $\boldsymbol{Q}$. Tseng and Yun [12] proposed a decomposition framework for a formulation similar to (2) that includes multiple equality constraints. They present a convergence analysis which assumes that the subproblems at each step of decomposition are solved exactly, although they do not discuss techniques for solving the subproblem. Their working set selection algorithm requires relatively high complexity ($\mathcal{O}(K^3n^2)$) in general, compared with the $\mathcal{O}(n \log n)$ complexity of our approach.

The (up to) n_c new components from (6) are augmented to a total of $n_{\mathcal{B}}$ entries by adding indices from the previous working set $\mathcal{B}$ according to a certain priority. We choose the indices of the off-bounds components ($0 < \mathbf{x}_i^{k+1} < C$) first, and then those of lower and upper bounds. We reduce n_c as the change between two consecutive working sets decreases, as in [6]. We observe that adaptive reduction of n_c provides better convergence of the Lagrange multiplier $\boldsymbol{\eta}^k$, and helps avoid zigzagging between two working sets without making further progress. Adaptive reduction also helps not to degrade the benefit of optimizing many new components in a single decomposition step.

Our decomposition framework is summarized in Algorithm 1.

Algorithm 1 Decomposition Framework

1. **Initialization.** Choose an initial point $\mathbf{x}^1$ of (2) (possibly infeasible), initial guess of the Lagrange multiplier $\boldsymbol{\eta}^1$, positive integers $n_{\mathcal{B}} \geq K$ and $0 < n_c < n_{\mathcal{B}}$, and convergence tolerance `tolD`. Choose an initial working set $\mathcal{B}$ and set $k \leftarrow 1$.

2. **Subproblem.** Solve the subproblem (5) for the current working set $\mathcal{B}$, to obtain solution $\mathbf{x}_{\mathcal{B}}^{k+1}$ together with Lagrange multiplier $\boldsymbol{\eta}^{k+1}$ of the equality constraints. Set $\mathbf{x}^{k+1} = (\mathbf{x}_{\mathcal{B}}^{k+1}, \mathbf{x}_{\mathcal{N}}^{k})$.

3. **Gradient Update.** Evaluate the gradient of the Lagrangian of (2), by incrementally updating ∇F, as indicated here:

$$\nabla F(\mathbf{x}^{k+1}) + (\boldsymbol{\eta}^{k+1})^T \boldsymbol{A} = \nabla F(\mathbf{x}^k) + \begin{bmatrix} \boldsymbol{Q}_{\mathcal{B}\mathcal{B}} \\ \boldsymbol{Q}_{\mathcal{N}\mathcal{B}} \end{bmatrix} (\mathbf{x}_{\mathcal{B}}^{k+1} - \mathbf{x}_{\mathcal{B}}^{k}) + (\boldsymbol{\eta}^{k+1})^T \boldsymbol{A} \ .$$

4. **Convergence Check.** If the maximal violation of the KKT conditions (4) falls below `tolD`, terminate with the primal-dual solution $(\mathbf{x}^{k+1}, \boldsymbol{\eta}^{k+1})$.

5. **Working Set Update.** Find a new working set $\mathcal{B}$ as described in Sect. 3.2.
6. Set $k \leftarrow k + 1$ and go to step 2.

4 Subproblem Solver

Recalling that the decomposition framework requires both a primal solution $\mathbf{x}_{\mathcal{B}}$ and Lagrange multipliers $\boldsymbol{\eta}$ to be obtained for the subproblem (5), we consider the following min-max formulation of (5):

$$\max_{\boldsymbol{\eta}} \min_{\mathbf{x}_{\mathcal{B}} \in \Omega} \quad L(\mathbf{x}_{\mathcal{B}}, \boldsymbol{\eta}) \ , \tag{7}$$

where $\Omega = \{\mathbf{x} \in \mathbb{R}^{n_{\mathcal{B}}} | \mathbf{0} \leq \mathbf{x} \leq C\mathbf{1}\}$ and

$$L(\mathbf{x}_{\mathcal{B}}, \boldsymbol{\eta}) := f(\mathbf{x}_{\mathcal{B}}) + \boldsymbol{\eta}^T(\boldsymbol{A}_{\mathcal{B}}\mathbf{x}_{\mathcal{B}} + \boldsymbol{A}_{\mathcal{N}}\mathbf{x}^k_{\mathcal{N}}) \ .$$

In this section we describe a primal-dual approach for solving (7), in which steps are taken in $\mathbf{x}_{\mathcal{B}}$ and $\boldsymbol{\eta}$ in an alternating fashion. Scalings that include second-order information are applied to both primal and dual steps. We call the approach PDSG (for "Primal-Dual Scaled Gradient").

