Training Support Vector Machines with Multiple Equality Constraints

Wolf Kienzle and Bernhard Schölkopf

Max-Planck-Institute for Biological Cybernetics,
Empirical Inference Department, Spemannstr. 38,
72076 Tübingen, Germany
{kienzle, bs}@tuebingen.mpg.de

Abstract. In this paper we present a primal-dual decomposition algorithm for support vector machine training. As with existing methods that use very small working sets (such as *Sequential Minimal Optimization* (SMO), *Successive Over-Relaxation* (SOR) or the *Kernel Adatron* (KA)), our method scales well, is straightforward to implement, and does not require an external QP solver. Unlike SMO, SOR and KA, the method is applicable to a large number of SVM formulations regardless of the number of equality constraints involved. The effectiveness of our algorithm is demonstrated on a more difficult SVM variant in this respect, namely semi-parametric support vector regression.

1 Introduction

Support Vector Machines (SVM) rank among the most widely used techniques in machine learning today. Besides their good generalization ability, a major benefit is that the training phase in SVMs is a convex optimization problem and hence, unlike that of many competing methods does not suffer from local minima. However, the nature of this training problem is such that a naïve implementation will require $\mathcal{O}(m^2)$ memory, where m is the number of training points.

This dilemma has motivated the development of algorithms with more efficient memory usage. Most approaches that have been studied in this respect exploit the fact that the SVM problem can be solved incrementally, i.e. it can be decomposed into a series of smaller problems that can be solved independently. In fact, such *decomposition methods* [10] not only solve larger problems for a given amount of memory, but have turned out to to give significant speed improvements. As a result, essentially all of the popular SVM solvers today are based on this idea, e.g. *SVMlight* [8], *LIBSVM* [3], *SVMTorch* [5] or *HeroSVM* [6]. These implementations differ in mainly two respects, namely in the size of the subproblems and in how the corresponding training subsets (i.e. the working sets) are chosen. A common choice for the working set size is *two*, in which case the subproblems can be solved analytically. Also, choosing a good working set is simpler when fewer points have to be selected. This was first proposed in [11] as *Sequential Minimal Optimization* (SMO), which forms the basis for three of the four packages listed above (the fourth one includes SMO as a special case).

It seems natural to ask whether *one* point working sets can further improve on the results of SMO. In fact, this has been studied in various forms, e.g. under the name of *Kernel Adatron* (KA) [7] or *Successive Over-Relaxation* (SOR) [9], and indeed, all these methods yielded performances that were comparable, if not superior to SMO. However, this comes at the price of solving a modified problem, since equality constraints (there is one in C-SVMs and two in ν-SVMs) cannot be treated using single point updates. To remedy this, the authors changed the SVM formulation such that the equality constraints vanished. A similar problem was faced by Chang and Lin [2] during the implementation of a ν-SVM solver for *LIBSVM*: the second equality constraint in ν-SVMs is cumbersome for an SMO based optimizer, since the latter can only deal with one such constraint naturally. The bottom line is that the number of equality constraints, which varies among different SVM formulations, determines if we can use the efficient two (or even one) point variant of the decomposition method.

In this paper we propose an algorithm that naturally handles any (small and constant) number of equality constraints, and at the same time allows for arbitrary working sets sizes, in particular, down to a single point. This is achieved through a primal-dual scheme that does not require feasibility in terms of the equality constraints, except at the solution. We demonstrate the benefits of our method by means of a generalized SVM formulation known as semi-parametric SVMs [13]. Here, the number of equality constraints corresponds to the number of

Table 1. Parameter settings for (1), for various support vector problems with no more than $k = 2$ equality constraints. Top two rows: C and ν soft margin SV classification, respectively. 3rd row: one-class SVM. Bottom two rows: ε and ν SV regression. Here, m is the number of data points $\mathbf{x}_i \in \mathbf{X}$. Note that in (1), the domain of $\boldsymbol{\alpha}$ is $\mathbb{R}^n$, and that $n = m$ in all cases except SVR, where we have $n = 2m$. $\mathbf{y} = (y_1, \ldots, y_m)^\top$ contains the target values ($y_i \in \{-1, 1\}$ for classification, $y_i = 1$ for single class problems, $y_i \in \mathbb{R}$ for regression). Furthermore, $\mathbf{0} = (0, \ldots, 0)^\top \in \mathbb{R}^m$, $\mathbf{1} = (1, \ldots, 1)^\top \in \mathbb{R}^m$, and $e_1 = 0$ in all cases. Finally, $\mathbf{K} \in \mathbb{R}^{m \times m}$, where $\mathbf{K}_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$ for regression and $\mathbf{K}_{ij} = y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$ otherwise, and $K(\cdot, \cdot) : \mathbf{X} \times \mathbf{X} \to \mathbb{R}$ denotes the kernel function.

