The Stochastic Gradient Descent for the Primal L1-SVM Optimization Revisited

Constantinos Panagiotakopoulos and Petroula Tsampouka

School of Technology, Aristotle University of Thessaloniki, Greece
costapan@eng.auth.gr, petroula@auth.gr

Abstract. We reconsider the stochastic (sub)gradient approach to the unconstrained primal L1-SVM optimization. We observe that if the learning rate is inversely proportional to the number of steps, i.e., the number of times any training pattern is presented to the algorithm, the update rule may be transformed into the one of the classical perceptron with margin in which the margin threshold increases linearly with the number of steps. Moreover, if we cycle repeatedly through the possibly randomly permuted training set the dual variables defined naturally via the expansion of the weight vector as a linear combination of the patterns on which margin errors were made are shown to obey at the end of each complete cycle automatically the box constraints arising in dual optimization. This renders the dual Lagrangian a running lower bound on the primal objective tending to it at the optimum and makes available an upper bound on the relative accuracy achieved which provides a meaningful stopping criterion. In addition, we propose a mechanism of presenting the same pattern repeatedly to the algorithm which maintains the above properties. Finally, we give experimental evidence that algorithms constructed along these lines exhibit a considerably improved performance.

1 Introduction

Support Vector Machines (SVMs) [1,20,5] have been extensively used as linear classifiers either in the space where the patterns originally reside or in high dimensional feature spaces induced by kernels. They appear to be very successful at addressing the classification problem expressed as the minimization of an objective function involving the empirical risk while at the same time keeping low the complexity of the classifier. As measures of the empirical risk various quantities have been proposed with the 1- and 2-norm loss functions being the most widely accepted ones giving rise to the optimization problems known as L1- and L2-SVMs [4]. SVMs typically treat the problem as a constrained quadratic optimization in the dual space. At the early stages of SVM development their efficient implementation was hindered by the quadratic dependence of their memory requirements on the number of training examples a fact which rendered prohibitive the processing of large datasets. The idea of applying optimization only to a subset of the training set in order to overcome this difficulty resulted in the development of decomposition methods [16,9]. Although such methods led

to improved convergence rates, in practice their superlinear dependence on the number of examples, which can be even cubic, can still lead to excessive runtimes when dealing with massive datasets. Recently, the so-called linear SVMs [10,7,8,13] taking advantage of linear kernels in order to allow parts of them to be written in primal notation succeeded in outperforming decomposition SVMs.

The above considerations motivated research in alternative algorithms naturally formulated in primal space long before the advent of linear SVMs mostly in connection with the large margin classification of linearly separable datasets a problem directly related to the L2-SVM. Indeed, in the case that the 2-norm loss takes the place of the empirical risk an equivalent formulation exists which renders the dataset linearly separable in a high dimensional feature space. Such alternative algorithms ([14,15] and references therein) are mostly based on the perceptron [17], the simplest online learning algorithm for binary linear classification, with their key characteristic being that they work in the primal space in an online manner, i.e., processing one example at a time. Cycling repeatedly through the patterns they update their internal state stored in the weight vector each time an appropriate condition is satisfied. This way, due to their ability to process one example at a time, such algorithms succeed in sparing time and memory resources and consequently become able to handle large datasets.

Since the L1-SVM problem is not known to admit an equivalent maximum margin interpretation via a mapping to an appropriate space fully primal large margin perceptron-like algorithms appear unable to deal with such a task.[1] Nevertheless, a somewhat different approach giving rise to online algorithms was developed which focuses on the minimization of the regularized 1-norm soft margin loss through stochastic gradient descent (SGD). Notable representatives of this approach are the pioneer NORMA [12] (see also [21]) and Pegasos [18,19] (see also [2,3]). SGD gives rise to a kind of perceptron-like update having as an important ingredient the "shrinking" of the current weight vector. Shrinking always takes place when a pattern is presented to the algorithm with it being the only modification suffered by the weight vector if no loss is incurred. Thus, due to lack of a meaningful stopping criterion the algorithm without user intervention keeps running forever. In that sense the algorithms in question are fundamentally different from the mistake-driven large margin perceptron-like classifiers which terminate after a finite number of updates. There is no proof even for their asymptotic convergence when they use as output the final hypothesis but they do exist probabilistic convergence results or results in terms of the average hypothesis.

In the present work we reconsider the straightforward version of SGD for the primal unconstrained L1-SVM problem assuming a learning rate inversely proportional to the number of steps. Therefore, such an algorithm can be regarded

[1] The Margin Perceptron with Unlearning (MPU) [13] addresses the L1-SVM problem by keeping track of the number of updates caused by each pattern in parallel with the weight vector which is updated according to a perceptron-like rule. In that sense MPU uses dual variables and should rather be considered a linear SVM which, however, possesses a finite time bound for achieving a predefined relative accuracy.

either as NORMA with a specific dependence of the learning rate on the number of steps or as Pegasos with no projection step in the update and with a single example contributing to the (sub)gradient ($k = 1$). We observe here that this algorithm may be transformed into a classical perceptron with margin [6] in which the margin threshold increases linearly with the number of steps. The obvious gain from this observation is that the shrinking of the weight vector at each step amounts to nothing but an increase of the step counter by one unit instead of the costly multiplication of all the components of the generally non-sparse weight vector with a scalar. Another benefit arising from the above simplified description is that we are able to demonstrate easily that if we cycle through the data in complete epochs the dual variables defined naturally via the expansion of the weight vector as a linear combination of the patterns on which margin errors were made satisfy automatically the box constraints of the dual optimization. An important consequence of this unexpected result is that the relevant dual Lagrangian which is expressed in terms of the total number of margin errors, the number of complete epochs and the length of the current weight vector provides during the run a lower bound on the primal objective function and gives us a measure of the progress made in the optimization process. Indeed, by virtue of the strong duality theorem the dual Lagrangian and the primal objective coincide at optimality. Therefore, assuming convergence to the optimum an upper bound on the relative accuracy involving the dual Lagrangian may be defined which offers a useful and practically achievable stopping criterion. Moreover, we may now provide evidence in favor of the asymptotic convergence to the optimum by testing experimentally the vanishing of the duality gap. Finally, aiming at performing more updates at the expense of only one costly inner product calculation we propose a mechanism of presenting the same pattern repeatedly to the algorithm consistently with the above interesting properties.

The paper is organized as follows. Section 2 describes the algorithm and its properties. In Section 3 we give implementational details and deliver our experimental results. Finally, Section 4 contains our conclusions.

2 The Algorithm and its Properties

Assume we are given a training set $\{(\boldsymbol{x}_k, l_k)\}_{k=1}^m$, with vectors $\boldsymbol{x}_k \in \mathbb{R}^d$ and labels $l_k \in \{+1, -1\}$. This set may be either the original dataset or the result of a mapping into a feature space of higher dimensionality [20,5]. By placing $\boldsymbol{x}_k$ in the same position at a distance ρ in an additional dimension, i.e., by extending $\boldsymbol{x}_k$ to $[\boldsymbol{x}_k, \rho]$, we construct an embedding of our data into the so-called augmented space [6]. The advantage of this embedding is that the linear hypothesis in the augmented space becomes homogeneous. Following the augmentation, a reflection with respect to the origin of the negatively labeled patterns is performed allowing for a uniform treatment of both categories of patterns. We define $R \equiv \max_k \|\boldsymbol{y}_k\|$ with $\boldsymbol{y}_k \equiv [l_k \boldsymbol{x}_k, l_k \rho]$ the k-th augmented and reflected pattern.