Our approach can be viewed as an extreme variant of the method of multipliers [14], in which we do not attempt to minimize the augmented Lagrangian between updates of the Lagrange multiplier estimates, but rather take a single step along a partial, scaled, and projected gradient direction in the primal space. In describing the general form of each iteration, we use superscripts ℓ to denote iteration counts, bearing in mind that they refer to the *inner* iterations of the decomposition framework (and hence are distinct from the superscripts k of the previous section, which denote outer iterations).

$$\mathbf{x}^{\ell+1}_{\mathcal{B}} \leftarrow \mathbf{x}^{\ell}_{\mathcal{B}} + s(\mathbf{x}^{\ell}_{\mathcal{B}}, \boldsymbol{\eta}^{\ell}) \tag{8a}$$

$$\boldsymbol{\eta}^{\ell+1} \leftarrow \boldsymbol{\eta}^{\ell} + t(\mathbf{x}^{\ell+1}_{\mathcal{B}}, \boldsymbol{\eta}^{\ell}) \ , \tag{8b}$$

where $s(\cdot,\cdot)$ and $t(\cdot,\cdot)$ are steps, defined below. In computational testing, we found PDSG to be superior to methods more like traditional method-of-multiplier approaches, which would take multiple steps in $\mathbf{x}_{\mathcal{B}}$ in between successive steps in $\boldsymbol{\eta}$.

Primal Step. In the ℓ-th iteration of the subproblem solver, we choose a small sub-working set $\mathcal{W}^{\ell} \subset \mathcal{B}$ containing at most $n_{\mathcal{W}}$ elements (where $n_{\mathcal{W}}$ is a user-defined parameter), containing those indices in $\mathcal{B}$ that are among the $n_{\mathcal{W}}$ most-violated KKT conditions (4a)-(4c) for the subproblem (5). We define the further subset $\bar{\mathcal{W}}^{\ell}$ by selecting those indices $i \in \mathcal{W}^{\ell}$ that are *not* at one of their bounds 0 and C. We then construct the block-diagonal $n_{\mathcal{B}} \times n_{\mathcal{B}}$ matrix $\boldsymbol{H}^{\ell}$, as follows:

$$\boldsymbol{H}^{\ell}_{ij} = \begin{cases} \boldsymbol{Q}_{ij} + \tau\delta_{ij} & \text{if } i \in \bar{\mathcal{W}}^{\ell} \text{ and } j \in \bar{\mathcal{W}}^{\ell} \\ \boldsymbol{Q}_{ii} & \text{if } i = j \text{ and } i \in \mathcal{W}^{\ell} \setminus \bar{\mathcal{W}}^{\ell} \\ \infty & \text{if } i = j \text{ and } i \notin \mathcal{W}^{\ell} \\ 0 & \text{otherwise,} \end{cases} \tag{9}$$

where $\delta_{ij} = 1$ if $i = j$ and 0 otherwise, while τ is a small positive parameter (we use $\tau = 10^{-8}$) chosen to ensure that the "block" part of $\boldsymbol{H}^{\ell}$ is numerically non-singular. Since we apply the inverse of this matrix to the gradient in computing

the step, the components of the matrix-vector product that correspond to the ∞ entries will evaluate to zero. Specifically, we obtain the search direction as follows:

$$\boldsymbol{d}^{\ell} := \mathbf{x}_{\mathcal{B}}^{\ell} - \mathbb{P}_{\Omega}\left(\mathbf{x}_{\mathcal{B}}^{\ell} - \left(\boldsymbol{H}^{\ell}\right)^{-1} \nabla_{\mathbf{x}_{\mathcal{B}}} L(\mathbf{x}_{\mathcal{B}}^{\ell}, \boldsymbol{\eta}^{\ell})\right) \tag{10}$$