	$\mathbf{H}$	d	$\mathbf{f}_0$	$\mathbf{f}_1$	e_1	$\mathbf{f}_2$	e_2
C-SVC	$\mathbf{K}$	$\frac{C}{m}$	$-\mathbf{1}$	$\mathbf{y}$	0	-	-
ν-SVC	$\mathbf{K}$	$\frac{1}{\nu m}$	$\mathbf{0}$	$\mathbf{y}$	0	$\mathbf{1}$	1
1-SVM	$\mathbf{K}$	$\frac{1}{\nu m}$	$\mathbf{0}$	$\mathbf{1}$	1	-	-
ε-SVR	$\begin{bmatrix} \mathbf{K} & -\mathbf{K} \\ -\mathbf{K} & \mathbf{K} \end{bmatrix}$	$\frac{C}{m}$	$\begin{bmatrix} -\mathbf{y} \\ \mathbf{y} \end{bmatrix} + \varepsilon$	$\begin{bmatrix} \mathbf{1} \\ -\mathbf{1} \end{bmatrix}$	0	-	-
ν-SVR	$\begin{bmatrix} \mathbf{K} & -\mathbf{K} \\ -\mathbf{K} & \mathbf{K} \end{bmatrix}$	$\frac{C}{m}$	$\begin{bmatrix} -\mathbf{y} \\ \mathbf{y} \end{bmatrix}$	$\begin{bmatrix} \mathbf{1} \\ -\mathbf{1} \end{bmatrix}$	0	$\frac{1}{C\nu} \begin{bmatrix} \mathbf{1} \\ \mathbf{1} \end{bmatrix}$	1

free parameters and is therefore arbitrary. As a result, nontrivial semi-parametric SVMs cannot be trained with SMO, but are well suited for our algorithm.

2 Problem Formulation

In the following, we consider the class of optimization problems

$$\begin{array}{lll} \min \frac{1}{2}\boldsymbol{\alpha}^\top \mathbf{H}\boldsymbol{\alpha} + \mathbf{f}_0^\top \boldsymbol{\alpha} & & \\ \text{s.t. } \; 0 \le \alpha_i \le d, & i = 1 \dots n & \\ \qquad \mathbf{f}_i^\top \boldsymbol{\alpha} = e_i & i = 1 \dots k & \end{array} \tag{1}$$

w.r.t. $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_n)^\top \in \mathbb{R}^n$, i.e. SVM problems with k equality constraints. The particular type of SVM that is implemented depends on the following parameters: the matrix $\mathbf{H} \succeq \mathbf{0} \in \mathbb{R}^{n \times n}$, symmetric and positive definite, $k+1$ column vectors $\mathbf{f}_0, \dots, \mathbf{f}_k \in \mathbb{R}^n$ and $k+1$ scalars $d \in \mathbb{R},\ d > 0$ and $e_i \in \mathbb{R},\ i = 1 \dots k$. Table 1 shows the parameter settings for (1) for several standard SVM problems [12]. Any of the SVMs in Table 1 can be made semi-parametric [13], which introduces one additional equality constraint per parameter (this will be discussed in more detail in Section 6). Please note that we will refer to (1) as the primal problem, for reasons that will become clear below.

3 Optimality Conditions

If the kernel $K(\cdot,\cdot)$ is positive definite, then so is the matrix $\mathbf{H}$ (see Table 1). This renders the primal objective convex. Since in addition, all constraints in (1) are affine, strong duality holds and the Karush-Kuhn-Tucker (KKT) conditions

$$\begin{array}{lll} \forall\, i = 1 \dots k & \text{(i)} & \mathbf{f}_i^\top \boldsymbol{\alpha} = e_i \\ & & \\ & \text{(ii)} & 0 \le \alpha_i \le d \\ \forall\, i = 1 \dots n & \text{(iii)} & \alpha_i = 0 \qquad \Rightarrow s_i \ge 0 \\ & \text{(iv)} & 0 < \alpha_i < d \Rightarrow s_i = 0 \\ & \text{(v)} & \alpha_i = d \qquad \Rightarrow s_i \le 0 \end{array} \tag{2}$$

where $\mathbf{s} = \mathbf{H}\boldsymbol{\alpha} + \mathbf{f}_0 + \mathbf{F}\boldsymbol{\eta}$, $\mathbf{F} = [\mathbf{f}_1, \dots, \mathbf{f}_k]$, and $\boldsymbol{\eta} \in \mathbb{R}^k$. are sufficient for optimality, i.e. any tuple $(\boldsymbol{\eta}, \boldsymbol{\alpha})$ that satisfies (2) is primal and dual optimal and has zero duality gap. Note that the dual variable(s) $\boldsymbol{\eta} = (\eta_1, \dots, \eta_k)^\top$ correspond to the k equality constraints in the primal problem (1). The n dual variables associated with the box constraints have been eliminated via a fusion with the complementary slackness conditions, resulting in the three implications (iii–v).

