Multi-Class Pegasos on a Budget

Zhuang Wang ZHUANG@TEMPLE.EDU
Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122 USA

Koby Crammer KOBY@EE.TECHNION.AC.IL
Department of Electronic Engineering, The Technion, Haifa, 32000 Israel

Slobodan Vucetic VUCETIC@IST.TEMPLE.EDU
Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122 USA

Abstract

When equipped with kernel functions, online learning algorithms are susceptible to the "curse of kernelization" that causes unbounded growth in the model size. To address this issue, we present a family of budgeted online learning algorithms for multi-class classification which have constant space and time complexity per update. Our approach is based on the multi-class version of the popular Pegasos algorithm. It keeps the number of support vectors bounded during learning through budget maintenance. By treating the budget maintenance as a source of the gradient error, we prove that the gap between the budgeted Pegasos and the optimal solution directly depends on the average model degradation due to budget maintenance. To minimize the model degradation, we study greedy multi-class budget maintenance methods based on removal, projection, and merging of support vectors. Empirical results show that the proposed budgeted online algorithms achieve accuracy comparable to non-budget multi-class kernelized Pegasos while being extremely computationally efficient.

1. Introduction

Online learning is an important framework in which a potentially unlimited sequence of training examples is presented one example at a time and can only be seen in a single pass. This is opposed to offline learning where the whole collection of training examples is at hand. The objective is to learn an accurate prediction model from the training stream. Online algorithms often update the prediction model they maintain upon observing a new training example. Considering the potentially high stream rates, it becomes very important to update the model in a computationally efficient manner.

The invention of the Support Vector Machines (SVMs) (Cortes & Vapnik, 1995) inspired a lot of interest in applying the kernel methods for online learning. A large number of online algorithms (e.g. perceptron by Rosenblatt (1958)) can be easily kernelized and result in prediction models that require storage of a subset of observed examples, called the Support Vectors (SVs). While kernelization allows solving highly nonlinear problems, it also introduces heavy computational burden. The main reason is that on noisy data the number of support vectors tends to grow without limit as the algorithm progresses. In addition to the danger of exceeding the physical memory, this also implies an unlimited growth in update and prediction time. We call this phenomenon *the curse of kernelization*. To solve the problem, budgeted online algorithms have been proposed to bound the number of SVs through budget maintenance. In practice, the assigned budget depends on the specific application requirements (e.g., memory limitations, processing speed, or data throughput).

In this paper, we address the problem of online multi-class classification on a budget. The basis of our work is the popular SVM training algorithm Pegasos (Shalev-Shwartz et al., 2007). Pegasos runs iteratively and alternates between a stochastic sub-gradient descent step and a projection step. If only a single example is used in the stochastic sub-gradient descent step, the algorithm can be naturally used for online learning. It was shown that Pegasos converges toward the optimal solution of SVM as the number of examples grows. Although in its original, non-kernelized, form it has constant update time and constant space, when combined with kernels Pegasos suffers from the same computational problems as other kernelized online algorithms.

Appearing in *Proceedings of the 27th International Conference on Machine Learning*, Haifa, Israel, 2010.

The main contributions of this paper are as follows. First, we develop a multi-class version of Pegasos based on the multi-class SVM formulation by Crammer & Singer (2001). The resulting multi-class Pegasos has similar algorithmic structure as its binary version. The second contribution is a budgeted version of the kernelized multi-class Pegasos that has constant update time and constant space. This is achieved by controlling the number of support vectors with one of the several budget maintenance strategies. We analyze the properties of the budgeted multi-class Pegasos in the framework of online convex learning with errors in the gradients proposed by Sutskever (2009). We show that, in the limit, the gap between loss of the budgeted algorithm and loss of the optimal solution is upper-bounded by the average model degradation induced by budget maintenance. In the absence of budget maintenance the multi-class Pegasos inherits convergence properties of its binary predecessor.

