Dual SVM Training on a Budget

Sahar Qaadan, Merlin Schüler and Tobias Glasmachers
Department of Neural Computing, Ruhr University Bochum, 44801, Bochum, Germany

Keywords: Dual Subspace Algorithm, Kernel SVM, Budget Maintenance, Merging Method.

Abstract: We present a dual subspace ascent algorithm for support vector machine training that respects a budget constraint limiting the number of support vectors. Budget methods are effective for reducing the training time of kernel SVM while retaining high accuracy. To date, budget training is available only for primal (SGD-based) solvers. Dual subspace ascent methods like sequential minimal optimization are attractive for their good adaptation to the problem structure, their fast convergence rate, and their practical speed. By incorporating a budget constraint into a dual algorithm, our method enjoys the best of both worlds. We demonstrate considerable speed-ups over primal budget training methods.

1 INTRODUCTION

Support Vector Machines (SVMs) introduced by (Cortes and Vapnik, 1995) are popular machine learning methods, in particular for binary classification. They are supported by learning-theoretical guarantees (Mohri et al., 2012), and they exhibit excellent generalization performance in many applications in science and technology (Son et al., 2010; Shigeo, 2005; Byun and Lee, 2002; Quinlan et al., 2003).They belong to the family of kernel methods, applying a linear algorithm in a feature space defined implicitly by a kernel function.

Training an SVM corresponds to solving a large-scale optimization problem, which can be cast into a quadratic program (QP). The primal problem can be solved directly with stochastic gradient descent (SGD) and accelerated variants (Shalev-Shwartz et al., 2007; Glasmachers, 2016), while the dual QP is solved with subspace ascent, see (Bottou and Lin, 2006) and references therein.

The computational complexity of each stochastic gradient or coordinate step is governed by the cost of evaluating the model of a training point. This cost is proportional to the number of support vectors, which grows at a linear rate with the data set size (Steinwart, 2003). This limits the applicability of kernel methods to large-scale data. Efficient algorithms are available for linear SVMs (SVMs without kernel) (Shalev-Shwartz et al., 2007; Fan et al., 2008). Parallelization can yield considerable speed-ups (Wen et al., 2017), but only by a constant factor. For non-linear (kernelized) SVMs there exists a wide variety of approaches for approximate SVM training, many of which aim to leverage fast linear solvers by approximating the feature space representation of the data. The approximation can either be fixed (e.g., random Fourier features) (Lu et al., 2016; Le et al., 2016; Nguyen et al., 2017) or data-dependent (e.g., Nyström sampling) (Lu et al., 2016; Rahimi and Recht, 2008; Yang et al., 2012; Calandriello et al., 2017).

Several online algorithms fix the number of SVs to pre-specified value $B \ll n$, the budget, and update the model in a greedy manner. This way they limit the iteration complexity. The *Stoptron* (Orabona et al., 2009) is a simple algorithm that terminates when the number of SVs reaches the budget B. The *Forgetron* algorithm (Dekel et al., 2008) removes the oldest SV when the number of SVs exceeds the budget. Under some mild assumptions, convergence of the algorithm has been proven. The *Projectron* (Orabona et al., 2009) projects the SV to be removed on the remaining SVs to minimize the weight degradation. While the latter outperforms the Forgetron, it requires $O(B)^3$ time to compute the projection. Recently, (Lu et al., 2018) have implemented an efficient stochastic sampling strategy based on turning the incoming training example into a new support vector with probability proportional to the loss suffered by the example. This method aims to convert the online classifiers for batch classification purposes.

The above approach can be based on stochastic gradient descent and combined with more elaborate budget maintenance techniques (Dekel and Singer, 2007). In particular with the popular budget maintenance heuristic of merging support vectors (Wang

Qaadan, S., Schüler, M. and Glasmachers, T.
Dual SVM Training on a Budget.
DOI: 10.5220/0007346400940106
In *Proceedings of the 8th International Conference on Pattern Recognition Applications and Methods (ICPRAM 2019)*, pages 94-106
ISBN: 978-989-758-351-3

et al., 2012), it goes beyond the above techniques by adapting the feature space representation during training. The technique is known as budgeted stochastic gradient descent (BSGD).

In this context we design the first dual SVM training algorithm with a budget constraint. The solver aims at the efficiency of dual subspace ascent as used in LIBSVM, ThunderSVM, and also in LIBLINEAR (Chang and Lin, 2011; Fan et al., 2008; Wen et al., 2017), while applying merging-based budget maintenance as in the BSGD method (Wang et al., 2012). The combination is far from straight-forward, since continually changing the feature representation also implies changing the dual QP, which hence becomes a moving target. Nevertheless, we provide guarantees roughly comparable to those available for BSGD.

In a nutshell, our contributions are:

- We present the first dual decomposition algorithm operating on a budget,
- we analyze its convergence behavior,
- and we establish empirically its superiority to primal BSGD.

The structure of the paper is as follows: In the next section we introduce SVMs and existing primal and dual solvers, including BSGD. Then we present our novel dual budget algorithm and analyze its asymptotic behavior. We compare our method empirically to BSGD and validate our theoretical analysis. We close with our conclusions.

2 SUPPORT VECTOR MACHINE TRAINING

A Support Vector Machine is a supervised kernel learning algorithm (Cortes and Vapnik, 1995). Given labeled training data $(x_1,y_1),\dots,(x_n,y_n) \in X \times Y$ and a kernel function $k : X \times X \to \mathcal{R}$ over the input space, the SVM decision function $f(x) \mapsto \langle w, \phi(x)\rangle$ (we drop the bias, c.f. (Steinwart et al., 2011)) is defined as the optimal solution w^* of the (primal) optimization problem

$$\min_{w\in\mathcal{H}} \quad P(w) = \frac{\lambda}{2}\|w\|^2 + \frac{1}{n}\sum_{i=1}^{n} L\big(y_i, f(x_i)\big), \qquad (1)$$

where $\lambda > 0$ is a regularization parameter, L is a loss function (usually convex in w, turning problem (1) into a convex problem), and $\phi : X \to \mathcal{H}$ is an only implicitly defined feature map into the reproducing kernel Hilbert space $\mathcal{H}$, fulfilling $\langle \phi(x), \phi(x')\rangle = k(x,x')$. The representer theorem allows to restrict the solution to the form $w = \sum_{i=1}^{n} \alpha_i y_i \phi(x_i)$ with coefficient vector $\alpha \in \mathcal{R}^n$, yielding $f(x) = \sum_{i=1}^{n} \alpha_i y_i k(x,x_i)$. Training points x_i with non-zero coefficients $\alpha_i \neq 0$ are called support vectors.

We focus on the simplest case of binary classification with label space $Y = \{-1,+1\}$, hinge loss $L(y, f(x)) = \max\{0, 1 - yf(x)\}$, and classifier $x \mapsto \text{sign}(f(x))$, however, noting that other tasks like multi-class classification and regression can be tackled in the exact same framework, with minor changes. For binary classification, the equivalent dual problem (Bottou and Lin, 2006) reads

$$\max_{\alpha\in[0,C]^n} \quad D(\alpha) = \mathbb{1}^T\alpha - \frac{1}{2}\alpha^T Q \alpha, \qquad (2)$$

which is a box-constrained quadratic program (QP), with $\mathbb{1} = (1,\dots,1)^T$ and $C = \frac{1}{\lambda n}$. The matrix Q consists of the entries $Q_{ij} = y_i y_j k(x_i,x_j)$.