4 Decomposition Methods

The need for customized SVM solvers arises from the fact that briefly, learning works better the more training data is used; but then, as mentioned earlier, storing $\mathbf{H}$, which holds the information about the training data, requires a quadratically increasing amount of memory. The use of generic QP solvers is often not possible, especially when they require $\mathbf{H}$ to reside in memory throughout the optimization process. To remedy this, various so-called decomposition methods have been proposed (originally by Osuna et al. [10]). In a decomposition method, at each step, all α_i except a working set of size q are fixed. Then, after solving the respective subproblem (w.r.t. the q variable α_i), a new working set is selected, and the new subproblem is solved. This procedure is repeated until the global optimality criterion (i.e. regarding all n α_i) is met. Note that this requires only q columns of $\mathbf{H}$ to reside in memory at a time, so we can solve very large problems by using small working sets, i.e. small q. The following two subsections give a short review of two extreme versions of the decomposition method, namely the cases $q = 2$ and $q = 1$.

4.1 SMO (q=2)

The *Sequential Minimal Optimization* (SMO) method, introduced in [11], was originally proposed for C-SVC. As opposed to decomposition methods that use larger working sets $q > 2$, SMO has the striking property that its subproblems can be solved analytically, which makes the algorithm very simple to implement. A slightly modified, very efficient implementation of this algorithm is at the core of the widely used *LIBSVM* package [3]. Further implementations can be found in *SVMTorch* [5] and *HeroSVM* [6].

In SMO, subproblems are of size $q = 2$. For C-SVC, where we have only one equality constraint $\mathbf{y}^\top \boldsymbol{\alpha} = 0$ (see Table 1), the algorithm is as follows: first, we initialize $\boldsymbol{\alpha} = \mathbf{0}$, which is a feasible point for (1), since we have $0 \leq \alpha_i \leq \frac{C}{m}$ and $\mathbf{y}^\top \boldsymbol{\alpha} = \mathbf{0}$. At each iteration, a working set $\{\alpha_i, \alpha_j\}$ is selected and the quadratic objective function is minimized along the line $\mathbf{y}^\top \boldsymbol{\alpha} = 0$. Then, the new values for α_i and α_j are clipped to the interval $\left[0; \frac{C}{m}\right]$ in order to meet the inequality constraints as well. Note that this ensures that $\boldsymbol{\alpha}$ is always primal feasible, i.e. ((2), (i–ii)) are always fulfilled. Since the KKT conditions (2) are sufficient for optimality (see Section 3), $\boldsymbol{\alpha}$ is optimal as soon as the remaining conditions ((2), (iii–v)) are satisfied. In practice, one usually tests whether the largest KKT violation falls below some predefined threshold. If so, the method stops; otherwise, the next iteration starts.

In general, SMO cannot be applied to problems of type (1) when there are multiple equality constraints. For instance, if we have $k = 2$, the two equality constraints and the $q = 2$ working set form a 2×2 linear system. If it has full rank (which we have to assume), it already restricts the feasible set to a single point, which leaves us with *zero* degrees of freedom for updating α_i and α_j.

4.2 GS, KA and SOR (q=1)

The term "minimal" in SMO stems from the fact that if $\boldsymbol{\alpha}$ is required to be primal feasible ((2), (i–ii)) at any time, and if we have one equality constraint (as in C-SVC), the working set size must be at least $q = 2$. To see this, consider the C-SVC equality constraint $\mathbf{y}^\top\boldsymbol{\alpha} = 0$: clearly, if $\boldsymbol{\alpha}$ is feasible, then changing the value of a single α_i leads to $\mathbf{y}^\top\boldsymbol{\alpha} \neq 0$ and thus to infeasibility. To overcome this problem, several modified SVM formulations have been proposed, based on the following idea: In C-SVC for instance, the equality constraint $\mathbf{y}^\top\boldsymbol{\alpha} = 0$ in (1) originates from the bias term in the decision function (usually referred to as b). Therefore, changing the role of the bias will also change the structure of the optimization problem (1). This boils down to two possible modifications. The bias term may be either penalized, as in *Successive Over-Relaxation* (SOR) by [9] or simply left out, as in *Kernel Adatron* (KA) [7]. In both cases, the primal problem (1) is left with the box constraints only and can be solved with $q = 1$ methods. KA and SOR are equivalent in the sense that they essentially implement *Gauss-Seidel* (GS) iterations (or the closely related SOR iterations) to solve the linear system

$$\mathbf{H}\boldsymbol{\alpha} = -\mathbf{f}_0 \tag{3}$$

subject to the box constraints $0 \leq \alpha_i \leq d$. An actual implementation can be very similar to SMO: first, we chose $\boldsymbol{\alpha} = \mathbf{0}$ as a feasible starting point. At each iteration, a working set $\{\alpha_i\}$ is selected and the objective is minimized — in contrast to SMO, without any constraints: $\Delta\alpha_i = -(\mathbf{H}\boldsymbol{\alpha} + \mathbf{f}_0)_i/\mathbf{H}_{ii}$, which is a GS step. Then, the new value for α_i is clipped to the interval $\left[0; \frac{C}{m}\right]$ which ensures primal feasibility. As with SMO, if ((2), (iii–v)) are met, the method stops. It can be shown that such box-constrained GS methods converge from any starting point $\boldsymbol{\alpha}$, given that $\mathbf{H} \succeq \mathbf{0}$.