Having shown that the quality of budgeted multi-class Pegasos directly depends on the quality of budget maintenance, our final contribution is exploring computationally efficient methods to maintain a classifier with a low budget. This problem has been subject of much recent work. The most popular strategy consists of removing a single support vector when the budget is exceeded. For example, (Crammer et al., 2004) proposed to learn budgeted perceptrons by removing the SV that will be predicted with the largest confidence after its own removal. Other ideas include removal of the oldest SV (Dekel et al., 2008), a random SV (Cesa-Bianchi & Gentile, 2006; Vucetic et al., 2009), one with the smallest coefficient (Cheng et al., 2007), or using a validation data set (Weston et al., 2005; Wang & Vucetic, 2009). Recently studied alternatives to removal are projecting an SV prior to its removal (Orabona et al., 2009) and merging of two SVs into a new SV (Nguyen & Ho, 2005; Wang & Vucetic, 2009). Instead of considering budget maintenance and model update as separate steps, Wang & Vucetic (2010) proposed to define it as a joint optimization problem. It is worth mentioning that much work has been done on the related problem of reduction of trained kernel machines (Schölkopf et al., 1999).

In this work we consider 3 major budget maintenance strategies: removal, projection, and merging. In case of removal, we show that it is optimal to remove the smallest support vector. Then, we show that optimal projection of one SV to the remaining ones is achieved by minimizing the accumulated loss of multiple sub-problems for each class, which extends the results of (Csató & Opper, 2001; Engel et al., 2002; Orabona et al., 2009) to the multi-class setting. In case of merging, when the Gaussian kernel is used, we show that the new SV is always on the line connecting two merged SVs, which generalizes the result of (Nguyen & Ho, 2005) to the multi-class setting. Both space and update time of the budgeted Pegasos scale quadratically with the budget size when projection is used and linearly when merging or removal are used.

2. Budgeted Multi-class Pegasos (BPegasos^M)

In this paper we focus on the problem of multi-class online learning on a budget. We are given a sequence of examples $S = \{(\mathbf{x}_t, y_t), t = 1,..., N\}$, where instance $\mathbf{x}_t \in \mathbb{R}^d$ is a d-dimensional input vector and the label y_t belongs to the multi-class set $Y = \{1,..., c\}$. In the online learning setting, examples arrive sequentially. The objective of online algorithms is to incrementally learn a function $f(\mathbf{x})$ that accurately maps instances from $\mathbb{R}^d$ to one of the possible labels in Y.

2.1 Multi-class Pegasos

Our algorithm is an extension of the recently proposed SVM training algorithm called Pegasos (Shalev-Shwartz et al., 2007). Pegasos is an iterative algorithm which alternates between stochastic sub-gradient descent steps and projection steps. Pegasos can be naturally used as an online learning algorithm when only a single example is used in the calculation of the stochastic sub-gradient.

To develop the multi-class Pegasos, we consider the multi-class SVM formulation by Crammer & Singer (2001). Let us define the multi-class model $f(\mathbf{x})$ as

$$f(\mathbf{x}) = \arg\max_{i \in Y} \{f^{(i)}(\mathbf{x})\} = \arg\max_{i \in Y} \{(\mathbf{w}^{(i)})^T \mathbf{x}\},$$

where $f^{(i)}$ is the prototype of the i-th class and $\mathbf{w}^{(i)}$ is its corresponding weight vector. By adding all class-specific weight vectors, we construct $\mathbf{w} = [\mathbf{w}^{(1)} ... \mathbf{w}^{(c)}]$ as the $d \times c$ weight matrix of $f(\mathbf{x})$. The predicted label of $\mathbf{x}$ is the class of the weight vector that achieves the maximum value $(\mathbf{w}^{(i)})^T \mathbf{x}$. Under this setup, Crammer and Singer (2001) defined the multi-class SVM objective function as

$$P(\mathbf{w}) = \frac{\lambda}{2} \| \mathbf{w} \|^2 + \frac{1}{N} \sum_{t=1}^{N} l(\mathbf{w}; (\mathbf{x}_t, y_t)), \quad (1)$$

where λ is the slack parameter, the norm of the weight matrix $\mathbf{w}$ is calculated as

$$\| \mathbf{w} \|^2 = \sum_{i \in Y} \| \mathbf{w}^{(i)} \|^2,$$

and the multi-class hinge-loss function is defined as

$$l(\mathbf{w}; (\mathbf{x}_t, y_t)) = \max\left(0, 1 + f^{(r_t)}(\mathbf{x}_t) - f^{(y_t)}(\mathbf{x}_t)\right), \quad \text{where } r_t = \arg\max_{i \in Y, i \neq y_t} f^{(i)}(\mathbf{x}_t). \quad (2)$$