where $\mathbb{P}_{\Omega}(\cdot)$ is a projection operator to the set Ω, which is trivial to compute since this set is defined by simple bounds. This is essentially the two-metric gradient projection search direction [15] applied to the subvector defined by $\mathcal{W}^{\ell}$. Given this direction, the primal step s from (8a) is defined to be

$$s(\mathbf{x}_{\mathcal{B}}^{\ell}, \boldsymbol{\eta}^{\ell}) = \alpha_{\ell} \boldsymbol{d}^{\ell} \ , \tag{11}$$

where $\alpha_{\ell} \in \mathbb{R}$ is the unconstrained minimizer of $L(\cdot, \boldsymbol{\eta}^{\ell})$ along the line segment connecting $\mathbf{x}_{\mathcal{B}}^{\ell}$ to $\mathbf{x}_{\mathcal{B}}^{\ell} + \boldsymbol{d}^{\ell}$.

Dual Update. The step in the dual variable $\boldsymbol{\eta}$ is a Newton-like step in the dual objective function for (5), which is

$$g(\boldsymbol{\eta}) := \min_{\mathbf{x}_{\mathcal{B}} \in \Omega} L(\mathbf{x}_{\mathcal{B}}, \boldsymbol{\eta}).$$

This is a piecewise quadratic concave function. Since its second derivative does not exist, we cannot take a true Newton step. However, we use a slight modification of the procedure in Kienzle and Schölkopf [9] to form a diagonal approximation $\boldsymbol{G}$ to this matrix. Their procedure progressively updates $\boldsymbol{G}$ by applying one step of Gauss-Jacobi-like procedure at each iteration of the MPD optimization scheme. Unlike MPD, our modification estimates $\boldsymbol{G}$ both internally and externally to the optimization loop. The external estimation ensures us to have an approximation with a certain quality before performing any dual updates. We refer the reader to [9] for additional details. The dual step t in (8b) is thus simply

$$t(\mathbf{x}_{\mathcal{B}}^{\ell+1}, \boldsymbol{\eta}^{\ell}) = -\boldsymbol{G}^{-1} \nabla_{\boldsymbol{\eta}} L(\mathbf{x}_{\mathcal{B}}^{\ell+1}, \boldsymbol{\eta}^{\ell}). \tag{12}$$

Our subproblem algorithm is summarized in Algorithm 2.

Algorithm 2 Subproblem solver: PDSG

1. **Initialization.** Given a index set $\mathcal{B}$, choose initial points $\mathbf{x}_{\mathcal{B}}^{1}$ and $\boldsymbol{\eta}^{1}$. Choose $n_{\mathcal{W}}$ such that $1 \leq n_{\mathcal{W}} \leq n_{\mathcal{B}}$. Choose small positive convergence tolerance `tolS`. Set $\ell \leftarrow 1$.

2. **Sub-Working Set Selection.** Construct $\mathcal{W}^{\ell}$ (with at most $n_{\mathcal{W}}$ elements) and $\bar{\mathcal{W}}^{\ell}$ as described above.

3. **Primal-Dual Update.** Take the primal step according to (8a) and (11), then the dual step according to (8b) and (12).

4. **Convergence Check.** If the maximal KKT violation of the current primal-dual pair $(\mathbf{x}_{\mathcal{B}}^{\ell+1}, \boldsymbol{\eta}^{\ell+1})$ is less than `tolS`, exit. Otherwise, go to step 2.

5 Experiments

We report on computational experiments that show the intrinsic benefits of the PDSG approach, as well as the benefits of the decomposition strategy, when applied to a simple semiparametric SVM regression problem. We compare PDSG with the MPD algorithm of Kienzle and Schölkopf [9], which has slightly better performance and lower memory requirement than the interior-point-based approach used in [1]. We also show the advantage of semiparametric modeling on a real world problem.