4.3 Platt vs. Gauss

To summarize,

- SVMs with *one* equality constraint (e.g. those listed in Table 1, except ν-SVC and ν-SVR) can be solved in a straightforward manner using SMO. In contrast, GS methods cannot deal with any equality constraint and are thus not applicable to any SVM problem in Table 1, unless the problem is modified accordingly.
- SVMs with *two* equality constraints (e.g. ν-SVC and ν-SVR) cannot be solved in general with either SMO or GS.
- SVMs with *three or more* equality constraints (e.g. semi-parametric SVMs) cannot be solved at all with either SMO or GS.

As an aside, the *LIBSVM* implementation [3] *does* solve ν-SVC and ν-SVR via SMO. In the initial version [2], this was accomplished through a bias penalty that eliminates one of the equality constraints (as in SOR). In its current version, *LIBSVM* uses a special working set selection heuristic which makes this modification obsolete [3, 4].

5 A Minimal Primal-Dual Method

This section develops the minimal primal-dual (MPD) algorithm for SVM training. It should be mentioned that other primal-dual methods have been proposed for this purpose [12]. However, to our knowledge, these approaches neither exploit the fact that the dual function is approximately quadratic, nor do they scale well. In the spirit of [11], we call our method "minimal", since the working sets have minimum possible size $q = 1$.

5.1 Motivation

Recall that SOR and KA (Section 4.2) exploit the fact that the equality constraint $\mathbf{y}^\top \boldsymbol{\alpha} = 0$ in C-SVC is closely connected to the bias term b in the decision function. In particular, if we *remove* the bias term from our problem (KA), or if we *penalize* it via $\frac{1}{2}b^2$ (SOR), the constraint $\mathbf{y}^\top \boldsymbol{\alpha} = 0$ vanishes. In fact, the equality constraint also vanishes if we *fix* the bias to *any* value. It is easy to show (see e.g. [3]) that the dual variable η_1 associated with the equality constraint $\mathbf{y}^\top \boldsymbol{\alpha} = 0$ actually *is* the bias b. As a result, we may remove equality constraints by fixing their associated dual variables η_i. In the following, we show how this motivates the MPD method. To begin, consider the Lagrangian of (1), for the moment without the box constraints, i.e.

$$L(\boldsymbol{\alpha}, \eta_1) = \frac{1}{2}\boldsymbol{\alpha}^\top \mathbf{H}\boldsymbol{\alpha} + \mathbf{f}_0^\top \boldsymbol{\alpha} + \boldsymbol{\eta}^\top (\mathbf{F}^\top \boldsymbol{\alpha} - \mathbf{e}) \tag{4}$$

which is quadratic in the primal variables $\boldsymbol{\alpha}$ and linear in the dual variables $\boldsymbol{\eta}$. Since we required $\mathbf{H} \succeq \mathbf{0}$, it is convex and bounded from below and therefore the infimum exists for any $\boldsymbol{\eta}$. Now we put the box constraints back in as a (non-relaxed) domain restriction on $\boldsymbol{\alpha}$. This yields the dual function

$$g(\boldsymbol{\eta}) = \inf_{0 \le \alpha_i \le d} L(\boldsymbol{\alpha}, \boldsymbol{\eta}), \tag{5}$$

whose maximization w.r.t $\boldsymbol{\eta}$ constitutes the dual problem of (1). Notice that unlike the equality constraints, which have been relaxed via the Lagrange multipliers $\boldsymbol{\eta}$, the inequalities are still present. In practice, the latter are met by clipping the unconstrained minimizer of (4) to the polytope described by the box-constraints. This operation has combinatoric complexity, but is well tractable for small working sets (see SMO etc.).

By construction, $g(\boldsymbol{\eta})$ is concave in $\boldsymbol{\eta}$, i.e. we are left with a k-dimensional, unconstrained, concave maximization problem. Now assume that we know the optimal dual variable $\boldsymbol{\eta}^*$. As mentioned earlier, since all constraints are affine and since the Lagrangian is convex in $\boldsymbol{\alpha}$, we have strong duality. Thus, finding the saddlepoint in $(\boldsymbol{\alpha}, \boldsymbol{\eta})$ is sufficient for the optimum. Since we know $\boldsymbol{\eta}^*$, we merely need to compute the corresponding $\boldsymbol{\alpha}$ via the new primal problem

$$\begin{aligned} &\min L(\boldsymbol{\alpha}, \boldsymbol{\eta}^*) \\ &\text{s.t. } \; 0 \le \alpha_i \le d,\, i = 1 \ldots n \end{aligned} \tag{6}$$

w.r.t. $\boldsymbol{\alpha}$. As opposed to the initial primal (1), (6) has no equality constraints. Thus, the feasibility of particular working set sizes is not affected, but we still obtain the optimal solution of the unaltered problem.