Let us consider the regularized empirical risk

$$\frac{\lambda}{2}\left\|\boldsymbol{w}\right\|^2+\frac{1}{m}\sum_{k=1}^{m}\max\{0,1-\boldsymbol{w}\cdot\boldsymbol{y}_k\}$$

involving the 1-norm soft margin loss $\max\{0,1-\boldsymbol{w}\cdot\boldsymbol{y}_k\}$ for the pattern $\boldsymbol{y}_k$ and the regularization parameter $\lambda>0$ controlling the complexity of the classifier $\boldsymbol{w}$. For a given dataset of size m minimization of the regularized empirical risk with respect to $\boldsymbol{w}$ is equivalent to the minimization of the objective function

$$\mathcal{J}(\boldsymbol{w},C)\equiv\frac{1}{2}\left\|\boldsymbol{w}\right\|^2+C\sum_{k=1}^{m}\max\{0,1-\boldsymbol{w}\cdot\boldsymbol{y}_k\}\ ,$$

where the "penalty" parameter $C>0$ is related to λ as

$$C=\frac{1}{\lambda m}\ .$$

This is the L1-SVM problem expressed as an unconstrained optimization.

The algorithms we are concerned with are classical SGD algorithms. The term stochastic refers to the fact that they perform gradient descent with respect to the objective function in which the empirical risk $(1/m)\sum_{k=1}^{m}\max\{0,1-\boldsymbol{w}\cdot\boldsymbol{y}_k\}$ is approximated by the instantaneous risk $\max\{0,1-\boldsymbol{w}\cdot\boldsymbol{y}_k\}$ on a single example. The general form of the update rule is then

$$\boldsymbol{w}_{t+1}=\boldsymbol{w}_t-\eta_t\nabla_{\boldsymbol{w}_t}\left[\frac{1}{2}\left\|\boldsymbol{w}_t\right\|^2+\frac{1}{\lambda}\max\{0,1-\boldsymbol{w}_t\cdot\boldsymbol{y}_k\}\right]\ ,$$

where η_t is the learning rate and $\nabla_{\boldsymbol{w}_t}$ stands for a subgradient with respect to $\boldsymbol{w}_t$ since the 1-norm soft margin loss is only piecewise differentiable ($t\geq 0$). We choose a learning rate $\eta_t=1/(t+1)$ which satisfies the conditions $\sum_{t=0}^{\infty}\eta_t^2<\infty$ and $\sum_{t=0}^{\infty}\eta_t=\infty$ usually imposed in the convergence analysis of stochastic approximations. Then, noticing that $\boldsymbol{w}_t-\frac{1}{t+1}\boldsymbol{w}_t=\frac{t}{t+1}\boldsymbol{w}_t$, we obtain the update

$$\boldsymbol{w}_{t+1}=\frac{t}{t+1}\boldsymbol{w}_t+\frac{1}{\lambda(t+1)}\boldsymbol{y}_k \tag{1}$$

whenever

$$\boldsymbol{w}_t\cdot\boldsymbol{y}_k\leq 1 \tag{2}$$

and

$$\boldsymbol{w}_{t+1}=\frac{t}{t+1}\boldsymbol{w}_t \tag{3}$$

otherwise. In deriving the above update rule we made the choice $\boldsymbol{w}_t-\lambda^{-1}\boldsymbol{y}_k$ for the subgradient at the point $\boldsymbol{w}_t\cdot\boldsymbol{y}_k=1$ where the 1-norm soft margin loss is not differentiable. We assume that $\boldsymbol{w}_0=\boldsymbol{0}$. We see that if $\boldsymbol{w}_t\cdot\boldsymbol{y}_k>1$ the update consists of a pure shrinking of the current weight vector by the factor $t/(t+1)$.

The update rule may be simplified considerably if we perform the change of variable

$$\boldsymbol{w}_t = \frac{\boldsymbol{a}_t}{\lambda t} \tag{4}$$

for $t > 0$ and $\boldsymbol{w}_0 = \boldsymbol{a}_0 = \mathbf{0}$ for $t = 0$. In terms of the new weight vector $\boldsymbol{a}_t$ the update rule becomes

$$\boldsymbol{a}_{t+1} = \boldsymbol{a}_t + \boldsymbol{y}_k \tag{5}$$

whenever

$$\boldsymbol{a}_t \cdot \boldsymbol{y}_k \leq \lambda t \tag{6}$$

and

$$\boldsymbol{a}_{t+1} = \boldsymbol{a}_t \tag{7}$$

otherwise.[2] This is the update of the classical perceptron algorithm with margin in which, however, the margin threshold in condition (6) increases linearly with the number of presentations of patterns to the algorithm independent of whether they lead to a change in the weight vector $\boldsymbol{a}_t$. Thus, t counts the number of times any pattern is presented to the algorithm which corresponds to the number of updates (including the pure shrinkings (3)) of the weight vector $\boldsymbol{w}_t$. Instead, the weight vector $\boldsymbol{a}_t$ is updated only if (6) is satisfied meaning that a margin error is made on $\boldsymbol{y}_k$.

In the original formulation of Pegasos [18] the update is completed with a projection step in order to enforce the bound $\|\boldsymbol{w}_t\| \leq 1/\sqrt{\lambda}$ which holds for the optimal solution. We show now that this is dynamically achieved to any desired accuracy after the elapse of sufficient time. In practice, however, it is in almost all cases achieved after one pass over the data.

Proposition 1. *For $t > 0$ the norm of the weight vector $\boldsymbol{w}_t$ is bounded from above as follows*

$$\|\boldsymbol{w}_t\| \leq \frac{1}{\sqrt{\lambda}} \sqrt{1 + \left(\frac{R^2}{\lambda} - 1\right) \frac{1}{t}} \ . \tag{8}$$

Proof. From the update rule (5) taking into account condition (6) under which the update takes place we get

$$\|\boldsymbol{a}_{t+1}\|^2 - \|\boldsymbol{a}_t\|^2 = \|\boldsymbol{y}_k\|^2 + 2\boldsymbol{a}_t \cdot \boldsymbol{y}_k \leq R^2 + 2\lambda t \ .$$

Obviously, this is trivially satisfied if (6) is violated and (7) holds. A repeated application of the above inequality with $\boldsymbol{a}_0 = \mathbf{0}$ gives

$$\|\boldsymbol{a}_t\|^2 \leq R^2 t + 2\lambda \sum_{k=0}^{t-1} k = R^2 t + \lambda t(t-1) = (R^2 - \lambda)t + \lambda t^2$$

from where using (4) and taking the square root we obtain (8). □

[2] For $t = 0$ (6) becomes $\boldsymbol{a}_0 \cdot \boldsymbol{y}_k \leq 0$ instead of $\boldsymbol{a}_0 \cdot \boldsymbol{y}_k \leq 1$ which is obtained from (2) with $\boldsymbol{w}_0 = \boldsymbol{a}_0$. Since both are satisfied with $\boldsymbol{a}_0 = \mathbf{0}$ (6) may be used for all t.