In multi-class Pegasos, instead of the objective function (1), we use the instantaneous objective function $P_t(\mathbf{w})$ upon receiving the t-th example,

$$P_t(\mathbf{w}) = \frac{\lambda}{2} \| \mathbf{w} \|^2 + l(\mathbf{w}; (\mathbf{x}_t, y_t)). \quad (3)$$

Similarly to its binary predecessor, the multi-class Pegasos works by iteratively executing the two-step updates. At t-th round, it first updates the current weight $\mathbf{w}_t$ using the sub-gradient ∇_t of (3) as $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \nabla_t$, where $\eta_t = 1/(\lambda t)$ is the learning rate. The sub-gradient matrix ∇_t is defined as $\nabla_t = [\nabla_t^{(1)} ... \nabla_t^{(c)}]$, where $\nabla_t^{(i)} = \nabla_{\mathbf{w}^{(i)}} P_t(\mathbf{w})$ is a column vector. If loss (2) is equal to zero then $\nabla_t^{(i)} = \lambda \mathbf{w}_t^{(i)}$. If loss (2) is above zero, then

$$\nabla_t^{(i)} = \begin{cases} \lambda \mathbf{w}_t^{(i)} - \mathbf{x}_t, & \text{if } i = y_t \\ \lambda \mathbf{w}_t^{(i)} + \mathbf{x}_t, & \text{if } i = r_t \\ \lambda \mathbf{w}_t^{(i)}, & \text{otherwise.} \end{cases}$$

This step is followed by projecting the weight $\mathbf{w}_{t+1}$ into the closed convex set $C = \{\mathbf{w} : \| \mathbf{w} \| \leq 1/\sqrt{\lambda}\}$.

The above two steps are summarized as

$$\mathbf{w}_{t+1} = \prod_C (\mathbf{w}_t - \eta_t \nabla_t),$$

where $\prod_C(\mathbf{u})$ defines the closest point in C to $\mathbf{u}$. We can rewrite the update rule of the multi-class Pegasos as

$$\mathbf{w}_{t+1} = \phi_t(\mathbf{w}_t - \eta_t \nabla_t) = \phi_t((1 - \eta_t \lambda)\mathbf{w}_t + \mathbf{x}_t \boldsymbol{\beta}_t),$$

where $\phi_t = \min\{1, 1/(\sqrt{\lambda} \| \mathbf{w}_t - \eta_t \nabla_t \|)\}$ and $\boldsymbol{\beta}_t$ is a row vector, $\boldsymbol{\beta}_t = [\beta_t^{(1)} ... \beta_t^{(c)}]$. If loss (2) is equal to zero, then $\boldsymbol{\beta} = \mathbf{0}$; otherwise,

$$\beta_t^{(i)} = \begin{cases} \eta_t, & \text{if } i = y_t \\ -\eta_t, & \text{if } i = r_t \\ 0, & \text{otherwise.} \end{cases}$$

2.2 Kernelization

Multi-class Pegasos can learn non-linear problems when the kernel trick is used. After introducing a nonlinear function Φ that maps $\mathbf{x}$ from the input to the feature space and replacing $\mathbf{x}$ with $\Phi(\mathbf{x})$, $\mathbf{w}_t^{(i)}$ can be described as

$$\mathbf{w}_t^{(i)} = \sum_{j=1}^{t} \alpha_j^{(i)} \Phi(\mathbf{x}_j),$$

where

$$\alpha_j^{(i)} = \beta_j^{(i)} \prod_{k=j}^{t} \phi_k \prod_{k=j+1}^{t} (1 - \eta_k \lambda). \quad (4)$$

We denote the row vector $\boldsymbol{\alpha}_j = [\alpha_j^{(1)} ... \alpha_j^{(c)}]$ as the c class-specific coefficients of j-th SV. From (4) it can be seen that the example whose loss (2) is zero has all zero α coefficients and can therefore be ignored. An input example with positive loss has one positive and one negative α coefficient, while the remaining α are zero. We call such examples Support Vectors (SVs). We can represent $f_t^{(i)}(\mathbf{x})$ as the kernel expansion

$$f_t^{(i)}(\mathbf{x}) = (\mathbf{w}_t^{(i)})^T \Phi(\mathbf{x}) = \sum_{j \in I_t} \alpha_j^{(i)} k(\mathbf{x}_j, \mathbf{x}),$$

where k s the Mercer kernel induced by Φ and I_t is the set of indexes of all the SVs of $\mathbf{w}_t$. From now on, we will represent a model using both the $\mathbf{w}$ and α notation interchangeably.