Needless to say, we cannot know $\boldsymbol{\eta}^*$ in advance. The basic idea of our MPD method is to fix some $\boldsymbol{\eta}$, solve (6) via GS — pretending that $\boldsymbol{\eta}$ is optimal — use the resulting $\boldsymbol{\alpha}$ to compute a new estimate for $\boldsymbol{\eta}$, and go back to solving (6). This is repeated until convergence, i.e. until $\boldsymbol{\eta} = \boldsymbol{\eta}^*$. The $\boldsymbol{\eta}$ estimates are updated via approximate Newton steps: since the dual function (5) is concave, we find its global optimum by merely following the direction of steepest ascent. While the derivatives of $g(\boldsymbol{\eta})$ cannot be written in closed form, the following approximation turns out to be sufficient in practice: neglecting the box constraints, we find that the minimizer of (4) reads $\boldsymbol{\alpha}^*(\boldsymbol{\eta}) = -\mathbf{H}^{-1}(\mathbf{f}_0 - \mathbf{F}\boldsymbol{\eta})$. Plugging this into (4), we get an approximation to the dual function (5), whose gradient w.r.t. $\boldsymbol{\eta}$ is given by $\mathbf{F}^\top \mathbf{H}^{-1}(\mathbf{f}_0 - \mathbf{F}\boldsymbol{\eta})$, or simply

$$\frac{\partial g}{\partial \eta_i} = \mathbf{f}_i^\top \boldsymbol{\alpha} - e_i. \tag{7}$$

if $\boldsymbol{\alpha} = \boldsymbol{\alpha}^*(\boldsymbol{\eta})$. For the second derivatives, we assume further that the Hessian is diagonal, which yields

$$\frac{\partial^2 g}{\partial \eta_i \eta_j} = \begin{cases} -\mathbf{f}_i \mathbf{H}^{-1} \mathbf{f}_i & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

Note that the gradient depends on $\boldsymbol{\eta}$, but is readily computed by virtue of equation (7). In contrast, the Hessian is constant, but costly to compute (8). In order to avoid a direct computation of $\mathbf{H}^{-1}$, we solve the k linear systems $\mathbf{H}\boldsymbol{\gamma}_i = -\mathbf{f}_i$ for γ_i via GS. After this, the diagonal elements of the Hessian (8) reduce to $\mathbf{f}_i^\top \gamma_i$. We will see that effectively, our method trades the k equality constraints for k additional GS steps per iteration.

As an aside, a less crude approximation is to *clamp* active box constraints, i.e. assume that for all $\boldsymbol{\eta}$, the set of α_i that are at bound, is constant. In (7) and (8), this amounts to setting the components of $\mathbf{f}_i$ for which the box constraint is active, to zero. In our experiments, we could not detect a significant difference between this and the above approximation (besides a slight computational disadvantage of the former), so we decided to not use the clamping heuristic.

5.2 The Algorithm

The following paragraphs describe the minimal primal-dual algorithm in detail. An outline of the method is given as Algorithm 1, we will refer to particular line numbers in the text.

Variables and Initialization. During the whole optimization process, we keep the following variables: the current primal and dual variable $\boldsymbol{\alpha}$ and $\boldsymbol{\eta}$, the primal and dual gradient of the Lagrangian $g_{\boldsymbol{\alpha}} = \nabla_{\boldsymbol{\alpha}} L(\boldsymbol{\alpha}, \boldsymbol{\eta}) = \mathbf{H}\boldsymbol{\alpha} + \mathbf{f}_0 + \mathbf{F}\boldsymbol{\eta}$ and $g_{\boldsymbol{\eta}} = \nabla_{\boldsymbol{\eta}} L(\boldsymbol{\alpha}, \boldsymbol{\eta}) = \mathbf{F}^\top \boldsymbol{\alpha} - \mathbf{e}$. Furthermore we keep estimates of the the k Hessian diagonal elements $h_i = -\mathbf{f}_i \mathbf{H}^{-1} \mathbf{f}_i$, as well as the residuals $\mathbf{r}_i = \mathbf{H}\boldsymbol{\gamma}_i + \mathbf{f}_i$ of the corresponding linear systems. The initial value of the primal variable $\boldsymbol{\alpha}$ can be

Algorithm 1. Minimal Primal-Dual SVM Training

```
 1: initialize primal/dual variables
 2: initialize gradients
 3: initialize variables for the Hessian estimation

 4: loop
 5:    update dual Hessian estimate
 6:    update primal variables
 7:    if (primal optimal) then
 8:       if (dual gradient is small) then
 9:          converged
10:       else
11:          update dual variables
12:       end if
13:    end if
14: end loop
```

chosen arbitrarily up to the box constraints $0 \leq \alpha_i \leq d$ (see the discussion on feasibility below). For the sake of simplicity we set $\boldsymbol{\alpha} = \mathbf{0}$ (Line 2). The dual variables η_i are completely unconstrained and we set them to zero as well. This implies to $g_{\boldsymbol{\alpha}} = \mathbf{f}_0$ and $g_{\boldsymbol{\eta}} = -\mathbf{e}$ (Line 3). Finally, we initialize $\boldsymbol{\gamma}_i = \mathbf{0}$ and thus $h_i = 0$, $\mathbf{r}_i = \mathbf{f}_i$, $i = 1 \ldots k$ (Line 4).