Implementations. We implemented both the decomposition framework (Algorithm 1) and the PDSG subproblem solver (Algorithm 2) in `C++`. The code was developed by modifying the GPDT code of Serafini, Zanghirati, and Zanni [5][1], and retains many features of this code. Our code caches once-computed kernel entries for reuse, with the least-recently-used (LRU) replacement strategy. For efficiency, our subproblem solver exploits warm starting; the most recent values of the primal and dual variables are used as the starting points in the next invocation of the subproblem solver. We also implemented the MPD solver [9] in `C++`, again basing the implementation on GPDT. Our codes can be invoked either with the decomposition framework, or in "stand-alone" mode, in which the solver is applied directly to the stated problem.

5.1 Toy Problem

For the semiparametric regression test problem, we choose the modified Mexican hat function studied in [1, 9]:

$$\omega(t) = \sin(t) + \operatorname{sinc}\left(2\pi(t-5)\right) \quad .$$

To generate data, we sample the function ω at uniform random points $t_i \in \mathbb{R}$ in the interval $[0, 10]$, making M samples in total. The observations y_i's are corrupted with additive Gaussian noise ζ_i with mean 0 and standard deviation 0.2, that is, $y_i = \omega(t_i) + \zeta_i$. In the training process, we use Gaussian kernel $\kappa(x, y) = \exp(-\gamma||x - y||^2)$ with $\gamma = 0.25$, and set the insensitivity width ϵ of the loss function to $\epsilon = 0.05$, as in [1]. The optimal tradeoff parameter value of $C = 0.5$ is found by 10-fold cross validation (CV) in [1] using very small samples ($M = 50$). Since we are interested in the convergence behavior of algorithms with larger samples, we performed computational experiments with $C = 0.1$, $C = 1$, and $C = 10$. Our model is $h(t) = \langle \boldsymbol{w}, \phi(t)\rangle + \sum_{j=1}^{K} \boldsymbol{\beta}_j \psi_j(t)$, with two basis functions $\psi_1(t) = \sin(t)$ and $\psi_2(t) = \operatorname{sinc}\left(2\pi(t-5)\right)$ as in [9].

The size of the sample dataset M is varied from 500 to 100000. The subproblem size $n_{\mathcal{B}}$ and the maximum number of new components in each subproblem n_{c} are fixed to 500 and 100, respectively, as these values gave good performance on the largest data set. Similarly, we fix the sub-working set size $n_{\mathcal{W}}$ to 2. (We tried

[1] GPDT is available at `http://mloss.org/software/view/54/`

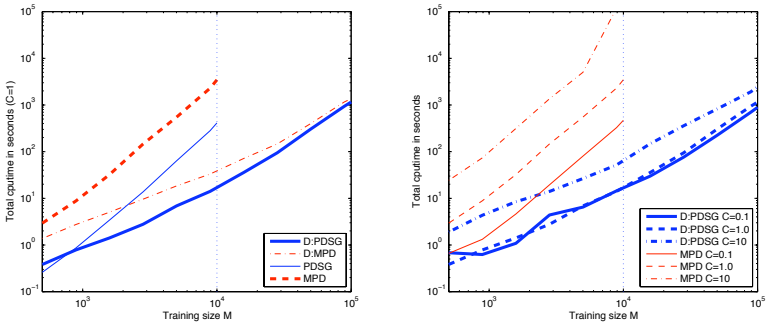

Fig. 1. Left plot shows total runtimes using solvers PDSG and MPD in stand-alone mode and inside of the decomposition framework (D:PDSG and D:MPD) with $C = 1$. Right plot shows the total runtimes of D:PDSG (our proposed method) and MPD with different C values. For larger number of training examples M, updating of the full gradient in Step 3 of Algorithm 1 dominates the computation, blurring the distinction between PDSG and MPD as subproblem solvers (left plot). D:PDSG outperforms MPD for all C values tried (right plot). Stand-alone algorithms are run only for training-set size up to 10000 because of their high computational cost.

various other values between 1 and 25, but 2 was slightly better than several alternatives.) In each setting, we use a kernel cache of 400MB in size.