The Stochastic Gradient Descent Algorithm with random selection of examples

Input: A dataset $S = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k, \ldots, \boldsymbol{y}_m)$ with augmentation and reflection assumed
Fix: C, $t_{\max}$
Define: $\lambda = 1/(Cm)$
Initialize: $t = 0$, $\boldsymbol{a}_0 = \boldsymbol{0}$
while $t < t_{\max}$ **do**
  Choose $\boldsymbol{y}_k$ from S randomly
  if $\boldsymbol{a}_t \cdot \boldsymbol{y}_k \leq \lambda t$ **then**
    $\boldsymbol{a}_{t+1} = \boldsymbol{a}_t + \boldsymbol{y}_k$
  else
    $\boldsymbol{a}_{t+1} = \boldsymbol{a}_t$
  $t \leftarrow t + 1$
$\boldsymbol{w}_t = \boldsymbol{a}_t/(\lambda t)$

Combining (8) with the initial choice $\boldsymbol{w}_0 = \boldsymbol{0}$ we see that for all t the weaker bound $\|\boldsymbol{w}_t\| \leq R/\lambda$ previously derived in [19] holds.

SGD gives naturally rise to online algorithms. Therefore, we may choose the examples to be presented to the algorithm at random. However, the L1-SVM optimization task is a batch learning problem which may be better tackled by online algorithms via the classical conversion of such algorithms to the batch setting. This is done by cycling repeatedly through the possibly randomly permuted training dataset and using the last hypothesis for prediction. This traditional procedure of presenting the training data to the algorithm in complete epochs has in our case, as we will see shortly, the additional advantage that there exists a lower bound on the optimal value of the objective function to be minimized which is expressed in terms of quantities available during the run. The existence of such a lower bound provides an estimate of the relative accuracy achieved by the algorithm.

Proposition 2. *Let us assume that at some stage the whole training set has been presented to the algorithm exactly T times. Then, it holds that*

$$\mathcal{J}_{\text{opt}}(C) \equiv \min_{\boldsymbol{w}} \mathcal{J}(\boldsymbol{w}, C) \geq \mathcal{L}^T \equiv C\frac{M}{T} - \frac{1}{2}\left\|\boldsymbol{w}^T\right\|^2 \quad , \tag{9}$$

where M is the total number of margin errors made up to that stage and $\boldsymbol{w}^T \equiv \boldsymbol{w}_{(mT)}$ the weight vector at $t = mT$ with m being the size of the training set.

Proof. Let I_k^t denote the number of margin errors made on the pattern $\boldsymbol{y}_k$ up to time t such that $\boldsymbol{a}_t = \sum_k I_k^t \boldsymbol{y}_k$. Obviously, it holds that

$$0 \leq I_k^{(mT)} \leq T \tag{10}$$

since $\boldsymbol{y}_k$ up to time $t = mT$ has been presented to the algorithm exactly T times. Then, taking into account (4) we see that at time t the dual variable α_k^t associated with $\boldsymbol{y}_k$ is $\alpha_k^t = I_k^t/(\lambda t)$ and consequently the dual variable $\alpha_k^{(mT)}$ after T complete epochs is given by

$$\alpha_k^{(mT)} = \frac{I_k^{(mT)}}{\lambda mT} = C\frac{I_k^{(mT)}}{T} \quad . \tag{11}$$

With use of (10) we readily conclude that the dual variables after T complete epochs automatically satisfy the box constraints

$$0 \leq \alpha_k^{(mT)} \leq C \quad . \tag{12}$$

From the weak duality theorem it follows that

$$\mathcal{J}(\boldsymbol{w}, C) \geq \mathcal{L}(\boldsymbol{\alpha}) = \sum_k \alpha_k - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j \boldsymbol{y}_i \cdot \boldsymbol{y}_j \ ,$$

where $\mathcal{L}(\boldsymbol{\alpha})$ is the dual Lagrangian[3] and the variables α_k obey the box constraints $0 \leq \alpha_k \leq C$. Thus, setting $\alpha_k = \alpha_k^{(mT)}$ in the above inequality, noticing that $\sum_k \alpha_k^{(mT)} = (C/T) \sum_k I_k^{(mT)} = CM/T$ and substituting $\sum_k \alpha_k^{(mT)} \boldsymbol{y}_k$ with $\boldsymbol{w}^T$ we obtain $\mathcal{J}(\boldsymbol{w}, C) \geq \mathcal{L}^T$ which is equivalent to (9). □

In the course of proving Proposition 2 we saw that although the algorithm is fully primal the dual variables α_k^t defined through the expansion $\boldsymbol{w}_t = \sum_k \alpha_k^t \boldsymbol{y}_k$ of the weight vector $\boldsymbol{w}_t$ as a linear combination of the patterns on which margin errors were made obey after T complete epochs automatically the box constraints (12) encountered in dual optimization.[4] This surprising result allows us to construct the dual Lagrangian $\mathcal{L}^T$ which provides a lower bound on the optimal value $\mathcal{J}_{\text{opt}}$ of the objective $\mathcal{J}$ and assuming $\mathcal{L}^T > 0$ to obtain an upper bound $\mathcal{J}/\mathcal{L}^T - 1$ on the relative accuracy $\mathcal{J}/\mathcal{J}_{\text{opt}} - 1$ achieved as the algorithm keeps running. Thus, we have for the first time a primal SGD algorithm which may use the relative accuracy as stopping criterion.[5] It is also worth noticing that $\mathcal{L}^T$ involves only the total number M of margin errors and does not require that we keep the values of the individual dual variables during the run.

Although the automatic satisfaction of the box constraints by the dual variables is very important it is by no means sufficient to ensure vanishing of the duality gap and consequently convergence to the optimal solution. To demonstrate convergence to the optimum relying on dual optimization theory we must make sure that the Karush-Kuhn-Tucker (KKT) conditions [20,5] are satisfied. Their approximate satisfaction demands that the only patterns which have a substantial loss be the ones which have dual variables equal or at least extremely close to C (bound support vectors) and moreover that the patterns which have zero loss and margin considerably larger than $1/\left\|\boldsymbol{w}^T\right\|$ should have vanishingly small dual variables. Patterns with margin very close to $1/\left\|\boldsymbol{w}^T\right\|$ may

[3] Maximization of $\mathcal{L}(\boldsymbol{\alpha})$ subject to the constraints $0 \leq \alpha_k \leq C$ is the dual of the primal L1-SVM problem expressed as a constrained minimization.

[4] We expect that the dual variables will also satisfy the box constraints in the limit $t \to \infty$ if the patterns presented to the algorithm are selected randomly with equal probability since asymptotically they will all be selected an equal number of times.

[5] It is, of course, computationally expensive to evaluate at the end of each epoch the exact primal objective. Thus, an approximate calculation of the loss using the value that the weight vector had the last time each pattern was presented to the algorithm is preferable. This way we exploit the already computed inner product $\boldsymbol{a}_t \cdot \boldsymbol{y}_k$ which is needed in order to decide whether condition (6) is satisfied. If this approximate calculation gives a value of the relative accuracy which is not larger than f times the one set as stopping criterion we proceed to a proper calculation of the primal objective. The comparison coefficient f is given empirically a value close to 1.