Kernel SVM Solvers. Dual decomposition solvers like LIBSVM (Chang and Lin, 2011; Bottou and Lin, 2006) are the method of choice for obtaining a high-precision non-linear (kernelized) SVM solution. They work by decomposing the dual problem into a sequence of smaller problems of size $O(1)$, and solving the overall problem in a subspace ascent manner. For problem (2) this can amount to coordinate ascent (CA). Keeping track of the dual gradient $\nabla_\alpha D(\alpha) = \mathbb{1} - Q\alpha$ allows for the application of elaborate heuristics for deciding which coordinate to optimize next, based on the violation of the Karush-Kuhn-Tucker conditions or even taking second order information into account. Provided that coordinate i is to be optimized in the current iteration, the sub-problem restricted to α_i is a one-dimensional QP, which is solved optimally by the truncated Newton step

$$\alpha_i \leftarrow \left[\alpha_i + \frac{1 - Q_i\alpha}{Q_{ii}}\right]_0^C, \qquad (3)$$

where Q_i is the i-th row of Q and $[x]_0^C = \max\{0, \min\{C, x\}\}$ denotes truncation to the box constraints. The method enjoys locally linear convergence (Lin, 2001), polynomial worst-case complexity (List and Simon, 2005), and fast convergence in practice.

In principle the primal problem (1) can be solved directly, e.g., with SGD, which is at the core of the kernelized Pegasos algorithm (Shalev-Shwartz et al., 2007). Replacing the average loss (empirical risk) in equation (1) with the loss $L(y_i, f(x_i))$ on a single training point selected uniformly at random provides an unbiased estimate. Following its (stochastic) subgradient with learning rate $1/(\lambda t) = (nC)/t$ in iteration t yields the update

$$\alpha \leftarrow \alpha - \frac{\alpha}{t} + \mathbf{1}_{\{y_i f(x_i) < 1\}} \frac{nC}{t} e_i, \qquad (4)$$

where e_i is the i-th unit vector and $\mathbf{1}_{\{E\}}$ is the indicator function of the event E. Despite fast initial progress, the procedure can take a long time to produce accurate results, since SGD suffers from the non-smooth hinge loss, resulting in slow convergence.

In both algorithms, the iteration complexity is governed by the computation of $f(x)$ (or equivalently, by the update of the dual gradient), which is linear in the number of non-zero coefficients α_i. This is a limiting factor when working with large-scale data, since the number of support vectors is usually linear in the data set size n (Steinwart, 2003).

Linear SVM Solvers. Specialized solvers for linear SVMs with $X = \mathcal{R}^d$ and ϕ chosen as the identity mapping exploit the fact that the weight vector $w \in \mathcal{R}^d$ can be represented directly. This lowers the iteration complexity from $O(n)$ to $O(d)$ (or the number of non-zero features in x_i), which often results in significant savings (Joachims, 2006; Shalev-Shwartz et al., 2007). This works even for dual CA by keeping track of the direct representation w and the (redundant) coefficients α, however, at the price that the algorithm cannot keep track of the dual gradient any more, which would be an $O(n)$ operation. Therefore the LIBLINEAR solver resorts to uniform coordinate selection (Fan et al., 2008), which amounts to stochastic coordinate ascent (SCA) (Nesterov, 2012).

Linear SVMs shine on application domains like text mining, with sparse data embedded in high-dimensional input spaces. In general, for moderate data dimension $d \ll n$, separation of the data with a linear model is a limiting factor that can result in severe under-fitting.

SVMs on a Budget. Lowering the iteration complexity is also the motivation for introducing an upper bound or budget $B \ll n$ on the number of support vectors. The budget B is exposed to the user as a hyperparameter of the method. The proceeding amounts to approximating w with a vector $\tilde{w}$ from the non-trivial fiber bundle

$$W_B = \left\{ \sum_{j=1}^{B} \beta_j \phi(\tilde{x}_j) \,\middle|\, \beta_1, \dots, \beta_B \in \mathcal{R};\ \ \tilde{x}_1, \dots, \tilde{x}_B \in \mathcal{R}^d \right\} \subset \mathcal{H}.$$

Critically, W_B is in general non-convex, and so are optimization problems over this set. Each SGD step (eq. (4)) adds at most one new support vector to the model. If the number of support vectors exceeds B after such a step, then the budgeted stochastic gradient descent (BSGD) method applies a budget maintenance heuristic to remove one support vector. Merging of two support vectors has proven to be a good compromise between the induced error and the resulting computational effort (Wang et al., 2012). It amounts to replacing $\beta_i\phi(\tilde{x}_i) + \beta_j\phi(\tilde{x}_j)$ (with carefully chosen indices i and j) with a single term $\beta'\phi(\tilde{x}')$, aiming to minimize the “weight degradation” error $\|\beta_i\phi(\tilde{x}_i) + \beta_j\phi(\tilde{x}_j) - \beta'\phi(\tilde{x}')\|^2$. For the widely used Gaussian kernel $k(x,x') = \exp(-\gamma\|x-x'\|^2)$ the optimal $\tilde{x}'$ lies on the line spanned by $\tilde{x}_i$ and $\tilde{x}_j$, and it is a convex combination if merging is restricted to points of the same class. The coefficient h of the convex combination $\tilde{x}' = (1-h)\tilde{x}_i + h\tilde{x}_j$ is found with golden section search, and the optimal coefficient β' is obtained in closed form. This procedure was defined by (Wang et al., 2012). We use it throughout the rest of the paper. For completeness sake, the pseudo-code of the procedure is found in Algorithm 1. It selects two points from the model with an $O(B)$ heuristic that aims to minimize the squared weight degradation. First the point with the smallest coefficient is selected. The second point is chosen to minimize the squared weight degradation when merged with the first. These two points are then merged, i.e., removed from the model and replaced with the best single-point approximation. We refer to (Wang et al., 2012) for details on the efficient calculation of the squared weight degradation, as well as on the golden section search procedure used to obtain the coefficient h.

Algorithm 1: Procedure Budget Maintenance for a sparse model M.

```
Input/Output: model $M = \{(\beta_i, \tilde{x}_i)\}_{i \in I}$
$(\beta_{\min}, \tilde{x}_{\min}) \leftarrow \arg\min\{|\beta| \mid (\beta, \tilde{x}) \in M\}$
$WD^* \leftarrow \infty$
for $(\beta, \tilde{x}) \in M \setminus \{(\beta_{\min}, \tilde{x}_{\min})\}$ do
    $m \leftarrow \beta/(\beta + \beta_{\min})$
    $\kappa \leftarrow k(\tilde{x}, \tilde{x}_{\min})$
    $h \leftarrow \arg\max\{m\kappa^{(1-h')^2} + (1-m)\kappa^{h'^2} \mid h' \in [0,1]\}$
    $\beta_z \leftarrow \beta_{\min} \cdot \kappa^{(1-h)^2} + \beta \cdot \kappa^{h^2}$
    $WD \leftarrow \beta_{\min}^2 + \beta^2 - \beta_z^2 + 2 \cdot \beta_{\min} \cdot \beta \cdot \kappa$
    if $(WD < WD^*)$ then
        $WD^* \leftarrow WD$
        $(\beta^*, \tilde{x}^*, h^*, \kappa^*) \leftarrow (\beta, \tilde{x}, h, \kappa)$
    end
end
$z \leftarrow h^* \cdot \tilde{x}_{\min} + (1-h^*) \cdot \tilde{x}^*$
$\beta_z \leftarrow \beta_{\min} \cdot (\kappa^*)^{(1-h^*)^2} + \beta^* \cdot (\kappa^*)^{(h^*)^2}$
$M \leftarrow M \setminus \{(\beta_{\min}, \tilde{x}_{\min}), (\beta^*, \tilde{x}^*)\} \cup \{(\beta_z, z)\}$
```

In effect, merging allows BSGD to move support vectors around in the input space. This is well justified since restricted to W_B the representer theorem does not hold. (Wang et al., 2012) show that asymptotically the performance is governed by the approximation error implied by $w^* \notin W_B$ (see their Theorem 1).