Growth of the total runtime of the algorithms with increasing size of the data set is shown in Fig. 1. When the decomposition framework is used, the stopping threshold values are set to $\texttt{tolD} = 0.001$ and $\texttt{tolS} = 0.0005$. In stand-alone settings, we set $\texttt{tolS} = 0.001$. We impose a slightly tighter threshold on subproblem solvers inside the decomposition framework to reduce the number of decomposition steps. Outer iterations in the decomposition framework become more costly as the number of variables increases, mainly because the full gradient update in Step 3 of Algorithm 1 becomes more expensive. The benefit of using decomposition framework becomes larger as the dataset size grows. For instance, D:PDSG is about 100 times faster than MPD when $M = 10000$. In decomposition settings, using PDSG as the inner solver found the solution two to three times faster than using MPD as the inner solver on average. Our proposed method D:PDSG shows quite stable scaling behavior for different values of C.

Convergence and Complexity. The different convergence behavior of PDSG and MPD is illustrated in Fig. 2. Here both solvers are asked to solve a semiparametric regression problem discussed above with 1000 samples, in stand-alone mode. In the top and middle plots, the dual and primal infeasibility, respectively, are more rapidly reduced with PDSG than with MPD. (Note that since we project the iterates $\mathbf{x}^k$ to the bound constraints set, the KKT condition (4e) is always satisfied.) The bottom plot of Fig. 2 shows the changes of the first Lagrange multiplier (the coefficient of the first basis function). In that, MPD is showing the typical behavior of the method of multipliers: sudden changes are made,

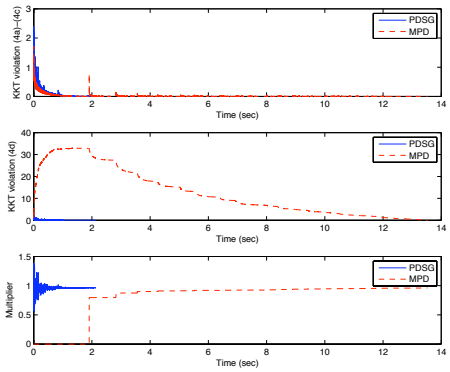

Fig. 2. Convergence of PDSG and MPD in stand-alone mode (Mexican hat, dataset size M=1000). PDSG requires about 2 seconds to reach convergence, whereas MPD takes about 14 seconds. (Top) maximum violation of the dual feasibility conditions (4a), (4b), (4c). (Middle) maximum violation of the primal equality constraints (4d). (Bottom) convergence of the first Lagrange multiplier to its optimal value of 1. The horizontal axis represents elapsed CPU time.

but time gaps between such changes are rather large. In contrast, PDSG keeps making changes to the multiplier, resulting in a faster approach to the optimal value.

When the sub-working-set size $n_{\mathcal{W}}$ is smaller than the working-set size $n_{\mathcal{B}}$ of the subproblem (5), PDSG has computational complexity $\mathcal{O}(Kn_{\mathcal{B}})$, the same as MPD, where K is the number of equality constraints in (2). Dual updates in Algorithm 2 requires $\mathcal{O}(Kn_{\mathcal{B}})$ operations; all primal updates are done in $\mathcal{O}(n_{\mathcal{B}})$. The effect of increasing K on the total time taken by D:PDSG is shown in Fig. 3. We use the basis functions

$$\psi_j(t) = \begin{cases} \cos(j\pi t) & j = 0, 2, 4, \ldots \\ \sin(j\pi t) & j = 1, 3, 5, \ldots \end{cases}$$

and datasets of size $M = 1000$ randomly sampled from the Mexican hat function. Other settings are the same as the previous experiment. As expected, we observe linear scaling of total runtime with K.

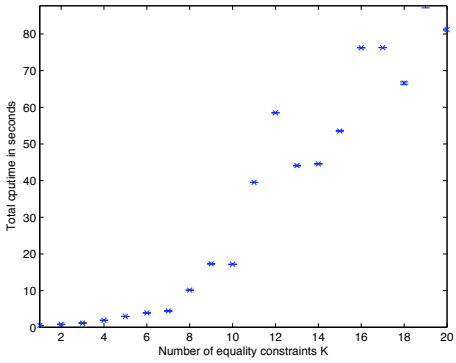

Fig. 3. Total solution time for D:PDSG with increasing number of equality constraints K. Measurements are averaged over 10 repetitions with different random datasets (M=1000) sampled from the Mexican hat function, and error bars (hardly visible) show the standard deviations. The time complexity of D:PDSG is $\mathcal{O}(uKn_{\mathcal{B}})$ where u is the number of outer iterations. Solver time appears to increase linearly with K.