2.3 Budgeted Multi-Class Pegasos

To maintain the fixed number of SVs, the algorithm executes a budget maintenance step whenever the number of SVs exceeds a pre-defined budget B (i.e. $| I_{t+1} | > B$) by reducing the size of I_{t+1} by one such that $\mathbf{w}_{t+1}$ is only spanned by B SVs. We summarize the proposed Budgeted Multi-class Pegasos (BPegasos^M) algorithm in Algorithm 1. We denote the weight degradation caused by budget maintenance step at t-th round as $\Delta \mathbf{w}_t$, defined as the difference between model weights before and after Line 7 of Algorithm 1.

Budget maintenance step (Line 7) is a critical design issue. All budget maintenance strategies mentioned in the Introduction are compatible with Algorithm 1 and can be implemented. We will describe several budget maintenance strategies for BPegasos^M in Section 4. In the next section we theoretically analyze how the budget maintenance step influences performance of BPegasos^M.

Algorithm 1: Budgeted multi-class Pegasos (BPegasos^M)

Input: data S, kernel k, slack parameter λ, budget B;
Initialize: $b = 0$, $\mathbf{w}_1 = \mathbf{0}$;
0. **for** $t = 1,2,\ldots$
1. receive $(\mathbf{x}_t, y_t)$;
2. $\mathbf{w}_{t+1} \leftarrow (1 - \eta_t \lambda)\mathbf{w}_t$;
3. **if** $l(\mathbf{w}_t; (\mathbf{x}_t, y_t)) > 0$
4. $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_{t+1} + \Phi(\mathbf{x}_t)\boldsymbol{\beta}_t$; // add an SV
5. $b = b+1$;
6. **if** $b > B$
7. $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_{t+1} - \Delta \mathbf{w}_t$; //budget maintenance
8. $b = b-1$;
9. $\mathbf{w}_{t+1} \leftarrow \phi_t \mathbf{w}_{t+1}$;
Output: $f_{t+1}(\mathbf{x})$

3. Analysis

We now analyze the convergence properties of BPegasos^M under the framework of online convex programming with errors in the gradients by Sutskever (2009), which is a variant of the classical online convex programming framework by Zinkevich (2003). We start

by showing that the averaged instantaneous objective of Algorithm 1 converges toward the optimal solution and that the gap between these two is upper-bounded by average gradient error (i.e. weighted averaged weight degradation) induced by budget maintenance. We first introduce the following lemma, generalizing a result of (Shalev-Shwartz et al., 2007). Without loss of generality we assume that $\| \Phi(\mathbf{x}) \| \leq 1$.

Lemma 1 The optimal solution of multi-class SVM, from problem (1), $\mathbf{w}^* = \arg\min_{\mathbf{w}} P(\mathbf{w})$ is in a closed convex set with radius $1/\sqrt{\lambda}$.

Proof: The dual objective of the multi-class SVM problem (1) (Crammer & Singer, 2001) is to maximize

$$-\frac{1}{2\lambda}\sum_{t=1}^{N}\sum_{j=1}^{N} k(\mathbf{x}_t, \mathbf{x}_j)\left(\sum_{i\in Y}(\delta_{y_j}^{(i)} - \gamma_j^{(i)})(\delta_{y_t}^{(i)} - \gamma_t^{(i)})\right) - \sum_{j=1}^{N}\sum_{i\in Y}\gamma_j^{(i)}\delta_{y_j}^{(i)} + 1$$

$$\text{s.t. } \forall j \in \{1,...,N\}, \forall i \in Y, \gamma_j^{(i)} \geq 0 \text{ and } \sum_{i\in Y}\gamma_j^{(i)} = 1/N, \tag{5}$$

where $\delta_{y_j}^{(i)} = 1$ if $i = y_j$ and $\delta_{y_j}^{(i)} = 0$ otherwise, and $\gamma_j^{(i)}$ are optimization variables. Let us denote the optimal solution for $\gamma_j^{(i)}$ as $(\gamma_j^{(i)})^*$. The optimal solution of (1) can be written as (Crammer & Singer, 2001)

$$(\mathbf{w}^{(i)})^* = \frac{1}{\lambda}\sum_{j=1}^{N}\left(\delta_{y_j}^{(i)} - (\gamma_j^{(i)})^*\right)\Phi(\mathbf{x}_j). \tag{6}$$