BSGD aims to achieve the best of two worlds, namely a reasonable compromise between statistical and computational demands: fast training is achieved through a bounded computational cost per iteration, and the application of a kernel keeps the model sufficiently flexible. This requires that $B \ll n$ basis functions are sufficient to represent a model $\tilde{w}$ that is sufficiently close to the optimal model w^*. This assumption is very reasonable, in particular for large n.

3 DUAL COORDINATE ASCENT WITH BUDGET CONSTRAINT

In this section we present our novel approximate SVM training algorithm. At its core it is a dual decomposition algorithm, modified to respect a budget constraint. It is designed such that the iteration complexity is limited to $O(B)$ operations, and is hence independent of the data set size n. Our solver combines components from decomposition methods (Osuna et al., 1997), dual linear SVM solvers (Fan et al., 2008), and BSGD (Wang et al., 2012) into a new algorithm. Like BSGD, we aim to achieve the best of two worlds: a-priori limited iteration complexity with a budget approach, combined with fast convergence of a dual decomposition solver. Both aspects speed-up the training process, and hence allow to scale SVM training to larger problems.

Introducing a budget into a standard decomposition algorithm as implemented in LIBSVM (Chang and Lin, 2011) turns out to be non-trivial. Working with a budget is rather straightforward on the primal problem (1). The optimization problem is unconstrained, allowing BSGD to replace w represented by α transparently with $\tilde{w}$ represented by coefficients β_j and flexible basis points $\tilde{x}_j$. This is not possible for the dual problem (2) with constraints formulated directly in terms of α.

This difficulty is solved by (Fan et al., 2008) for the linear SVM training problem by keeping track of w and α. We follow the same approach, however, in our case the correspondence between w represented by α and $\tilde{w}$ represented by β_j and $\tilde{x}_j$ is only approximate. This is unavoidable by the very nature of the problem. Luckily, this does not impose major additional complications.

Algorithm 2: Budgeted Stochastic Coordinate Ascent (BSCA) Algorithm.

```
**Input:** training data $(x_1,y_1),\dots,(x_n,y_n)$,
$k : X \times X \to \mathcal{R}$, $C > 0$, $B \in \mathbb{N}$
$\alpha \leftarrow 0$, $M \leftarrow \emptyset$
while not happy do
    select index $i \in \{1,\dots,n\}$ uniformly at random
    $\tilde{f}(x_i) = \sum_{(\beta,\tilde{x}) \in M} \beta k(x_i,\tilde{x})$
    $\delta = \left[\alpha_i + \left(1 - y_i \tilde{f}(x_i)\right)/Q_{ii}\right]_0^C - \alpha_i$
    if $\delta \neq 0$ then
        $\alpha_i \leftarrow \alpha_i + \delta$
        $M \leftarrow M \cup \{(\delta, x_i)\}$
        if $|M| > B$ then
            trigger budget maintenance, i.e., merge two support vectors
        end
    end
end
return $\beta$
```

The pseudo-code of our Budgeted Stochastic Coordinate Ascent (BSCA) approach is detailed in algorithm 2. It represents the approximate model $\tilde{w}$ as a set M containing tuples $(\beta, \tilde{x})$.[1] Critically, in line 2 the approximate model $\tilde{w}$ is used to compute $\tilde{f}(x_i) = \langle \tilde{w}, x_i \rangle$, so the complexity of this step is $O(B)$. This is in contrast to the computation of $f(x_i) = \langle w, x_i \rangle$, with effort linear in n. At the target iteration cost of $O(B)$ it is not possible to keep track of the dual gradient, simply because it consists of n entries that would need updating with a dense matrix row Q_i. Consequently, and in line with (Fan et al., 2008), we resort to uniform variable selection in an SCA scheme, and the role of the coefficients α is reduced to keeping track of the constraints.

For the budget maintenance procedure, the same options are available as in BSGD. It is usually implemented as merging of two support vectors, reducing a model from size $|M| = B + 1$ back to size $|M| = B$. It is understood that also the complexity of the budget maintenance procedure should be bounded by $O(B)$ operations. Furthermore, for the overall algorithm to work properly, it is important to maintain the approximate relation $\tilde{w} \approx w$. For reasonable settings of the budget B, this is achieved by non-trivial budget maintenance procedures like merging and projection (Wang et al., 2012).

We leave the stopping criterion for the algorithm open. A stopping criterion akin to (Fan et al., 2008) based on thresholding KKT violations is not viable,

[1]Note that summing over an empty index set simply yields zero.

as shown by the subsequent analysis. We therefore run the algorithm for a fixed number of iterations (or epochs), as is common for BSGD.

4 ANALYSIS OF BSCA

BSCA is an approximate dual training scheme. Therefore two questions of major interest are how quickly it approaches w^*, and how close it gets. To simplify matters somewhat, we make the assumption that the matrix Q is strictly positive definite. This ensures that the optimal coefficient vector α^* corresponding to w^* is unique. For a given weight vector $w = \sum_{i=1}^n \alpha_i y_i \phi(x_i)$, we write $\alpha(w)$ when referring to the corresponding coefficients, which are also unique. Let $w^{(t)}$ and $\alpha^{(t)} = \alpha(w^{(t)})$, $t \in \mathbb{N}$, denote the sequence of solutions generated by an iterative algorithm, using the labeled training point $(x_{i^{(t)}}, y_{i^{(t)}})$ for its update in iteration t. The indices $i^{(t)} \in \{1, \dots, n\}$ are drawn i.i.d. from the uniform distribution.

Optimization Progress of BSCA. We start by computing the single-iteration progress.

Lemma 1. *The change $D(\alpha^{(t)}) - D(\alpha^{(t-1)})$ of the dual objective function in iteration t operating on the coordinate index $i = i^{(t)} \in \{1, \dots, n\}$ equals*

$$J\left(\alpha^{(t-1)}, i, \alpha_i^{(t)} - \alpha_i^{(t-1)}\right) := \frac{Q_{ii}}{2} \left(\left[\frac{1 - Q_i \alpha^{(t-1)}}{Q_{ii}} \right]^2 - \left[\left(\alpha_i^{(t)} - \alpha_i^{(t-1)}\right) - \frac{1 - Q_i \alpha^{(t-1)}}{Q_{ii}} \right]^2 \right).$$

Proof. Consider the function $s(\delta) = D(\alpha^{(t-1)} + \delta e_i)$. It is quadratic with second derivative $-Q_{ii} < 0$ and with its maximum at $\delta^* = (1 - Q_i \alpha^{(t-1)})/Q_{ii}$. Represented by its second order Taylor series around δ^* it reads $s(\delta) = s(\delta^*) - \frac{Q_{ii}}{2}(\delta - \delta^*)^2$. This immediately yields the result. ☐