5.2 Milan Respiratory Illness Dataset

We consider a dataset[2] from the study on the effect of air pollution on respiratory illness in Milan, Italy, during 1980–89 [16]. This dataset consists of daily records of environmental conditions and the number of deaths due to respiratory diseases (total 3652 records, 9 features). All features are scaled linearly to the range $[0, 1]$. We construct a test set by holding out 20% of randomly chosen records from the dataset, using the remaining records for training.

We hypothesize a simple semiparametric model to predict the number of respiratory deaths, inspired by [16]:

$$h_{\mathrm{sp}}(\boldsymbol{t}) = \langle \boldsymbol{w}, \phi(\boldsymbol{t}) \rangle + \boldsymbol{\beta}_1(\boldsymbol{t}_{\mathrm{temp}}) + \boldsymbol{\beta}_2(\boldsymbol{t}_{\mathrm{SO}_2}) + \boldsymbol{\beta}_3(\boldsymbol{t}_{\mathrm{temp}})^2 + \boldsymbol{\beta}_4(\boldsymbol{t}_{\mathrm{SO}_2})^2 + \boldsymbol{\beta}_5 \ ,$$

where the features $\boldsymbol{t}_{\mathrm{temp}}$ and $\boldsymbol{t}_{\mathrm{SO}_2}$ correspond to mean temperature and SO_2 level of the day, respectively. Our purpose is to study how those two elements affect the respiratory illness.

We fit our semiparametric model to the training data, and compare its prediction performance on the test set to that of a nonparametric model

$$h_{\mathrm{np}}(\boldsymbol{t}) = \langle \boldsymbol{w}, \phi(\boldsymbol{t}) \rangle + \boldsymbol{\beta}_1 \ .$$

With Gaussian kernel ($\gamma = 25.0$) and ϵ-insensitive loss function ($\epsilon = 0.01$), we perform 10-fold CV on the training set to determine the best balancing parameter C for each of semiparametric and nonparametric models independently.

The results are shown in Table 1. The semiparametric model attained smaller prediction error on the test set than the nonparametric model, indicating that the embedding of prior knowledge in h_{sp} while retaining the power of nonparametric approaches is beneficial. Moreover, the parametric components in the trained semiparametric model

$$h_{\mathrm{sp}}(\boldsymbol{t}) = \langle \boldsymbol{w}^*, \phi(\boldsymbol{t}) \rangle - 0.30(\boldsymbol{t}_{\mathrm{temp}}) + 0.26(\boldsymbol{t}_{\mathrm{SO}_2}) + 0.22(\boldsymbol{t}_{\mathrm{temp}})^2 - 0.07(\boldsymbol{t}_{\mathrm{SO}_2})^2 + 0.22 \ .$$

reveal that (i) deaths are lower in the middle of the temperature range, and (ii) there is an almost linear increase of death rate with SO_2 level. These results broadly agree with the outcomes of [16], which were acquired from completely different statistical analysis techniques. It is difficult to perform model interpretation of this type with nonparametric approaches.

6 Conclusions

We have presented a new method for semiparametric SVM regression problems, which extends a number of previous approaches in being able to handle multiple equality constraints. Our method combines a decomposition framework with a primal-dual scaled gradient solver for the subproblems. Computational tests indicate that the approach improves on previously proposed methods.

[2] Available at `http://www.uow.edu.au/~mwand/webspr/data.html`

Table 1. Nonparametric and semiparametric regression on Milan dataset. The loss penalty parameter C is determined by cross validation. Comparing the prediction performance on the test set by mean square error (MSE) values, the semiparametric model performed better than the nonparametric model by 2.8%. No significant difference of the number of support vectors (SVs) was found between the two methods.