Since the strong duality holds, the optimal primal and dual objectives coincide. Plugging (6) into (5) we get

$$\frac{\lambda}{2}\| \mathbf{w}^* \|^2 + \frac{1}{N}\sum_{j=1}^{N} l(\mathbf{w}^*; (\mathbf{x}_j, y_j)) = -\frac{\lambda}{2}\| \mathbf{w}^* \|^2 - \sum_{j=1}^{N}\sum_{i\in Y}(\gamma_j^{(i)})^*\delta_{y_j}^{(i)} + 1 \tag{7}$$

Rearranging (7) and applying $l(\mathbf{w}^*; (\mathbf{x}_j, y_j)) \geq 0$ and $(\gamma_j^{(i)})^*\delta_{y_j}^{(i)} \geq 0$ we get

$$\lambda \| \mathbf{w}^* \|^2 = 1 - \sum_{j=1}^{N}\sum_{i\in Y}(\gamma_j^{(i)})^*\delta_{y_j}^{(i)} - \frac{1}{N}\sum_{j=1}^{N} l(\mathbf{w}^*; (\mathbf{x}_j, y_j)) \leq 1,$$

leading to the desired bound. ■

With Lemma 1, we are ready to prove the following theorem, which is a variant of Theorem 1 in (Shalev-Shwartz et al., 2007) under the budgeted multi-class setting.

Theorem 1 (*Bounding average instantaneous objective of BPegasos^M*) Let BPegasos^M (Algorithm 1) run on a sequence of examples S. Let $\Delta\mathbf{w}_t$ be the weight degradation due to budget maintenance (Line 7 in Algorithm 1). Define the gradient error as $E_t = \Delta\mathbf{w}_t / \eta_t$ and $\bar{E} = \sum_{t=1}^{N}\| E_t \| / N$. Let $\mathbf{w}^*$ be the optimal solution of (1). Then, we have

$$\frac{1}{N}\sum_{t=1}^{N} P_t(\mathbf{w}_t) \leq \frac{1}{N}\sum_{t=1}^{N} P_t(\mathbf{w}^*) + \frac{G^2(1+\ln(N))}{2\lambda N} + \frac{2\bar{E}}{\sqrt{\lambda}},$$

where $G = \sqrt{\lambda} + 2 + \sqrt{2}$. (8)

Proof: First, we write the update rule of BPegasos^M by treating E_t as the error in the gradient,

$$\mathbf{w}_{t+1} = \prod_C (\mathbf{w}_t - \eta_t \partial_t), \text{ where } \partial_t = \nabla_t + E_t.$$

Let us define the relative progress toward the optimal solution $\mathbf{w}^*$ at t-th round as

$$D_t = \| \mathbf{w}_t - \mathbf{w}^* \|^2 - \| \mathbf{w}_{t+1} - \mathbf{w}^* \|^2.$$

D_t can be lower bounded as

$$\begin{aligned} D_t &= \| \mathbf{w}_t - \mathbf{w}^* \|^2 - \| \textstyle\prod_C(\mathbf{w}_t - \eta_t\partial_t) - \mathbf{w}^* \|^2 \\ &\geq_1 \| \mathbf{w}_t - \mathbf{w}^* \|^2 - \| \mathbf{w}_t - \eta_t\partial_t - \mathbf{w}^* \|^2 \\ &= -\eta_t^2 \| \partial_t \|^2 + 2\eta_t\nabla_t^T(\mathbf{w}_t - \mathbf{w}^*) + 2\eta_t E_t^T(\mathbf{w}_t - \mathbf{w}^*) \\ &\geq_2 -\eta_t^2 G^2 + 2\eta_t\left(P_t(\mathbf{w}_t) - P_t(\mathbf{w}^*) + \frac{\lambda}{2}\| \mathbf{w}_t - \mathbf{w}^* \|^2\right) - 4\eta_t \| E_t \| / \sqrt{\lambda}. \end{aligned} \tag{9}$$