The lemma is in line with the optimality of the update (3). Based thereon we define the relative approximation error

$$E(w, \tilde{w}) := 1 - \max_{i \in \{1,\dots,n\}} \left\{ \frac{J\left(\alpha(w), i, \left[\alpha_i(w) + \frac{1 - y_i \langle \tilde{w}, \phi(x_i) \rangle}{Q_{ii}}\right]_0^C - \alpha_i(w)\right)}{J\left(\alpha(w), i, \left[\alpha_i(w) + \frac{1 - y_i \langle w, \phi(x_i) \rangle}{Q_{ii}}\right]_0^C - \alpha_i(w)\right)} \right\}. \tag{5}$$

The margin calculation in the numerator is based on $\tilde{w}$, while it is based on w in the denominator. Hence $E(w, \tilde{w})$ captures the effect of using $\tilde{w}$ instead of w in BSCA. Informally, we interpret it as a dual quantity related to the weight degradation error $\|\tilde{w} - w\|^2$. Lemma 1 implies that the relative approximation error is non-negative, because the optimal step is in the denominator, which upper bounds the fraction by one. It is continuous (and in fact piecewise linear) in $\tilde{w}$, for fixed w. Finally, it fulfills $\tilde{w} = w \Rightarrow E(w, \tilde{w}) = 0$. The following theorem bounds the suboptimality of BSCA, and it captures the intuition that the relative approximation error poses a principled limit on the achievable solution precision.

Theorem 1. *The sequence $\alpha^{(t)}$ produced by BSCA fulfills*

$$D(\alpha^*) - \mathbb{E}\left[D(\alpha^{(t)})\right] \leq \left(D(\alpha^*) + \frac{nC^2}{2}\right) \cdot \prod_{\tau=1}^{t} \left(1 - \frac{2\kappa\left(1 - E\left(w^{(\tau-1)}, \tilde{w}^{(\tau-1)}\right)\right)}{(1+\kappa)n}\right),$$

where κ is the smallest eigenvalue of Q.

Proof. Theorem 5 by (Nesterov, 2012) applied to the non-budgeted setting ensures linear convergence

$$\mathbb{E}[D(\alpha^*) - D(\alpha^{(t)})] \leq \left(D(\alpha^*) + \frac{nC^2}{2}\right) \cdot \left(1 - \frac{2\kappa}{(1+\kappa)n}\right)^t,$$

and in fact the proof establishes a linear decay of the expected suboptimality by the factor $1 - \frac{2\kappa}{(1+\kappa)n}$ in each single iteration. With a budget in place, the improvement is further reduced by the difference between the actual dual progress (the numerator in eq. 5) and the progress in the non-budgeted case (the denominator in eq. 5). By construction (see lemma 1 and eq. 5), this additive difference, when written as a multiplicative factor, amounts to $1 - E(w, \tilde{w})$ in the worst case. The worst case is reflected by taking the maximum over $i \in \{1, \dots, n\}$ in eq. 5. ☐

We conclude from Theorem 1 that the behavior of BSCA can be divided into an early and a late phase. For fixed weight degradation, the relative approximation error is small as long as the progress is sufficiently large, which is the case in early iterations. Then the algorithm is nearly unaffected by the budget constraint, and multiplicative progress at a fixed rate is achieved. Progress gradually decays when approaching the optimum, which increases the relative approximation error, until BSCA stalls. In fact, the theorem does not witness further progress for $E(w, \tilde{w}) \geq 1$. Due to $w^* \notin W_B$, the KKT violations do not decay to zero, and the algorithm approaches a limit distribution.[2] The precision to which the optimal SVM solution can be approximated is hence limited by the relative approximation error, or indirectly, by the weight degradation.

[2]BSCA does not converge to a unique point. It does not become clear from the analysis provided by (Wang et al., 2012) whether this is also the case for BSGD, or whether the decaying learning rate allows BSGD to converge to a local minimum.

Budget Maintenance Rate. The rate at which budget maintenance is triggered can play a role, in particular if the procedure consumes a considerable share of the overall runtime. In the following we highlight a difference between BSGD and BSCA. For an algorithm A let

$$p_A = \lim_{T\to\infty} \mathbb{E}\left[\frac{1}{T}\cdot\left|\left\{t\in\{1,\dots,T\}\,\middle|\, y_{i^{(t)}}\langle w^{(t-1)},\phi(x_{i^{(t)}})\rangle < 1\right\}\right|\right]$$

denote the expected fraction of optimization steps in which the target margin is violated, in the limit $t\to\infty$ (if the limit exists). The following lemma establishes the fraction for primal SGD (eq. (4)) and dual SCA (eq. (3)), both without budget.

Lemma 2. *Under the conditions (i)* $\alpha_i^*\in\{0,C\}\Rightarrow \frac{\partial D(\alpha^*)}{\partial\alpha_i}\neq 0$ *and (ii)* $\frac{\partial D(\alpha^{(t)})}{\partial\alpha_{i_t}}\neq 0$ *(excluding only a zero-set of cases) it holds* $p_{SGD}=\frac{1}{n}\sum_{i=1}^n\frac{\alpha_i^*}{C}$ *and* $p_{SCA}=\frac{1}{n}|\{i\,|\,0<\alpha_i^*<C\}|$.

Proof. The lemma considers the non-budgeted case, therefore the training problem is convex. Then the condition $\sum_{t=1}^{\infty}\frac{1}{t}=\infty$ for the learning rates ensures convergence $\alpha^{(t)}\to\alpha^*$ with SGD. This can happen only if the subtraction of $\alpha_i^{(t-1)}$ and the addition of nC with learning rate $\frac{1}{t}$ cancel out in the update equation (4) in the limit $t\to\infty$, in expectation. Formally speaking, we obtain

$$\lim_{T\to\infty}\mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T}\mathbf{1}_{\{i^{(t)}=i\}}\mathbf{1}_{\{y_{i^{(t)}}\langle w^{(t-1)},\phi(x_{i^{(t)}})\rangle<1\}}nC-\alpha_i^{(t-1)}\right]=0\quad\forall i\in\{1,\dots,n\},$$

and hence $\lim_{T\to\infty}\mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T}\mathbf{1}_{\{i^{(t)}=i\}}\mathbf{1}_{\{y_{i^{(t)}}\langle w^{(t-1)},\phi(x_{i^{(t)}})\rangle<1\}}\right]=\frac{\alpha_i^*}{nC}$. Summation over i completes the proof of the result for SGD.

In the dual algorithm, with condition (i) and the same argument as in (Lin, 2001) there exists an iteration t_0 so that for $t>t_0$ all variables fulfilling $\alpha_i^*\in\{0,C\}$ remain fixed: $\alpha_i^{(t)}=\alpha_i^{(t_0)}$, while all other variables remain free: $0<\alpha_i^{(t)}<C$. Assumption (ii) ensures that all steps on free variables are non-zero and hence contribute $1/n$ to p_{SCA} in expectation, which yields $p_{\text{SCA}}=\frac{1}{n}|\{i\,|\,0<\alpha_i^*<C\}|$. □

A point $(x_{i^{(t)}},y_{i^{(t)}})$ that violates the target margin of one is added as a new support vector in BSGD as well as in BSCA. After the first B such steps, all further additions trigger budget maintenance. Hence Lemma 2 gives an asymptotic indication of the number of budget maintenance events, provided $\tilde{w}\approx w$, i.e., if the budget is not too small. The different rates for primal and dual algorithm underline the quite different optimization behavior of the two algorithms: while (B)SGD keeps making non-trivial steps on all training points corresponding to $\alpha_i^*>0$ (support vectors w.r.t. w^*), after a while the dual algorithm operates only on the free variables $0<\alpha_i^*<C$.