Hessian and Primal Updates At the beginning of each iteration, we update the Hessian estimates h_i (Line 7). To this end, we do the following for all $i = 1 \ldots k$: we pick the largest component of the residual $\mathbf{r}_i$, $\mathbf{r}_{ij}$ say, and perform a GS step $\Delta\boldsymbol{\gamma}_{ij} = -\mathbf{r}_{ij}/\mathbf{H}_{jj}$ in that direction. Note that we never actually compute the $\boldsymbol{\gamma}_i$, instead, we update h_i and r_i via $\Delta h_i = -\Delta\boldsymbol{\gamma}_{ij}\mathbf{f}_{ij}$ and $\Delta\mathbf{r}_i = \Delta\boldsymbol{\gamma}_{ij}\mathbf{H}_j$. The next step is the primal update (Line 8) via box-constrained GS. The working set is chosen to be the α_i corresponding to the largest KKT violation ((2), (iii-v)), which is in fact a $q = 1$ version of the heuristic used by [8] and [3]. Note that this is an $\mathcal{O}(n)$ operation since we keep $\nabla_{\boldsymbol{\alpha}} L(\boldsymbol{\alpha}, \boldsymbol{\eta})$, which is in fact the quantity $\mathbf{s}$ in (2). The GS update amounts to $\Delta\alpha_i = -s_i/\mathbf{H}_{ii}$ and the subsequent clipping of the new value to $[0; d]$. A change in α_i also affects the gradients, so we update them as well via $\Delta g_{\boldsymbol{\alpha}} = \delta\mathbf{H}_i$ and $\Delta g_{\boldsymbol{\eta}} = \delta\mathbf{F}_i$, where δ is the actual change in α_i after clipping.

Primal Stopping The primal updates ensured that the box constraints (and thus ((2), (ii)) are always met. Therefore, the condition in Line 9 evaluates to true as soon as ((2), (iii-v)) hold. To this end, we check if the largest (in magnitude) violation of ((2), (iii-v)) falls below a predefined primal threshold ϵ_p. Note that this can be done efficiently since the variable $\mathbf{s}$ in (2) is simply the primal gradient $\nabla_{\boldsymbol{\alpha}} L(\boldsymbol{\alpha}, \boldsymbol{\eta})$.

Dual Stopping In Line 10, we check the current solution $\boldsymbol{\alpha}, \boldsymbol{\eta}$ for optimality. At this point we already have ((2), (ii-v)) satisfied, so what remains to be checked are the violations ((2), (i)), which coincide with the components of the dual gradient $\nabla_{\boldsymbol{\eta}} L(\boldsymbol{\alpha}, \boldsymbol{\eta})$. We therefore say that the algorithm has converged if the largest (in magnitude) component of $\nabla_{\boldsymbol{\eta}} L(\boldsymbol{\alpha}, \boldsymbol{\eta})$ is smaller than the dual stopping threshold ε_d.

Dual Updates. If the optimality check in Line 10 fails, we update the dual variables via Newton steps (Line 13) $\Delta\eta_i = -(\mathbf{f}_i^\top \boldsymbol{\alpha} - \mathbf{e})/h_i$. As with the primal updates, a change in $\boldsymbol{\eta}$ results in a change in $g_{\boldsymbol{\alpha}}$, so we update the latter via $\Delta g_{\boldsymbol{\alpha}} = \mathbf{F}\Delta\boldsymbol{\eta}$.

5.3 Implementation

The minimal primal-dual method is designed to solve various SVM problems of the form (1) in a consistent manner. Thus, it seems natural to implement it in a two-layered fashion: a core function that implements Algorithm 1, and a wrapper function that implements the mappings in Table 1. The user may then call the wrapper function with the data and the type of SVM as arguments. The wrapper functions translates this into values for $\mathbf{H}$, $\mathbf{f}_i$, etc. and calls the core function. When the core function returns, the wrapper computes the values of the so-called free variables (such as the bias term b). This is extremely simple, since the free variables not only correspond to the equality constraints, but actually coincide with the respective dual variables η_i. This is sometimes referred to as the "dual-dual trick" [12]. It can be seen as an advantage of our algorithm (or primal-dual methods in general) over the methods described in Section 4, since the latter require a manual recovery of these values. The latter becomes difficult if the solution is degenerate, e.g. if the solution of a C-SVC lacks unbounded support vectors (see [3]). Algorithm 1 (the core) and Table 1 (the wrapper) are readily implemented in a few lines of *Matlab*, which is the version we used in our experiments. It can be downloaded from `www.kyb.mpg.de/∼kienzle`.