The Stochastic Gradient Descent Algorithm with relative accuracy ϵ

Input: A dataset $S = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k, \ldots, \boldsymbol{y}_m)$ with augmentation and reflection assumed
Fix: $C,\ \epsilon,\ f,\ T_{\max}$
Define: $q_k = \|\boldsymbol{y}_k\|^2,\ \ \lambda = 1/(Cm),\ \ \epsilon' = f\epsilon$
Initialize: $t = 0,\ T = 0,\ M = 0,\ r_0 = 0,\ \boldsymbol{a}_0 = \boldsymbol{0}$
while $T < T_{\max}$ **do**
  Permute(S)
  $L = 0$
  for $k = 1$ **to** m **do**
    $p_{tk} = \boldsymbol{a}_t \cdot \boldsymbol{y}_k$
    $\theta_t = \lambda t$
    if $p_{tk} \leq \theta_t$ **then**
      $\boldsymbol{a}_{t+1} = \boldsymbol{a}_t + \boldsymbol{y}_k$
      $r_{t+1} = r_t + 2p_{tk} + q_k$
      $M \leftarrow M + 1$
      if $t > 0$ **then**
        $L \leftarrow L + 1 - p_{tk}/\theta_t$
      else
        $L \leftarrow L + 1$
    else
      $\boldsymbol{a}_{t+1} = \boldsymbol{a}_t$
      $r_{t+1} = r_t$
    $t \leftarrow t + 1$
  $T \leftarrow T + 1$
  $\theta = \lambda t$
  $w2 = r_t/(2\theta^2)$
  $\mathcal{J} = w2 + CL$
  $\mathcal{L} = CM/T - w2$
  if $\mathcal{J} - \mathcal{L} \leq \epsilon'\mathcal{L}$ **then**
    $L = 0$
    for $k = 1$ **to** m **do**
      $p_k = \boldsymbol{a}_t \cdot \boldsymbol{y}_k$
      if $p_k < \theta$ **then**
        $L \leftarrow L + \theta - p_k$
    $L \leftarrow L/\theta$
    $\mathcal{J} = w2 + CL$
    if $\mathcal{J} - \mathcal{L} \leq \epsilon\mathcal{L}$ **then**
      break
$\boldsymbol{w}_t = \boldsymbol{a}_t/(\lambda t)$

have dual variables with values between 0 and C and play the role of the non-bound support vectors. From (11) we see that the dual variable associated with the k-th pattern is equal to CT_k/T where $T_k \equiv I_k^{(mT)}$ is the number of epochs for which the k-th pattern was found to be a margin error. It is apparent that if there exists a number of epochs no matter how large it may be after which a pattern is consistently found to be a margin error then in the limit $T \to \infty$ we will have $(T_k/T) \to 1$ and the dual variable associated with it will asymptotically approach C. In contrast, if a pattern after a specific number of epochs is never found to be a margin error then $(T_k/T) \to 0$ and its dual variable will tend asymptotically to zero reflecting the accumulated effect of the shrinking that the weight vector suffers each time a pattern is presented to the algorithm. Therefore, the algorithm has the necessary ingredients for asymptotic satisfaction of the KKT conditions for the vanishing of the duality gap. The potential danger remains, however, that they may exist patterns with margin not very close to $1/\left\|\boldsymbol{w}^T\right\|$ which do not belong to any of the above categories and occasionally either become margin errors although most of the time are not or become classified with sufficiently large margin despite of the fact that they are most of the time margin errors. The hope is that with time the changes in the weight vector $\boldsymbol{w}_t$ will become smaller and smaller and such events will become more and more rare leading eventually to convergence to the optimal solution.

The above discussion cannot be regarded as a formal proof of the asymptotic convergence of the algorithm. We believe, however, that it does provide a convincing argument that assuming convergence (not necessarily to the opti-

mum) the duality gap will eventually tend to zero and the lower bound $\mathcal{L}^T$ on the primal objective $\mathcal{J}$ given in Proposition 2 will approach the optimal primal objective $\mathcal{J}_{\rm opt}$, thereby proving that convergence to the optimum has been achieved. If, instead, we make the stronger assumption of convergence to the optimum then, of course, the vanishing of the duality gap follows from the strong duality theorem. In any case the stopping criterion exploiting the upper bound $\mathcal{J}/\mathcal{L}^T - 1$ on the relative accuracy $\mathcal{J}/\mathcal{J}_{\rm opt} - 1$ is a meaningful one.

Our discussion so far assumes that in an epoch each pattern is presented only once to the algorithm. We may, however, consider the option of presenting the same pattern $\boldsymbol{y}_k$ repeatedly ℓ times to the algorithm[6] aiming at performing more updates at the expense of only one calculation of the costly inner product $\boldsymbol{a}_t \cdot \boldsymbol{y}_k$. Proposition 2 and the analysis following it will still be valid on the condition that all patterns in each epoch are presented exactly the same number ℓ of times to the algorithm. Then, such an epoch should be regarded as equivalent to ℓ usual epochs with single presentations of patterns to the algorithm and will have as a result the increase of t by an amount equal to $m\ell$.

It is, of course, important to be able to decide in terms of just the initial value of $\boldsymbol{a}_t \cdot \boldsymbol{y}_k$ how many, let us say ℓ_+, out of these ℓ consecutive presentations of the pattern $\boldsymbol{y}_k$ to the algorithm will lead to a margin error, i.e., to an update of $\boldsymbol{a}_t$, with each of the remaining $\ell_- = \ell - \ell_+$ presentations necessarily corresponding to just an increase of t by 1 which amounts to a pure shrinking of $\boldsymbol{w}_t$.

Proposition 3. *Let the pattern $\boldsymbol{y}_k$ be presented at time t repeatedly ℓ times to the algorithm. Also let*

$$P = \boldsymbol{a}_t \cdot \boldsymbol{y}_k - \lambda t \ .$$

Then, the number ℓ_+ of times that $\boldsymbol{y}_k$ will be found to be a margin error is given by the following formula

$$\begin{aligned} &\text{if } \ P > (\ell-1)\lambda \qquad \ell_+ = 0 \ , \\ &\text{if } \ P \le (\ell-1)\lambda \qquad \ell_+ = \min\left\{\ell, \left[\frac{(\ell-1)\lambda - P}{\max\{\|\boldsymbol{y}_k\|^2, \lambda\}}\right] + 1\right\} \ . \end{aligned} \tag{13}$$

Here $[x]$ denotes the integer part of $x \ge 0$.

Proof. For the sake of brevity we call a plus-step a presentation of the pattern $\boldsymbol{y}_k$ to the algorithm which leads to a margin error and a minus-step a presentation which does not. If at time t a plus-step takes place $\boldsymbol{a}_{t+1} \cdot \boldsymbol{y}_k - \lambda(t+1) = (\boldsymbol{a}_t \cdot \boldsymbol{y}_k - \lambda t) + (\|\boldsymbol{y}_k\|^2 - \lambda)$ while if a minus-step takes place $\boldsymbol{a}_{t+1} \cdot \boldsymbol{y}_k - \lambda(t+1) = (\boldsymbol{a}_t \cdot \boldsymbol{y}_k - \lambda t) - \lambda$. Thus, a plus-step adds to P the quantity $\|\boldsymbol{y}_k\|^2 - \lambda$ while a minus-step the quantity $-\lambda$. Clearly, after ℓ consecutive presentations of $\boldsymbol{y}_k$ to the algorithm it holds that $\boldsymbol{a}_{t+\ell} \cdot \boldsymbol{y}_k - \lambda(t+\ell) = P + \ell_+(\|\boldsymbol{y}_k\|^2 - \lambda) - (\ell - \ell_+)\lambda$.