5 EXPERIMENTS

In this section we compare our dual BSCA algorithm on the binary classification problems ADULT, COD-RNA, COVERTYPE, IJCNN, and SUSY, covering a range of different sizes. Moreover, we run BSCA on a smaller budget. The regularization parameter $C=\frac{1}{n\cdot\lambda}$ and the kernel parameter γ were tuned with grid search and cross-validation, see table 1. The C++ implementation of our algorithm is published on the third author website.

Table 1: Data sets used in this study, hyperparameter settings, test accuracy, number of support vectors, and training time of the full SVM model (using LIBSVM).

data set	size	features	C	γ	accuracy	#SVs	training time
SUSY	4,500,000	18	2^5	2^{-7}	80.02%	2,052,555	$504h25m38s$
COVTYPE	581,012	54	2^7	2^{-3}	75.88%	187,626	$10h5m7s$
COD-RNA	59,535	8	2^5	2^{-3}	96.33%	9,120	$53.951s$
IJCNN	49,990	22	2^5	2^1	98.77%	2,477	$46.914s$
ADULT	32,561	123	2^5	2^{-7}	84.82%	11,399	$97.152s$

Data Sets and Hyperparameters. The test problems were selected according to the following criteria, which taken together imply that applying the budget method is a reasonable choice:

- The feature dimension is not too large. Therefore a linear SVM performs rather poorly compared to a kernel machine.
- The problem size is not too small. The range of sizes spans more than two orders of magnitude.

The hyperparameters C and γ were $\log_2$ encoded and tuned on an integer grid with cross-validation using LIBSVM, i.e., aiming for the best possible performance of the non-budgeted machine. The budget was set to $B=500$ in all experiments, unless stated otherwise. This value turns out to offer a reasonable compromise between speed and accuracy on all problems under study.

5.1 Primal vs. Dual Solver

The first step is to compare dual BSCA to its closest sibling, the primal BSGD method. Both algorithms maintain the budget by means of merging of pairs of support vectors.

Optimization Performance. BSCA and BSGD are optimization algorithms. Hence it is natural to compare them in terms of primal and dual objective func-

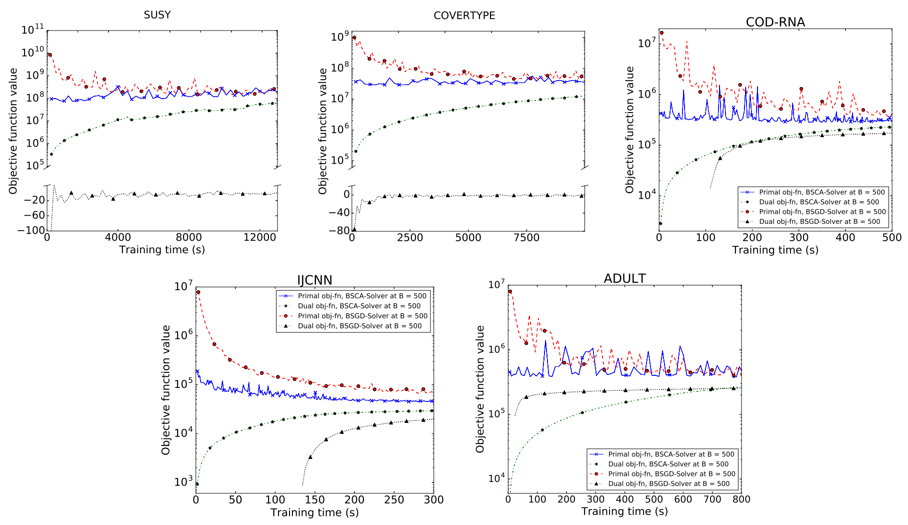

Figure 1: Primal and dual objective function curves of BSGD and BSCA solvers with a budget of $B = 500$. We use a mixed linear and logarithmic scale where the dual objective stays negative, which happens for BSGD on two problems.

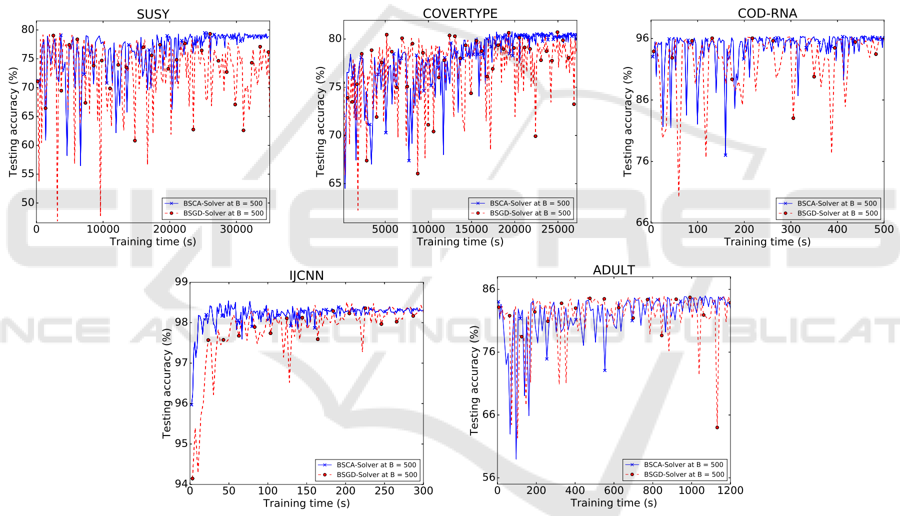

Figure 2: Test accuracy for the primal and dual solvers at a budget of $B = 500$ over training time.

tion, see equations (1) and (2). Since the solvers optimize different functions, we monitor both. However, we consider the primal as being of primary interest since its minimization is the goal of SVM training, by definition. Convergence plots are displayed in figure 1. Overall, the dual BSCA solver clearly outperforms the primal BSGD method across all problems. While the dual objective function values are found to be smooth and monotonic in all cases, this is not the case for the primal. BSCA generally oscillates less and stabilizes faster (with the exception of the ADULT problem), while BSGD remains somewhat unstable and is hence at risk of delivering low-quality solutions for a much longer time when it happens to stop at one of the peaks.

Learning Performance. Figure 2 shows the corresponding evolution of the test error. In our experiment, all budgeted models reach an accuracy that is nearly indistinguishable from the exact SVM solution. The accuracy converges significantly faster for the dual solver. For the primal solver we observe a long series of peaks corresponding to significantly reduced performance. This observation is in line with the observed optimization behavior. The effect is particularly pronounced for the largest data sets SUSY and COVERTYPE. Moreover, we conduct extensive experiments to check if the learning performance results vary in different runs depending upon the order of data-point considered in the algorithm 1. Therefore, a variance plot for each data set is shown in fig-

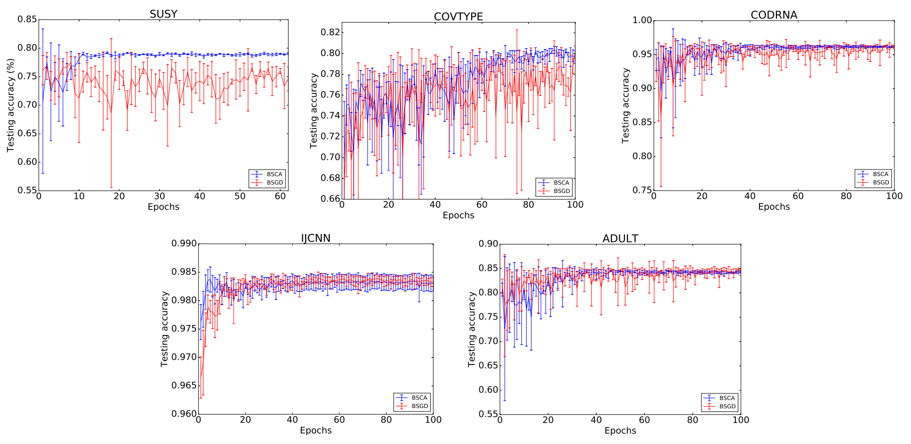

Figure 3: Stability of the test accuracy for the primal and dual solvers at a budget of $B = 500$. The plots indicate mean and standard deviation of the test accuracy over 10 independent runs.

ure 3. The plots provide variance curves for the average over 10 set of experiments.