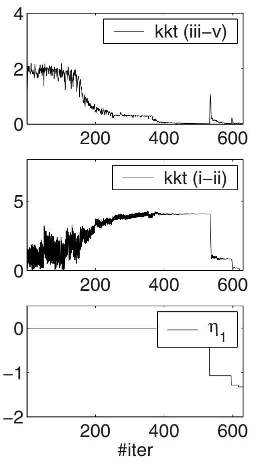

Fig. 1. Typical values for the KKT violations and the dual variable η_1 during MPD training of a C-SVC, plotted against the number of iterations. The data were two 2D Gaussians with 20 points each, standard deviation $\sigma = 1$, and their means $\sqrt{2}$ apart. We trained a C-SVC with $C = 1000$ and a Gaussian kernel with $\sigma = 1$, the stopping thresholds were set to $\varepsilon_p = \varepsilon_d = .01$. The method performed four dual updates (at iteration 532, 595, 616 and 627) among a total of 630 iterations. Notice the complementary nature of the top two plots: the primal/dual updates reduce the (iii-v)/(i) KKT violations, while sacrificing the fulfillment of the "competing" conditions (i)/(iii-v), respectively.

5.4 Complexity and Convergence

The MPD method can be seen as solving $k+1$ linear systems in parallel, where every time the primal (box constrained) system has converged, its parameters are updated using the current values from the k Hessian (unconstrained) systems. All updates are $\mathcal{O}(n)$, so each loop has time complexity $\mathcal{O}(kn)$. The total number of required operations is therefore $\mathcal{O}(tkn)$, where t is problem-dependent, as it is common to iterative methods. The memory usage is $\mathcal{O}(n)$.

It has been shown that box-constrained GS iterations (i.e. those that solve our primal problem) converge for any starting point, given that $\mathbf{H} \succeq \mathbf{0}$ (see e.g. [7] and references). The natural question arises whether the dual variables converge to their optimum $\boldsymbol{\eta}^*$. A sketch of the proof that we are currently working on is based on an idea from [1] (the actual proof will be included in a longer version of this paper): training with MPD is equivalent to solving a slightly modified problem whose solution is equivalent to that of (1), using an *Augmented Lagrangian Method*. In this setting, the method can be shown to converge if the dual stepsize is bounded from above by some threshold. Interestingly, the approximations (7) and (8) aim at satisfying this very condition. It should be mentioned that due to the approximative nature of the derivatives or because of numerical effects, the stepsizes may yet turn out too large, although we have not experienced this in our experiments. Figure 1 illustrates the typical behavior of MPD on a toy problem.

6 Experiments

In this experiment, we have tested our method on a multiple equality constraint problem called semi-parametric SV regression. To introduce this concept, recall that SVM decision (or regression) functions are linear in the reproducing kernel Hilbert space (RKHS) induced by the kernel function $K(\cdot,\cdot)$. In [13], the class of admissible (linear) functions is augmented by a parametric term. As an example, for ε-SVR we have

$$f(\mathbf{x}) = \langle \mathbf{w}, K(\mathbf{x},\cdot)\rangle + \sum_{i=1}^{k} \beta_i \varphi_i(\mathbf{x}) \tag{9}$$

instead of the usual $f(\mathbf{x}) = \langle \mathbf{w}, K(\mathbf{x},\cdot)\rangle + b$. The $\varphi_i(\mathbf{x})$ allow us to incorporate domain knowledge about the target function (e.g. the presence of a sinusoidal component in f). It has been shown that each parameter β_i introduces one equality constraint $\mathbf{f}_i$, e_i to (1), where $e_i = 0$ and $\mathbf{f}_i$ contains the values of $\varphi_i(\mathbf{x})$ on the training points $\mathbf{x}_i$. Moreover, as with all free variables, at the solution we have $\beta_i = \eta_i$ [13]. In particular, the bias term b in standard SVMs can be seen as a special case of such a β_i, namely for $\varphi_i(\mathbf{x}) = 1$.

Let us now consider the problem studied by [13], namely a semi-parametric ε-SVR on $y_i = f(\mathbf{x}_i) + \xi_i$, with $\mathbf{x}_i$ uniformly sampled from the interval $[0; 10]$ and $f(\mathbf{x}) = \sin(\mathbf{x}) + \text{sinc}(2\pi(\mathbf{x}-5)) + \xi_i$. The noise ξ_i is zero-mean and uniform with standard deviation 0.2 and the parameter of the ε-insensitive loss function

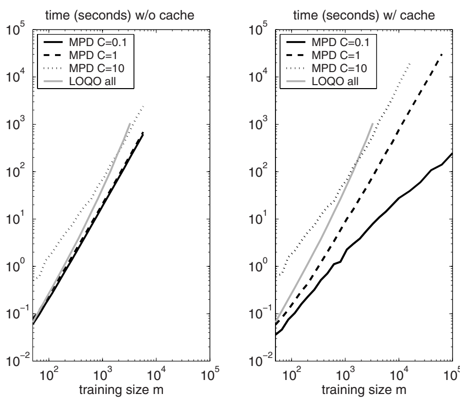

Fig. 2. Training a large-scale semi-parametric SVR with MPD vs. LOQO. The plots show cpu times against training sizes (between $m = 50$ to $m = 100000$) on a loglog scale. For the left plot, we explicitly computed the matrix $\mathbf{H}$ for both methods. Here, we increased m, until *Matlab* ran out of memory ($m \approx 3300$ for LOQO, $m \approx 5700$ for MPD). For the right plot, we used a 2Gbyte least-recently-used (LRU) kernel cache (as it is common practice, e.g. in [3]) for MPD, the LOQO plot is the same as on the left.