[6] Multiple updates were introduced in [13,14]. A discussion in a context related to the present work is given in [11]. However, a proper SGD treatment in the presence of a regularization term for the 1-norm soft margin loss was not provided. Instead, a "forward-backward splitting" approach was adopted in which a multiple update in the absence of the regularizer is followed by ℓ pure regularizer-induced $\boldsymbol{w}_t$ shrinkings.

If $P > (\ell-1)\lambda$ it follows that $P-(\ell-1)\lambda > 0$ which means that after $\ell-1$ consecutive minus-steps condition (6) is still violated and an additional minus-step must take place. Thus, $\ell_- = \ell$ and $\ell_+ = 0$.

For $P \leq (\ell-1)\lambda$ we first treat the subcase $\max\{\|\boldsymbol{y}_k\|^2, \lambda\} = \lambda$. If $\|\boldsymbol{y}_k\|^2 \leq \lambda$ and $P \leq 0$ condition (6) is initially satisfied and will still be satisfied after any number of plus-steps since the quantity $\|\boldsymbol{y}_k\|^2 - \lambda$ that is added to P with a plus-step is non-positive. Thus, $\ell_+ = \ell$. This is in accordance with (13) since $((\ell-1)\lambda - P)/\lambda \geq \ell-1$ or $[((\ell-1)\lambda-P)/\lambda]+1 \geq \ell$ leading to $\ell_+ = \ell$. It remains for $\|\boldsymbol{y}_k\|^2 \leq \lambda$ to consider P in the interval $0 < P \leq (\ell-1)\lambda$ which can be further subdivided as $(\ell_1-1)\lambda < P \leq \ell_1\lambda$ with the integer ℓ_1 satisfying $1 \leq \ell_1 \leq \ell-1$. For P belonging to such a subinterval condition (6) is initially violated and will still be violated after ℓ_1-1 minus-steps while after one more minus-step will be satisfied. It will still be satisfied after any number of additional plus-steps because the quantity $\|\boldsymbol{y}_k\|^2 - \lambda$ that is added to P with a plus-step is non-positive. Thus, $\ell_- = \ell_1$ and $\ell_+ = \ell-\ell_1$. This is in accordance with (13) since $(\ell-\ell_1-1)\lambda \leq (\ell-1)\lambda - P < (\ell-\ell_1)\lambda$ leads to $[((\ell-1)\lambda-P)/\lambda]+1 = \ell-\ell_1$.

The subcase $\|\boldsymbol{y}_k\|^2 > \lambda$ of the case $P \leq (\ell-1)\lambda$ is far more complicated. If $\|\boldsymbol{y}_k\|^2 > \lambda$ with $P \leq -(\ell-1)(\|\boldsymbol{y}_k\|^2 - \lambda)$ condition (6) is initially satisfied and will still be satisfied after $\ell-1$ plus-steps since $P+(\ell-1)(\|\boldsymbol{y}_k\|^2-\lambda) \leq 0$. Thus, $\ell_+ = \ell$. This is consistent with (13) because $(\ell-1)\lambda - P \geq (\ell-1)\|\boldsymbol{y}_k\|^2$ or $[((\ell-1)\lambda-P)/\|\boldsymbol{y}_k\|^2]+1 \geq \ell$ leading to $\ell_+ = \ell$. It remains to be examined the case $\|\boldsymbol{y}_k\|^2 > \lambda$ with P in the interval $-(\ell-1)(\|\boldsymbol{y}_k\|^2-\lambda) < P \leq (\ell-1)\lambda$. The above interval can be expressed as a union of subintervals $(\ell-\ell_1-1)\lambda - \ell_1(\|\boldsymbol{y}_k\|^2-\lambda) < P \leq (\ell-\ell_1)\lambda - (\ell_1-1)(\|\boldsymbol{y}_k\|^2-\lambda)$ with the integer ℓ_1 satisfying $1 \leq \ell_1 \leq \ell-1$. Let P belong to such a subinterval. Let us also assume that the pattern $\boldsymbol{y}_k$ has been presented $\kappa \leq \ell$ consecutive times to the algorithm as a result of which κ_+ plus-steps and κ_- minus-steps have taken place and the quantity $\kappa_+(\|\boldsymbol{y}_k\|^2-\lambda) - \kappa_-\lambda$ has been added to P. Then $P_{\kappa_+,\kappa_-} \equiv P + \kappa_+(\|\boldsymbol{y}_k\|^2-\lambda) - \kappa_-\lambda$ satisfies $(\ell-\ell_1-1-\kappa_-)\lambda - (\ell_1-\kappa_+)(\|\boldsymbol{y}_k\|^2-\lambda) < P_{\kappa_+,\kappa_-} \leq (\ell-\ell_1-\kappa_-)\lambda - (\ell_1-1-\kappa_+)(\|\boldsymbol{y}_k\|^2-\lambda)$. As κ increases either κ_+ will first reach the value ℓ_1 with $\kappa_- < \ell-\ell_1$ or κ_- will first reach the value $\ell-\ell_1$ with $\kappa_+ < \ell_1$. In the former case $0 \leq (\ell-\ell_1-1-\kappa_-)\lambda < P_{\kappa_+,\kappa_-}$. This means that condition (6) is violated and will continue being violated until the number of minus-steps becomes equal to $\ell-\ell_1-1$ in which case one more minus-step must take place. Thus, all steps taking place after κ_+ has reached the value ℓ_1 are minus-steps. In the latter case $P_{\kappa_+,\kappa_-} \leq -(\ell_1-1-\kappa_+)(\|\boldsymbol{y}_k\|^2-\lambda) \leq 0$. This means that condition (6) is satisfied and will continue being satisfied until the number of plus-steps becomes equal to ℓ_1-1 in which case one more plus-step must take place. Thus, all steps taking place after κ_- has reached the value $\ell-\ell_1$ are plus-steps. In both cases $\ell_+ = \ell_1$. This is again in accordance with (13) because $(\ell_1-1)\|\boldsymbol{y}_k\|^2 \leq (\ell-1)\lambda - P < \ell_1\|\boldsymbol{y}_k\|^2$ or $[((\ell-1)\lambda-P)/\|\boldsymbol{y}_k\|^2]+1 = \ell_1$. □

With ℓ_+ given in Proposition 3 the update of multiplicity ℓ of the weight vector $\boldsymbol{a}_t$ is written formally as

$$\boldsymbol{a}_{t+\ell} = \boldsymbol{a}_t + \ell_+ \boldsymbol{y}_k \ . \tag{14}$$

3 Implementation and Experiments

We implement three types of SGD algorithms[7] along the lines of the previous section. The first is the plain algorithm with random selection of examples, denoted SGD-r, which terminates when the maximum number $t_{\max}$ of steps is reached. Its pseudocode is given in Section 2. The dual variables in this case do not satisfy the box constraints as a result of which relative accuracy cannot be used as stopping criterion. The SGD algorithm with relative accuracy ϵ, the pseudocode of which is also given in Section 2, is denoted SGD-s where s designates that in an epoch each pattern is presented a single time to the algorithm. It terminates when the relative deviation of the primal objective $\mathcal{J}$ from the dual Lagrangian $\mathcal{L}^T$ just falls below ϵ provided the maximum number $T_{\max}$ of full epochs is not exhausted. A variation of this algorithm, denoted SGD-m, replaces in the T-th epoch the usual update with the multiple update (14) of multiplicity $\ell = 5$ only if $0 < T \bmod 9 < 5$. For both SGD-s and SGD-m the comparison coefficient takes the value $f = 1.2$ unless otherwise explicitly stated.