Convergence Behavior. The next experiment validates the predictions of Lemma 2 when using a budget. Figure 8 displays the fraction of merging steps for different budget sizes applied to the dual and primal solvers. We find the predictions of the lemma being approximately valid also when using a budget. The figure highlights an interesting difference in the optimization behavior between BSGD and BSCA: while the former makes non-zero steps on all support vectors (training points with a margin of at most one), the latter manages to fix the dual variables of margin violators (training points with a margin strictly less than one) at the upper bound C.

5.2 Impact of the Budget

In this section we investigate the impact of the budget and its size on optimization and learning behavior. We start with an experiment comparing primal and dual solver without budget. The results are presented in figure 4. It is apparent that the principal differences between BSGD and BSCA remain the same when run without budget constraint, i.e., the most significant differences stem from the quite different optimization behavior of stochastic gradient descent and stochastic coordinate ascent. The SGD learning curves are quite noisy with many downwards peaks. The results are in line with experiments on linear SVMs by (Fan et al., 2008; Hsieh et al., 2008). To investigate the effect of the budget size, Figure 5 provides test accuracy curves for a reduced budget size of $B = 200$. For some of the test problems this budget it already rather small, resulting in sub-optimal learning behavior. Generally speaking, BSCA clearly outperforms BSGD. However, BSCA fails on the IJCNN data set, while BSGD fails to deliver high quality solutions on SUSY and COVERTYPE. Figure 7 aggregates the data in a different way, comparing the test accuracy achieved with different budgets on a common time axis. In this presentation it is easy to read off the speed-up achievable with a smaller budget. Unsurprisingly, BSCA with budget $B = 200$ is much faster than the same algorithm with budget $B = 500$ when run for the same number of epochs. However, when it comes to achieving a good test error quickly, the results are mixed. While the small budget apparently suffices on COVERTYPE and SUSY, the provided number of epochs does not suffice to reach good results on IJCNN, where the solver with $B = 500$ is significantly faster. Figure 6 presents a similar analysis, but with primal and dual objective function. Overall it underpins the learning (test accuracy) results, however, it also reveals a drift effect of the dual solver in particular for the smaller budget $B = 200$, with both objectives rising. This can happen if the weight degradation becomes large and the gradient computed based on the budgeted representation does not properly reflect the dual gradient any more.

5.3 Comparison of BSCA with Alternative Budgeted Solvers

In this section, we conduct an extensive set of experiments to examine the efficiency of the proposed algorithm in comparison to state-of-the-art approaches for online kernel learning. To make a fair comparison of algorithms with different parameters, the RBF kernel parameter γ and the regularization parameter C were optimized with 5-fold cross validation on the training

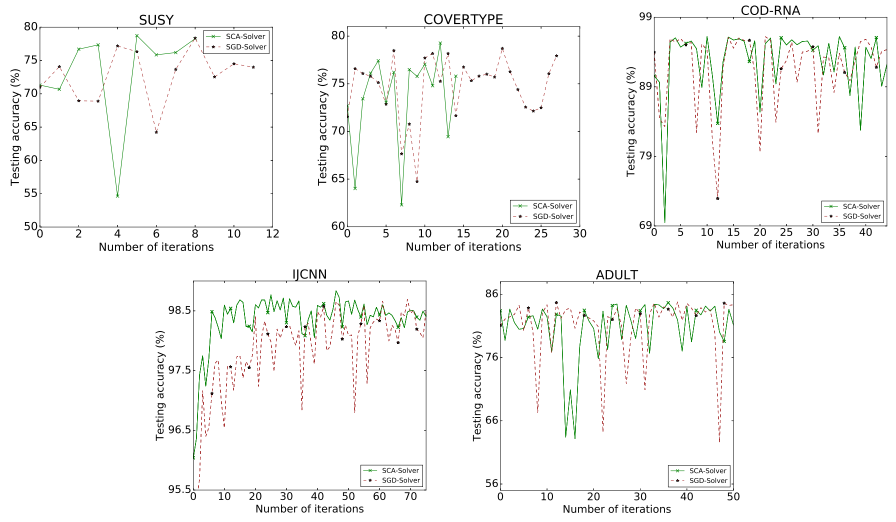

Figure 4: Test accuracy results of solvers based directly on SCA and SGD, without budget. The results are monitored every 300,000 iterations for SUSY and COVERTYPE, and every 10,000 iterations for all other data sets.

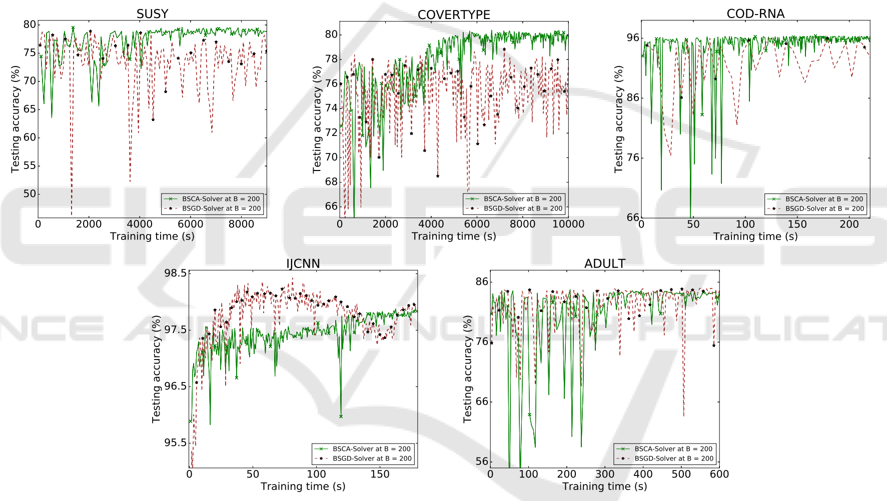

Figure 5: Test accuracy results for BSCA and BSGD at a budget of 200.

dataset using LIBSVM. The budget size *B* and learning rate η in NOGD and Pegasos and the feature dimension *D* and learning rate η in FOGD are set individually for different datasets according to the recommendation of (Lu et al., 2016). For each dataset, the model is trained in a single pass through the data. For comparison, we employ 11 state-of-the-art online kernel learning methods:

- perceptron (Freund and Schapire, 1998),
- online gradient descent (OGD) (Kivinen et al., 2003),
- randomized budget perceptron (RBP) (Cavallanti et al., 2007),
- forgetron (Dekel et al., 2008),
- projectron and projectron++ (Orabona et al., 2009),
- budgeted passive-aggressive simple (BPAS) (Wang and Vucetic, 2010),
- bounded OGD (BOGD) (Zhao et al., 2012),
- budgeted SGD using merging strategy (BSGD) (Wang et al., 2012),
- Fourier OGD (FOGD) and Nystrom OGD (NOGD) (Lu et al., 2016).