In $\geq_1$ we use the fact that $\| \prod_C(a) - b \| \leq \| a - b \|$ for all $b \in C$ and a, since C is convex. In $\geq_2$, we bound $\| \partial_t \|$ as $\| \partial_t \| \leq \| \nabla_t \| + \| E_t \| \leq \lambda \| \mathbf{w}_t \| + 2 + \sqrt{2} = \sqrt{\lambda} + 2 + \sqrt{2} \equiv G$,

where, by using (4), we observed that $\| E_t \|$ is upper bounded by $\sqrt{2}$ when the budget maintenance removes an arbitrary SV with index t',

$$\| E_t \| = \| \frac{\boldsymbol{\alpha}_{t'}}{\eta_t} \| = \sqrt{2}\eta_{t'}\left((\prod_{j=t'}^{t-1}\phi_j)\prod_{j=t'+1}^{t}(1-\eta_j\lambda)\right)/(\frac{1}{\lambda t}) = \sqrt{2}(\prod_{j=t'}^{t-1}\phi_j) \leq \sqrt{2}. \tag{10}$$

We also apply the property of strong convexity to bound

$$\nabla_t^T(\mathbf{w}_t - \mathbf{w}^*) \geq P_t(\mathbf{w}_t) - P_t(\mathbf{w}^*) + \lambda \| \mathbf{w}_t - \mathbf{w}^* \|^2 / 2,$$

since P_t is λ-strongly convex function w.r.t. $\| \mathbf{w} \|^2 / 2$ (according to Lemma 1 in (Shalev-Shwartz & Singer, 2007)) and $\nabla_t = \nabla_{\mathbf{w}} P_t(\mathbf{w})$. We bound $\| \mathbf{w}_t - \mathbf{w}^* \| \leq 2/\sqrt{\lambda}$, since both $\mathbf{w}_t$ and $\mathbf{w}^*$ are in the closed convex set with radius $1/\sqrt{\lambda}$ (Lemma 1).

Dividing both sides of inequality (9) by $2\eta_t$ and rearranging we obtain

$$P_t(\mathbf{w}_t) - P_t(\mathbf{w}^*) \le \frac{D_t}{2\eta_t} - \frac{\lambda}{2} \| \mathbf{w}_t - \mathbf{w}^* \|^2 + \frac{\eta_t G^2}{2} + \frac{2 \| E_t \|}{\sqrt{\lambda}}.$$

Summing over all t we get

$$\sum_{t=1}^{N} P_t(\mathbf{w}_t) - \sum_{t=1}^{N} P_t(\mathbf{w}^*) \le \sum_{t=1}^{N} \frac{D_t}{2\eta_t} - \sum_{t=1}^{N} \frac{\lambda}{2} \| \mathbf{w}_t - \mathbf{w}^* \|^2 + \frac{G^2}{2} \sum_{t=1}^{N} \eta_t + \frac{2}{\sqrt{\lambda}} \sum_{t=1}^{N} \| E_t \|. \quad (11)$$

We bound the first and second terms in (11) as

$$\frac{1}{2} \sum_{t=1}^{N} \left(\frac{D_t}{\eta_t} - \lambda \| \mathbf{w}_t - \mathbf{w}^* \|^2 \right) = \frac{1}{2} \left((\frac{1}{\eta_1} - \lambda) \| \mathbf{w}_1 - \mathbf{w}^* \|^2 + \sum_{t=2}^{N} \left(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}} - \lambda \right) \| \mathbf{w}_t - \mathbf{w}^* \|^2 - \frac{1}{\eta_N} \| \mathbf{w}_{N+1} - \mathbf{w}^* \|^2 \right)$$

$$=_3 - \frac{1}{2\eta_N} \| \mathbf{w}_{N+1} - \mathbf{w}^* \|^2 \le 0 \quad (12)$$

In $=_3$, the first and second components vanish after plugging $\eta_t \equiv 1/(\lambda t)$.

Next, we bound the third term in inequality (11) according to the divergence rate of harmonic series

$$\frac{G^2}{2} \sum_{t=1}^{N} \eta_t = \frac{G^2}{2\lambda} \sum_{t=1}^{N} \frac{1}{t} \le \frac{G^2}{2\lambda} (\ln(N) + 1). \quad (13)$$

Combing inequality (12) and (13) with (11) and dividing two sides of inequality by N we get the stated bound. ■

Observe that as N grows, the second term in the right side of bound (8) converges to zero. Therefore, the averaged instantaneous loss of Algorithm 1 converges toward the averaged instantaneous loss of optimal solution, and the gap between these two is upper bounded by the averaged gradient error $\overline{E}$. This result directly indicates that *an optimal budget maintenance strategy is to minimizes* $\overline{E}$.