Algorithms performing SGD on the primal objective are expected to perform better if linear kernels are employed. Therefore the feature space in our experiments will be chosen to be the original instance space. As a consequence, our algorithms should most naturally be compared with linear SVMs. Among them we choose SVM$^{\text{perf}}$ [8] [10], the first cutting-plane algorithm for training linear SVMs, the Optimized Cutting Plane Algorithm for SVMs[9] (OCAS) [7], the Dual Coordinate Descent[10] (DCD) algorithm [8] and the Margin Perceptron with Unlearning[11] (MPU) [13]. We also include in our study Pegasos[12] ($k = 1$). Finally, we briefly considered the SvmSgd[13] [3] and SGD-QN[14] [2] algorithms implemented in single precision.

The datasets we used for training are the binary Adult and Web datasets as compiled by Platt,[15] the training set of the KDD04 Physics dataset[16] (with 70 attributes after removing the 8 columns containing missing features), the Real-sim, News20 and Webspam (unigram treatment) datasets,[17] the multiclass Covertype UCI dataset[18] and the full Reuters RCV1 dataset.[19] Their number of instances and attributes are listed in Table 1. In the case of the Covertype dataset

[7] Sources available at `http://users.auth.gr/costapan`
[8] Source (version 2.50) available at `http://svmlight.joachims.org`
[9] Source (version 0.96) available at `http://cmp.felk.cvut.cz/~xfrancv/ocas/html`
[10] Source available at `http://www.csie.ntu.edu.tw/~cjlin/liblinear`. We used the slightly faster older liblinear version 1.7 instead of the latest 1.93.
[11] Source available at `http://users.auth.gr/costapan`
[12] Source available at `http://ttic.uchicago.edu/~shai/code`
[13] Source (version 2) available at `http://leon.bottou.org/projects/sgd`
[14] Source available at `https://www.hds.utc.fr/~bordesan/dokuwiki/doku.php?id=en:sgdqn`
[15] `http://research.microsoft.com/en-us/projects/svm/`
[16] `http://osmot.cs.cornell.edu/kddcup/datasets.html`
[17] `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets`
[18] `http://archive.ics.uci.edu/ml/datasets.html`
[19] `http://www.jmlr.org/papers/volume5/lewis04a/lyrl2004_rcv1v2_README.htm`

Table 1. The number T of complete epochs required in order for the SGD-s algorithm to achieve $(\mathcal{J}-\mathcal{L}^{\mathcal{T}})/\mathcal{L}^{\mathcal{T}} \leq 10^{-5}$ for $C=0.1$.

data set	#instances	#attributes	SGD-s $\epsilon=10^{-5}$ $C=0.1$		
			T	$\mathcal{J}$	$\mathcal{L}^T$
Adult	32561	123	208174	1149.904	1149.893
Web	49749	300	16849	755.1139	755.1064
Physics	50000	70	13668	4995.139	4995.089
Realsim	72309	20958	4209	1437.315	1437.301
News20	19996	1355191	2178	902.5611	902.5521
Webspam	350000	254	27680	8284.781	8284.698
Covertype	581012	54	712648	36427.52	36427.16
C11	804414	47236	5670	5174.432	5174.381
CCAT	804414	47236	7987	12114.29	12114.17

we study the binary classification problem of the first class versus rest while for the RCV1 we consider both the binary text classification tasks of the C11 and CCAT classes versus rest. The Physics and Covertype datasets were rescaled by multiplying all the features with 0.001. The experiments were conducted on a 2.5 GHz Intel Core 2 Duo processor with 3 GB RAM running Windows Vista. The C++ codes were compiled using the g++ compiler under Cygwin.

First we perform an experiment aiming at demonstrating that our SGD algorithms are able to obtain extremely accurate solutions. More specifically, with the algorithm SGD-s employing single updating we attempt to diminish the gap between the primal objective $\mathcal{J}$ and the dual Lagrangian $\mathcal{L}^T$ setting as a goal a relative deviation $(\mathcal{J}-\mathcal{L}^{\mathcal{T}})/\mathcal{L}^{\mathcal{T}} \leq 10^{-5}$ for $C=0.1$. In the present and in all subsequent experiments we do not include a bias term in any of the algorithms (i.e., in our case we assign to the augmentation parameter the value $\rho=0$). In order to keep the number T of complete epochs as low as possible we increase the comparison coefficient f until the number of epochs required gets stabilized. This procedure does not entail, of course, the shortest training time but this is not our concern in this experiment. In Table 1 we give the values of both $\mathcal{J}$ and $\mathcal{L}^T$ and the number T of epochs needed to achieve these values. If multiple updates are used a larger number of epochs is, in general, required due to the slower increase of $\mathcal{L}^T$. Thus, SGD-s achieves, in general, relative accuracy closer to ϵ than SGD-m does. This is confirmed by subsequent experiments.

In our comparative experimental investigations we aim at achieving relative accuracy $(\mathcal{J}-\mathcal{J}_{\text{opt}})/\mathcal{J}_{\text{opt}} \leq 0.01$ for various values of the penalty parameter C assuming knowledge of the value of $\mathcal{J}_{\text{opt}}$. For Pegasos and SGD-r we use as stopping criterion the exhaustion of the maximum number of steps (iterations) $t_{\max}$ which, however, is given values which are multiples of the dataset size m. The ratio $t_{\max}/m$ may be considered analogous to the number T of epochs of the algorithm SGD-s since equal values of these two quantities indicate identical numbers of $\boldsymbol{w}_t$ updates. The input parameter for SGD-s and SGD-m is the (upper bound on) the relative accuracy ϵ. For MPU we use the parameter $\epsilon=\delta=\delta_{\text{stop}}$,

Table 2. Training times of SGD algorithms to achieve $(\mathcal{J} - \mathcal{J}_{\text{opt}})/\mathcal{J}_{\text{opt}} \leq 0.01$ for $C = 1$.

$C = 1$										
data set	Pegasos		SGD-r		SGD-s			SGD-m		
	$t_{\max}/m$	s	$t_{\max}/m$	s	ϵ	T	s	ϵ	T	s
Adult	181	2.4	116	0.52	0.105	111	0.52	0.33	50	0.23
Web	53	0.67	46	0.31	0.054	26	0.19	0.1	14	0.11
Physics	2	0.11	6	0.09	2.1	1	0.02	0.14	3	0.05
Realsim	66	2.8	70	2.0	0.046	20	0.56	0.061	16	0.45
News20	89	9.4	88	7.0	0.023	39	3.1	0.029	25	2.1
Webspam	8	2.9	9	2.0	0.068	14	2.9	0.21	5	1.2
Covertype	-	-	62	10.6	0.264	65	8.2	1.12	18	2.4
C11	41	28.8	39	20.8	0.05	16	7.7	0.136	8	4.1
CCAT	37	29.0	36	19.5	0.055	16	7.8	0.163	7	4.0

Table 3. Training times of linear SVMs to achieve $(\mathcal{J} - \mathcal{J}_{\text{opt}})/\mathcal{J}_{\text{opt}} \leq 0.01$ for $C = 1$.