Their implementations are published as a part of the LIBSVM[3] and LSOKL5 toolboxes.[4] The results are summarized in Table 2.

Our first observation is that the budgeted online approaches demonstrate their effectiveness with substantially shorter runtimes than the ones without bud-

[3]http://www.csie.ntu.edu.tw/ cjlin/libsvm/
[4]http://lsokl.stevenhoi.org

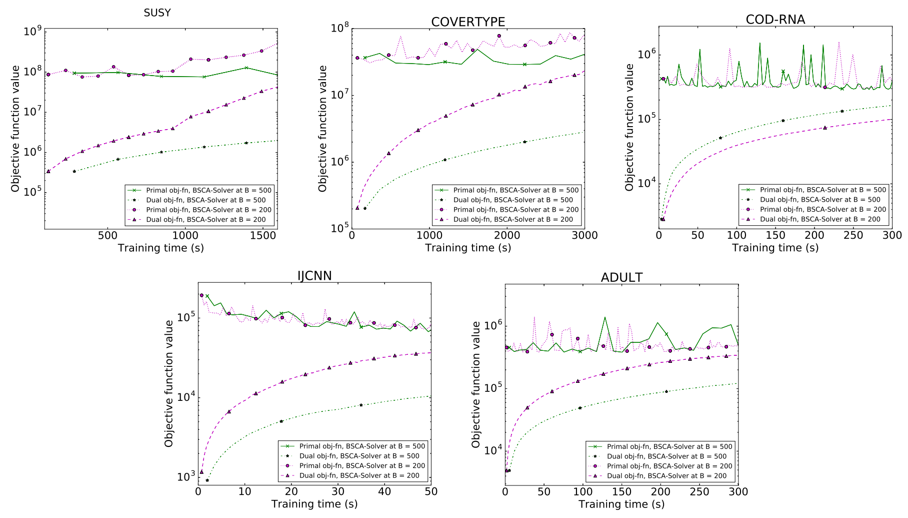

Figure 6: Primal and dual objective function over time for BSCA at budgets $B \in \{200, 500\}$.

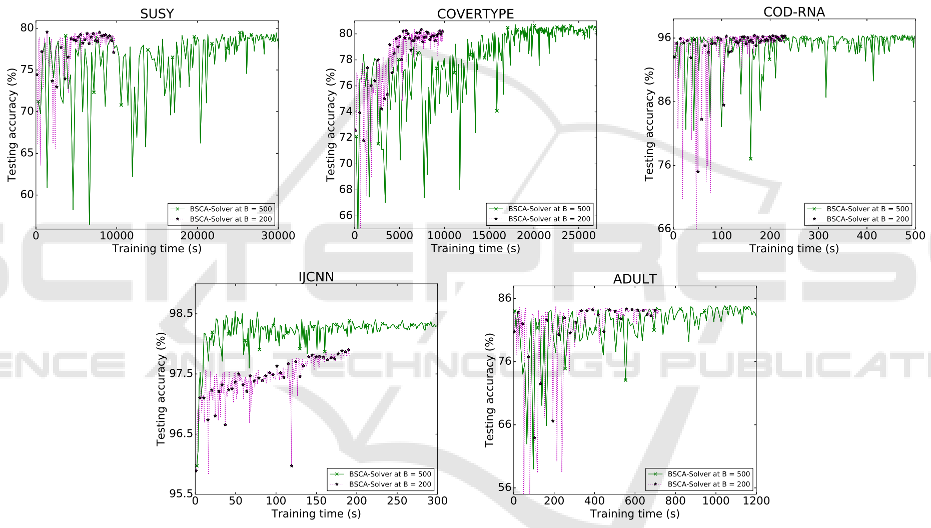

Figure 7: Test accuracy over time for BSCA at budgets $B \in \{200, 500\}$.

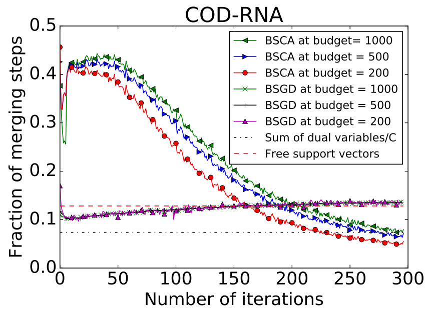

Figure 8: Fraction of merging steps over a large number of epochs at budgets $B \in \{200, 500, 1000\}$.

gets. More specifically, the execution time of our proposed model BSCA is much faster than that of alternative online algorithms and the recent fast algorithms FOGD and NOGD. BCDA is only slightly slower than the fastest methods which, with the notable exception of BPAS, achieve significantly worse results.

In terms of accuracy, the BSCA method performs best on IJCNN, and more importantly, it performs consistently well across all problems. Only the rather slow online gradient descent approach achieves similar performance, but using 8 to 33 times as much training time, which would allow BSCA to perform many epochs.

We excluded the SUSY and COVTYPE data sets from table 2 due to the effort of tuning the parameter D for the FOGD method. For example, an experiment using $D = 10{,}000$ for FOGD was running for approximately two hours, achieving a test accu-

Table 2: Training time (seconds) and testing error (%) comparison between BSCA algorithm and all approaches implemented by (Lu et al., 2016). The budget size *B* is fixed to 500 for consistency with the previous experiment. The number of random features *D* for FOGD is selected depending on the settings in (Lu et al., 2016), where *D* is set to 4000, 1600 and 1000 for ADULT, CODRNA and IJCNN, respectively. For better readability, column-wise best values are highlighted.

	ADULT		CODRNA		IJCNN	
Algorithm	training time	test accuracy	training time	test accuracy	training time	test accuracy
Perceptron	53.37	83.35%	60.04	88.12%	25.31	95.75%
OGD	107.02	**84.50%**	227.25	**95.77%**	92.90	94.59%
RBP	**7.21**	76.38%	6.42	95.31%	**8.04**	91.74%
Forgetron	9.31	83.39%	7.24	92.55%	9.06	92.74%
Projectron	38.80	25.65%	61.79	33.31%	11.98	96.24%
Projectron++	36.00	82.59%	46.89	71.95%	51.02	96.77%
BPAS	7.69	83.44%	6.76	94.12%	8.86	95.01%
BOGD	8.02	76.38%	6.59	80.77%	8.66	91.01%
BSGD	8.48	80.74%	**4.37**	89.85%	11.28	96.68%
FOGD	18.00	82.00%	9.17	72.20%	26.13	90.50%
NOGD	37.10	83.15%	38.17	95.62%	40.30	90.50%
BSCA	9.84	83.18%	6.85	95.68%	11.53	**97.27%**

racy of 49.98% (guessing performance), where BSCA achieved 77.03%. This observation further validates the effectiveness and efficiency of our proposed technique. Thus, we believe that our BSCA algorithm is a promising technique for building large kernel learning algorithms for large-scale classification tasks.

5.4 Discussion

Our results do not only indicate that the optimization behavior of BSGD and BSCA is significantly different, they also demonstrate the superiority of the dual approach for SVM training, in terms of optimization behavior as well as in terms of test accuracy. We attribute its success to the good fit of coordinate ascent to the box-constrained dual problem, while the primal solver effectively ignores the upper bound (which is not represented explicitly in the primal problem), resulting in large oscillations. Importantly, the improved optimization behavior translates directly into better learning behavior. The introduction of budget maintenance techniques into the dual solver does not change this overall picture, and hence yields a viable route for fast training of kernel machines.