$\|\cdot\|_\varepsilon$ is $\varepsilon = 0.05$. The latter is not to be confused with the stopping thresholds $\varepsilon_p, \varepsilon_d$, which we set to 0.01. As in [13], our model for $f(\mathbf{x})$ has two parameters β_1, β_2 that are associated with the functions $\varphi_1(\mathbf{x}) = \sin(\mathbf{x})$, $\varphi_2(\mathbf{x}) = \mathrm{sinc}(2\pi(\mathbf{x} - 5))$. In the original experiment, the authors computed cross-validation errors for the above problem of size $m = 50$ and various values of C, which yielded an optimum at $C = 1$. Here, we test our method for three values around the optimal C, i.e. 0.1, 1 and 10, and for various training sizes $m = 50 \ldots 100000$.

Figure 2 illustrates the scaling properties of MPD training compared to that of the LOQO method [14] used in [13]; both methods are implemented in *Matlab*. We tested two versions of MPD, with and without using a cache for $\mathbf{H}$. The left plot shows the former case for $m < 3300$, which is where Matlab runs out of memory. Here, both methods yielded the same accuracy, and comparable time complexity (note that slopes on the loglog scale correspond to exponents on a linear scale). The right plot shows the performance of MPD with kernel caching (see caption). This experiment supports that MPD can deal with large training sets, e.g. up to $m = 10^5$ or more while it has a similar (in fact, slightly lower) time complexity than the LOQO method. Note that in [13], LOQO was chosen due to the lack of a more appropriate method. We suspect that it is the training size limit which prevented semi-parametric SVMs from gaining more attention in the past. MPD does not have this limitation and thus allows for further exploration of not only this but other equally interesting paradigms.

7 Discussion and Future Work

We have presented a large scale primal-dual method for SVM training. The main contributions are the following: First, the MPD method combines all standard SVM formulations into a unifying framework. Second, as with SMO or GS methods, it is straightforward to implement, scales well, and does not require external QP packages. Third, MPD is able to implement more general SVM paradigms, e.g. semi-parametric SVMs, that cannot be solved with SMO based packages, and for which generic solvers run out of memory.

What remains to be explored is whether the convergence of the Hessian estimates should be enforced *before* any dual updates are performed, since this is required for the Newton step approximations to be reliable. In the current version, we merely rely on the fact that the estimate will *eventually* have converged. Finally, in order to compare MPD to existing methods, note that for $k = 0$, MPD is equivalent to KA/SOR. A comparison of SOR and SMO can be found in [9].

Acknowledgments

The authors would like to thank Oliver Chapelle, Alexander Zien and Kristin Bennett for useful comments. This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views.

References

1. D. P. Bertsekas. *Nonlinear Programming.* Athena Scientific, 1999.
2. C.-C. Chang and C.-J. Lin. Training ν-support vector classifiers: Theory and algorithms. *Neural Computation*, 13(9):2119–2147, 2001.
3. C.-C. Chang and C.-J. Lin. LIBSVM – a library for support vector machines, version 2.71, `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`, August 2004.
4. P.-H. Chen, C.-J. Lin, and B. Schölkopf. A tutorial on ν-support vector machines. *Applied Stochastic Models in Business and Industry (to appear)*, 2005.
5. R. Collobert and S. Bengio. SVMTorch: Support vector machines for large-scale regression problems. *Journal of Machine Learning Research*, 1:143–160, 2001.
6. J. Dong, A. Krzyzak, and C. Y. Suen. Fast svm training algorithm with decomposition on very large data sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(4):603–618, April 2005.
7. T. Friess, N. Cristianini, and C. Campbell. The kernel adatron algorithm: a fast and simple learning procedure for support vector machine. In *International Conference on Machine Learning*, pages 188–196, 1998.
8. T. Joachims. Making large-scale SVM learning practical. In B. Schlkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods*, pages 42–56. MIT Press, 1999.
9. O. L. Mangasarian and D. R. Musicant. Successive overrelaxation for support vector machines. *IEEE Transactions on Neural Networks*, 10:1032–1037, 1999.
10. E. Osuna, R. Freund, and F. Girosi. Support vector machines: Training and applications. Technical Report AIM-1602, 1997.
11. J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schlkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods*, pages 185–208. MIT Press, 1998.
12. B. Schölkopf and A. J. Smola. *Learning with Kernels.* MIT Press, Cambridge, MA, 2002.
13. A. J. Smola, T. Friess, and B. Schölkopf. Semiparametric support vector and linear programming machines. In *Advances in Neural Information Processing Systems 11*, pages 585–591, 1998.
14. R. J Vanderbei. LOQO: An interior point code for quadratic programming. Technical Report SOR 94-15, Princeton University, NJ, 1994.