$C = 1$							
data set	SVM$^{\text{perf}}$		OCAS	DCD		MPU	
	ϵ	s	s	ϵ	s	ϵ	s
Adult	0.7	1.3	0.06	2.8	0.14	0.02	0.09
Web	0.2	0.27	0.27	6	0.06	0.01	0.03
Physics	1.0	0.30	0.02	23	0.06	0.06	0.06
Realsim	0.08	0.75	0.59	0.7	0.20	0.06	0.22
News20	0.14	12.5	6.0	0.4	0.64	0.03	1.5
Webspam	0.5	7.3	4.2	2.5	1.4	0.1	0.95
Covertype	4.2	45.6	3.3	6.5	9.1	0.1	5.8
C11	0.09	12.7	8.9	1.4	3.5	0.09	2.4
CCAT	0.25	19.1	12.9	1.6	3.6	0.1	3.1

where δ is the before-run relative accuracy and δ_{stop} the stopping threshold for the after-run relative accuracy. For SVM$^{\text{perf}}$ and DCD we use as input their parameter ϵ while for OCAS the primal objective value $q = 1.01\mathcal{J}_{\text{opt}}$ (not given in the tables) with the relative tolerance taking the default value $r = 0.01$. Any difference in training time between Pegasos and SGD-r for equal values of $t_{\max}/m$ should be attributed to the difference in the implementations. Any difference between $t_{\max}/m$ for SGD-r and T for SGD-s is to be attributed to the different procedure of choosing the patterns that are presented to the algorithm. Finally, the difference in the number T of epochs between SGD-s and SGD-m reflects the effect of multiple updates. It should be noted that in the runtime of SGD-s and SGD-m several calculations of the primal and the dual objective are included which are required for checking the satisfaction of the stopping criterion. If SGD-s and SGD-m were using the exhaustion of the maximum number $T_{\max}$ of epochs as stopping criterion their runtimes would certainly be shorter.

Tables 2 and 3 contain the results of the experiments involving the SGD algorithms and linear SVMs for $C = 1$. We observe that, in general, there is a

Table 4. Training times of SGD algorithms to achieve $(\mathcal{J} - \mathcal{J}_{\rm opt})/\mathcal{J}_{\rm opt} \leq 0.01$ for $C = 10$.

$C = 10$										
data set	Pegasos		SGD-r		SGD-s			SGD-m		
	$t_{\max}/m$	s	$t_{\max}/m$	s	ϵ	T	s	ϵ	T	s
Adult	-	-	1146	5.1	0.098	1172	5.4	0.35	455	2.2
Web	338	4.3	330	2.3	0.049	220	1.5	0.105	99	0.70
Physics	12	0.66	51	0.75	0.203	17	0.27	0.223	19	0.33
Realsim	746	27.7	738	20.7	0.0162	487	12.7	0.027	261	6.9
News20	796	72.0	797	58.3	0.0104	719	52.4	0.012	279	21.0
Webspam	-	-	64	14.1	0.125	62	12.1	0.4	26	5.6
Covertype	-	-	472	80.3	0.343	462	56.8	0.77	269	34.4
C11	446	308.2	441	234.2	0.0415	178	81.1	0.085	116	53.2
CCAT	-	-	387	207.9	0.0471	170	78.0	0.112	98	46.2

Table 5. Training times of linear SVMs to achieve $(\mathcal{J} - \mathcal{J}_{\rm opt})/\mathcal{J}_{\rm opt} \leq 0.01$ for $C = 10$.

$C = 10$							
data set	SVM$^{\rm perf}$		OCAS	DCD		MPU	
	ϵ	s	s	ϵ	s	ϵ	s
Adult	0.6	37.0	0.30	2.6	1.1	0.04	0.59
Web	0.23	1.2	0.50	8	0.17	0.02	0.06
Physics	1.3	2.8	0.09	23	0.30	0.05	0.20
Realsim	0.031	4.2	2.4	0.25	0.45	0.02	0.39
News20	0.019	146.2	40.7	0.2	1.5	0.02	2.1
Webspam	0.36	36.6	6.8	2.2	5.2	0.2	2.5
Covertype	2.1	51.8	15.8	6.1	90.8	0.08	36.3
C11	0.079	38.6	25.0	0.65	9.1	0.02	5.5
CCAT	0.14	71.1	35.2	0.85	11.7	0.02	7.8

progressive decrease in training time as we move from Pegasos to SGD-m through SGD-r and SGD-s due to the additive effect of several factors. These factors are the more efficient implementation of our algorithms exploiting the change of variable given by (4), the presentation of the patterns to SGD-s and SGD-m in complete epochs (see also [3,19]) and the use by SGD-m of multiple updating. The overall improvement made by SGD-m over Pegasos is quite substantial. DCD and MPU are certainly statistically faster but their differences from SGD-m are not very large especially for the largest datasets. Moreover, SGD-s and SGD-m are considerably faster than SVM$^{\rm perf}$ and statistically faster than OCAS. Pegasos failed to process the Covertype dataset due to numerical problems.

Tables 4 and 5 contain the results of the experiments involving the SGD algorithms and linear SVMs for $C = 10$. Although the general characteristics resemble the ones of the previous case the differences are magnified due to the intensity of the optimization task. Certainly, the training time of linear SVMs scales much better as C increases. Moreover, MPU clearly outperforms DCD and OCAS for most datasets. SGD-m is still statistically faster than SVM$^{\rm perf}$ but slower than OCAS. Finally, Pegasos runs more often into numerical problems.

Table 6. Training times of SGD algorithms to achieve $(\mathcal{J} - \mathcal{J}_{\rm opt})/\mathcal{J}_{\rm opt} \leq 0.01$ for $C = 0.05$.

$C = 0.05$										
data set	Pegasos		SGD-r		SGD-s			SGD-m		
	$t_{\max}/m$	s	$t_{\max}/m$	s	ϵ	T	s	ϵ	T	s
Adult	7	0.09	12	0.05	0.07	10	0.05	0.15	6	0.03
Web	4	0.06	4	0.03	0.06	2	0.02	0.06	2	0.02
Physics	1	0.06	1	0.02	0.11	1	0.02	0.06	1	0.02
Realsim	3	0.19	3	0.08	0.09	1	0.06	0.14	1	0.06
News20	4	0.66	3	0.31	0.08	1	0.19	0.12	1	0.20
Webspam	1	0.39	2	0.45	0.4	1	0.42	0.18	1	0.42
Covertype	5	2.4	4	0.69	0.25	4	0.59	0.27	5	0.75
C11	2	1.4	3	1.6	0.03	2	1.4	0.1	1	0.95
CCAT	2	1.7	2	1.1	0.12	1	0.95	0.12	1	0.98

Table 7. Training times of linear SVMs to achieve $(\mathcal{J} - \mathcal{J}_{\rm opt})/\mathcal{J}_{\rm opt} \leq 0.01$ for $C = 0.05$.