Model	C	Fraction of SVs	Training Time (s)	Test Error (MSE)
Nonparametric (h_{np})	0.025	46.7%	1.17	0.019368
Semiparametric (h_{sp})	0.01	46.9%	5.35	0.018828

Future work includes reducing the cost of the full gradient update by using a randomized sampling procedure for the components of the gradient, as has been tried in a different context in [17]. While the concept is simple, it is not straightforward to implement this technique in conjunction with caching of kernel entries, which is so important to efficient implementation of SVM solvers based on QP formulations. Other research topics include devising a more effective update strategy for the dual variables $\boldsymbol{\eta}$ in the subproblem solver, and theoretical analyses both of the decomposition framework (including the working set selection technique) and the subproblem solver.

Acknowledgements

The authors acknowledge the support of NSF Grants CCF-0430504, DMS-0427689, CNS-0540147, and DMS-0914524. The first author was supported in part by Samsung Scholarship from the Samsung Foundation of Culture.

References

1. Smola, A.J., Frieß, T.T., Schölkopf, B.: Semiparametric support vector and linear programming machines. In: Advances in Neural Information Processing Systems 11, Cambridge, MA, USA, MIT Press (1999) 585–591
2. Platt, J.C.: Fast training of support vector machines using sequential minimal optimization. In Schölkopf, B., Burges, C., Smola, A., eds.: Advances in Kernel Methods - Support Vector Learning. MIT Press, Cambridge, MA (1999) 185–208
3. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines. (April 2009) version 2.89, `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.
4. Joachims, T.: Making large-scale support vector machine learning practical. In Schölkopf, B., Burges, C., Smola, A., eds.: Advances in Kernel Methods - Support Vector Learning. MIT Press, Cambridge, MA (1999) 169–184
5. Serafini, T., Zanghirati, G., Zanni, L.: Gradient projection methods for large quadratic programs and applications in training support vector machines. Optimization Methods and Software **20**(2–3) (2004) 353–378
6. Serafini, T., Zanni, L.: On the working set selection in gradient projection-based decomposition techniques for support vector machines. Optimization Methods and Software **20** (2005) 583–596

7. Keerthi, S.S., Gilbert, E.G.: Convergence of a generalized smo algorithm for svm classifier design. Machine Learning **46**(1-3) (2002) 351–360
8. Dai, Y.H., Fletcher, R.: New algorithms for singly linearly constrained quadratic programs subject to lower and upper bounds. Mathematical Programming, Series A **106** (2006) 403–421
9. Kienzle, W., Schölkopf, B.: Training support vector machines with multiple equality constraints. In: Machine Learning: ECML 2005. Volume 16. (October 2005)
10. Boser, B.E., Guyon, I.M., Vapnik, V.N.: A training algorithm for optimal margin classifiers. In: COLT '92: Proceedings of the fifth annual workshop on Computational learning theory, New York, NY, USA, ACM (1992) 144–152
11. Scholkopf, B., Smola, A.J.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge, MA, USA (2001)
12. Tseng, P., Yun, S.: A coordinate gradient descent method for linearly constrained smooth optimization and support vector machines training. Published online in Computational Optimization and Applications (October 2008)
13. Lin, C.J.: Linear convergence of a decomposition method for support vector machines. Technical report, Department of Computer Science and Information Engineering, National Taiwan University (2001)
14. Bertsekas, D.P.: Nonlinear Programming. Second edn. Athena Scientific (1999)
15. Gafni, E.M., Bertsekas, D.P.: Two-metric projection methods for constrained optimization. SIAM Journal on Control and Optimization **22** (1984) 936–964
16. Vigotti, M.A., Rossi, G., Bisanti, L., Zanobetti, A., Schwartz, J.: Short term effects of urban air pollution on respiratory health in Milan, Italy, 1980-89. Journal of Epidemiology Community Health **50** (1996) s71–75
17. Shi, W., Wahba, G., Wright, S.J., Lee, K., Klein, R., Klein, B.: LASSO-Patternsearch algorithm with application to opthalmology data. Statistics and its Interface **1** (January 2008) 137–153