The BSCA method does not only clearly outperform the primal BSGD method, but it also performs very well when compared with alternative fast training schemes for training kernelized SVMs. The fast epoch times indicate a low overhead over the margin computation, which is at the heart of every iterative method. At the same time our new approach achieves excellent accuracy.

6 CONCLUDING REMARKS

We have presented the first dual decomposition algorithm for support vector machine training honoring a budget, i.e., an upper bound on the number of support vectors. This approximate SVM training algorithm combines fast iterations enabled by the budget approach with the fast convergence of a dual decomposition algorithm. Like its primal cousin, it is fundamentally limited only by the approximation error induced by the budget. We demonstrate significant speed-ups over primal budgeted training, as well as increased stability. Overall, for training SVMs on a budget, we can clearly recommend our method as a plug-in replacement for primal methods. It is rather straightforward to extend our algorithm to other kernel machines with box-constrained dual problems.

ACKNOWLEDGEMENTS

We acknowledge support by the Deutsche Forschungsgemeinschaft (DFG) through grant GL 839/3-1.

REFERENCES

Bottou and Lin, 2006Bottou, L. and Lin, C.-J. (2006). Support vector machine solvers.

Byun and Lee, 2002Byun, H. and Lee, S.-W. (2002). *Applications of Support Vector Machines for Pattern Recognition: A Survey*, pages 213–236. Springer Berlin Heidelberg, Berlin, Heidelberg.

Calandriello et al., 2017Calandriello, D., Lazaric, A., and Valko, M. (2017). Efficient second-order online kernel learning with adaptive embedding. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 6140–6150. Curran Associates, Inc.

Cavallanti et al., 2007Cavallanti, G., Cesa-Bianchi, N., and Gentile, C. (2007). Tracking the best hyperplane with a simple budget perceptron. *Machine Learning*, 69(2):143–167.

Chang and Lin, 2011Chang, C.-C. and Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3).

Cortes and Vapnik, 1995Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

Dekel et al., 2008Dekel, O., Shalev-Shwartz, S., and Singer, Y. (2008). The forgetron: A kernel-based perceptron on a budget. *SIAM J. Comput.*, 37(5):1342–1372.

Dekel and Singer, 2007Dekel, O. and Singer, Y. (2007). Support vector machines on a budget. MIT Press.

Fan et al., 2008Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). Liblinear: A library for large linear classification. *J. Mach. Learn. Res.*, pages 1871–1874.

Freund and Schapire, 1998Freund, Y. and Schapire, R. E. (1998). Large margin classification using the perceptron algorithm. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT' 98, New York, NY, USA. ACM.

Glasmachers, 2016Glasmachers, T. (2016). Finite sum acceleration vs. adaptive learning rates for the training of kernel machines on a budget. In *NIPS workshop on Optimization for Machine Learning*.

Hsieh et al., 2008Hsieh, C.-J., Chang, K.-W., Lin, C.-J., Keerthi, S. S., and Sundararajan, S. (2008). A dual coordinate descent method for large-scale linear SVM. In *Proceedings of the 25th International Conference on Machine Learning*, pages 408–415. ACM.

Joachims, 2006Joachims, T. (2006). Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–226. ACM.

Kivinen et al., 2003Kivinen, J., Smola, A. J., and Williamson, R. C. (2003). Online learning with kernels. volume 52, pages 2165–2176.

Le et al., 2016Le, T., Nguyen, T., Nguyen, V., and Phung, D. (2016). Dual space gradient descent for online learning. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems 29*, pages 4583–4591. Curran Associates, Inc.

Lin, 2001Lin, C.-J. (2001). On the convergence of the decomposition method for support vector machines. *IEEE Transactions on Neural Networks*, 12(6):1288–1298.

List and Simon, 2005List, N. and Simon, H. U. (2005). General polynomial time decomposition algorithms. In *International Conference on Computational Learning Theory*, pages 308–322. Springer.

Lu et al., 2016Lu, J., Hoi, S. C., Wang, J., Zhao, P., and Liu, Z.-Y. (2016). Large scale online kernel learning. *Journal of Machine Learning Research*, 17(47):1–43.

Lu et al., 2018Lu, J., Sahoo, D., Zhao, P., and Hoi, S. C. H. (2018). Sparse passive-aggressive learning for bounded online kernel methods. *ACM Trans. Intell. Syst. Technol.*, 9(4).

Mohri et al., 2012Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2012). *Foundations of Machine Learning*. MIT press.

Nesterov, 2012Nesterov, Y. (2012). Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362.

Nguyen et al., 2017Nguyen, T. D., Le, T., Bui, H., and Phung, D. (2017). Large-scale online kernel learning with random feature reparameterization. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*.

Orabona et al., 2009Orabona, F., Keshet, J., and Caputo, B. (2009). Bounded kernel-based online learning. *J. Mach. Learn. Res.*, pages 2643–2666.

Osuna et al., 1997Osuna, E., Freund, R., and Girosi, F. (1997). An improved training algorithm of support vector machines. In *Neural Networks for Signal Processing VII*, pages 276 – 285.

Quinlan et al., 2003Quinlan, M. J., Chalup, S. K., and Middleton, R. H. (2003). Techniques for improving vision and locomotion on the Sony AIBO robot. In *In Proceedings of the 2003 Australasian Conference on Robotics and Automation*.

Rahimi and Recht, 2008Rahimi, A. and Recht, B. (2008). Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184.

Shalev-Shwartz et al., 2007Shalev-Shwartz, S., Singer, Y., and Srebro, N. (2007). Pegasos: Primal estimated subgradient solver for SVM. In *Proceedings of the 24th International Conference on Machine Learning*, pages 807–814.

Shigeo, 2005Shigeo, A. (2005). *Support Vector Machines for Pattern Classification (Advances in Pattern Recognition)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Son et al., 2010Son, Y.-J., Kim, H.-G., Kim, E.-H., Choi, S., and Lee, S.-K. (2010). Application of support vector machine for prediction of medication adherence in heart failure patients. *Healthcare Informatics Research*, pages 253–259.

Steinwart, 2003Steinwart, I. (2003). Sparseness of support vector machines. *Journal of Machine Learning Research*, 4:1071–1105.

Steinwart et al., 2011Steinwart, I., Hush, D., and Scovel, C. (2011). Training SVMs without offset. *Journal of Machine Learning Research*, 12(Jan):141–202.

Wang et al., 2012Wang, Z., Crammer, K., and Vucetic, S. (2012). Breaking the curse of kernelization: Budgeted stochastic gradient descent for large-scale SVM training. *J. Mach. Learn. Res.*, 13(1):3103–3131.

Wang and Vucetic, 2010Wang, Z. and Vucetic, S. (2010). Online passive-aggressive algorithms on a budget. In

Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, volume 9, pages 908–915. PMLR.

Wen et al., 2017Wen, Z., Shi, J., He, B., Li, Q., and Chen, J. (2017). ThunderSVM: A fast SVM library on GPUs and CPUs. *To appear in arxiv*.

Yang et al., 2012Yang, T., Li, Y.-F., Mahdavi, M., Jin, R., and Zhou, Z.-H. (2012). Nyström method vs. random fourier features: A theoretical and empirical comparison. In *Advances in neural information processing systems*, pages 476–484.

Zhao et al., 2012Zhao, P., Wang, J., Wu, P., Jin, R., and Hoi, S. C. H. (2012). Fast bounded online gradient descent algorithms for scalable kernel-based online learning. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, USA.