$C = 0.05$							
data set	SVM^{perf}		OCAS	DCD		MPU	
	ϵ	s	s	ϵ	s	ϵ	s
Adult	1.1	0.11	0.05	3	0.02	0.01	0.03
Web	0.3	0.08	0.09	4	0.03	0.01	0.02
Physics	1.1	0.08	0.02	12	0.03	0.02	0.02
Realsim	0.7	0.23	0.19	0.7	0.14	0.2	0.14
News20	0.9	1.3	0.72	3	0.23	0.2	0.55
Webspam	1.1	2.8	2.3	1	1.1	0.2	0.69
Covertype	2.9	53.2	1.5	6	1.2	0.3	1.3
C11	0.11	4.6	2.9	4	1.4	0.1	1.4
CCAT	0.25	7.1	5.7	1	2.6	0.1	1.6

In contrast, as C decreases the differences among the algorithms are alleviated. This is apparent from the results for $C = 0.05$ reported in Tables 6 and 7. SGD-r, SGD-s and SGD-m all appear statistically faster than the linear SVMs. Also Pegasos outperforms SVM^{perf} for all datasets and OCAS for the majority of them. Seemingly, lowering C favors the SGD algorithms.

Before concluding our experimental investigation we also consider the SGD algorithms SvmSgd and SGD-QN both implemented in single precision. For a fair comparison we implemented the algorithms SGD-m and MPU in single precision as well. SvmSgd and SGD-QN do not perform random permutations of the dataset but rather assume that it has already been shuffled. Thus, we provided them with the dataset produced by SGD-m as a result of the first random permutation of the dataset given. The fact that the computational cost of the random permutation is not included in the runtime of SvmSgd and SGD-QN gives these algorithms a certain advantage which becomes more crucial for tasks requiring short runtimes. For this reason we did not consider values of the penalty parameter C much smaller than 1. Table 8 contains the results of the experiments

Table 8. Training times of the algorithms SvmSgd, SGD-QN, SGD-m and MPU implemented in single precision to achieve $(\mathcal{J} - \mathcal{J}_{\rm opt})/\mathcal{J}_{\rm opt} \leq 0.01$ for $C = 1$.

$C = 1$									
data set	SvmSgd		SGD-QN		SGD-m			MPU	
	T	s	T	s	ϵ	T	s	ϵ	s
Adult	261	0.59	39	0.06	0.33	50	0.11	0.02	0.06
Web	49	0.16	39	0.11	0.1	14	0.05	0.01	0.03
Physics	2	0.03	1	0.02	0.14	3	0.03	0.06	0.03
Realsim	29	0.41	27	0.37	0.061	16	0.33	0.06	0.16
News20	199	9.3	-	-	0.029	25	1.7	0.03	1.2
Webspam	10	0.84	6	0.58	0.21	5	0.84	0.1	0.69
Covertype	573	28.6	14	0.65	0.57	42	4.4	0.1	5.6
C11	23	5.0	21	4.7	0.136	8	3.0	0.09	1.9
CCAT	23	5.2	23	5.7	0.163	7	2.9	0.1	2.4

involving the algorithms SvmSgd, SGD-QN, SGD-m and MPU for $C = 1$. We observe that statistically SvmSgd is the slowest and MPU the fastest. Moreover, SGD-m statistically outperforms SGD-QN with a preference for sparse multidimentional datasets. In the case of the News20 dataset SGD-QN failed to reach the required relative accuracy.

4 Conclusions

We reexamined the classical SGD approach to the primal unconstrained L1-SVM optimization task and made some contributions concerning both theoretical and practical issues. Assuming a learning rate inversely proportional to the number of steps a simple change of variable allowed us to simplify the algorithmic description and demonstrate that in a scheme presenting the patterns to the algorithm in complete epochs the naturally defined dual variables satisfy automatically the box constraints of the dual optimization. This opened the way to obtaining an estimate of the progress made in the optimization process and enabled the adoption of a meaningful stopping criterion, something the SGD algorithms were lacking. Moreover, it made possible a qualitative discussion of how the KKT conditions will be asymptotically satisfied provided the weight vector $\boldsymbol{w}_t$ gets stabilized. Besides, we showed that in the limit $t \to \infty$ even without a projection step in the update it holds that $\|\boldsymbol{w}_t\| \leq 1/\sqrt{\lambda}$, a bound known to be obeyed by the optimal solution. On the more practical side by exploiting our simplified algorithmic description and employing a mechanism of multiple updating we succeeded in substantially improving the performance of SGD algorithms. For optimization tasks of low or medium intensity the algorithms constructed are comparable to or even faster than the state-of-the-art linear SVMs.

References

1. Boser, B., Guyon, I., Vapnik, V.: A training algorithm for optimal margin classifiers. In: COLT, pp. 144–152 (1992)
2. Bordes, A., Bottou, L., Gallinari, P.: SGD-QN: Careful quasi-Newton stochastic gradient descent. JMLR **10**, 1737–1754 (2009)
3. Bottou, L. (Web Page). Stochastic gradient descent examples. `http://leon.bottou.org/projects/sgd`
4. Cortes, C., Vapnik, V.: Support vector networks. Mach Learn **20**, 273–297 (1995)
5. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines. Cambridge University Press, Cambridge (2000)
6. Duda, R.O., Hart, P.E.: Pattern Classsification and Scene Analysis. Wiley, Chichester (1973)
7. Frank, V., Sonnenburg, S.: Optimized cutting plane algorithm for support vector machines. In: ICML, pp. 320-327 (2008)
8. Hsieh, C.-J., Chang, K.-W., Lin, C.-J., Keerthi, S.S., Sundararajan, S.: A dual coordinate descent method for large-scale linear SVM. In: ICML, pp. 408–415 (2008)
9. Joachims, T.: Making large-scale SVM learning practical. In: Advances in Kernel Methods-Support Vector Learning. MIT Press, Cambridge (1999)
10. Joachims, T.: Training linear SVMs in linear time. In: KDD, pp. 217–226 (2006)
11. Karampatziakis, N., Langford, J.: Online importance weight aware updates. In: UAI, pp. 392–399 (2011)
12. Kivinen, J., Smola, A., Williamson, R.: Online learning with kernels. IEEE Transactions on Signal Processing **52(8)**, 2165-2176 (2004)
13. Panagiotakopoulos, C., Tsampouka, P.: The margin perceptron with unlearning. In: ICML, pp. 855-862 (2010)
14. Panagiotakopoulos, C., Tsampouka, P.: The margitron: A generalized perceptron with margin. IEEE Transactions on Neural Networks **22(3)**, 395-407 (2011)
15. Panagiotakopoulos, C., Tsampouka, P.: The perceptron with dynamic margin. In: Kivinen, J., et. al. (eds.) ALT 2011. LNCS (LNAI) vol. 6925, pp. 204-218. Springer, Heidelberg (2011)
16. Platt, J.C.: Sequential minimal optimization: A fast algorithm for training support vector machines. Microsoft Res. Redmond WA, Tech. Rep. MSR-TR-98-14 (1998)
17. Rosenblatt, F.: The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review, **65 (6)**, 386–408 (1958)
18. Shalev-Schwartz, S., Singer, Y., Srebro, N.: Pegasos: Primal estimated sub-gradient solver for SVM. In: ICML, pp. 807–814 (2007)
19. Shalev-Schwartz, S., Singer, Y., Srebro, N., Cotter, A.: Pegasos: Primal estimated sub-gradient solver for SVM. Mathematical Programming, **127(1)**, 3-30 (2011)
20. Vapnik, V.: Statistical learning theory. Wiley, Chichester (1998)
21. Zhang, T.: Solving large scale linear prediction problems using stochastic gradient descent algorithms. In: ICML (2004)