Corollary 1 (*Mistake bound*) Assume that the conditions stated in Theorem 1 hold, then the number of mistakes made by BPegasos^{M} on the sequence S is

$$M \le \sum_{t=1}^{N} P_t(\mathbf{w}^*) + \frac{G^2(1 + \ln(N))}{2\lambda} + \frac{2\overline{E}N}{\sqrt{\lambda}},$$

where $G = \sqrt{\lambda} + 2 + \sqrt{2}$.

Proof: Using the fact that $l(\mathbf{w}; (\mathbf{x}_t, y_t)) \ge 1$ whenever the algorithm made the mistake, as well as the fact that the accumulated multi-class hinge loss is less than the accumulated instantaneous objective, we get

$$M \le \sum_{t=1}^{N} l(\mathbf{w}_t; (\mathbf{x}_t, y_t)) \le \sum_{t=1}^{N} P_t(\mathbf{w}).$$

Combining with the conclusion in Theorem 1 leads to the stated mistake bound. ■

It is easy to show that the other convergence properties of Pegasos (Theorem 2 and 3 in (Shalev-Shwartz et al., 2007)) are inherited by BPegasos^{M} under the constraint of $\overline{E}$. If there is no budget maintenance step (i.e. $\overline{E} = 0$), we obtain the multi-class counterparts of Shalev-Shwartz et al.'s theorems. We omit this part due to the lack of space.

4. Budget Maintenance Strategies

Theorem 1 indicates that budget maintenance should attempt to minimize the gradient error $\overline{E}$. To minimize $\overline{E}$ in the setting of online learning on a budget, we propose to minimize the gradient error $\| E_t \|$ at each round. From the definition of $\| E_t \|$ in Theorem 1, minimizing $\| E_t \|$ is equivalent to minimizing weight degradation $\| \Delta \mathbf{w}_t \|$,

$$\min \| \Delta \mathbf{w}_t \|^2. \quad (14)$$

4.1 Budget maintenance through removal

If budget maintenance removes j-th SV, $\Delta \mathbf{w}_t = \Phi(\mathbf{x}_j) \boldsymbol{\alpha}_j$. Then, the optimal solution of (14) is removal of SV with the smallest norm, $p = \arg\min_{j \in I_{t+1}} \| \boldsymbol{\alpha}_j \|^2 k(\mathbf{x}_j, \mathbf{x}_j)$. Let us consider the Gaussian kernel case where $k(\mathbf{x}_j, \mathbf{x}_j) = 1$. Then, as seen from (4), the optimal removal always selects the oldest SV and this strategy becomes similar to Forgetron (Dekel et al., 2008).

4.2 Budget maintenance through projection

Let us consider budget maintenance through projecting the p-th SV to the remaining SVs. The objective is to update α coefficients of the remaining SVs to best represent α coefficients of the p-th SV.

$$\min_{\Delta \boldsymbol{\alpha}} \sum_{i \in Y} \| \alpha_p^{(i)} \Phi(\mathbf{x}_p) - \sum_{j \in I_{t+1} - p} \Delta \alpha_j^{(i)} \Phi(\mathbf{x}_j) \|^2. \quad (15)$$

After setting the gradient of (15) with respect to the class-specific column vector of coefficients $\Delta \boldsymbol{\alpha}^{(i)}$ to zero, one can obtain the optimal solution as

$$\forall i \in Y, \Delta \boldsymbol{\alpha}^{(i)} = \alpha_p^{(i)} \mathbf{K}^{-1} \mathbf{k}_p, \quad (16)$$

where $\mathbf{K} = [k_{ij}], \forall i, j \in I_{t+1} - p$ is the kernel matrix, $k_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, and $\mathbf{k}_p = [k_{pj}]^T, \forall j \in I_{t+1} - p$ is the column vector. It is worth observing that inverting $\mathbf{K}$ can be done in $O(B^2)$ time using Woodbury formula (Cauwenberghs &

Poggio, 2000). Finally, upon removal of p-th SV, $\Delta\boldsymbol{\alpha}$ are added to α of the remaining SVs.

The remaining issue is finding the best among B+1 candidate SVs for projection. After plugging (16) into (15) we can observe that the minimal weight degradation of projecting equals

$$\min \| \Delta \mathbf{w}_t \|^2 = \min_{p \in I_{t+1}} \| \boldsymbol{\alpha}_p \|^2 \left(k_{pp} - \mathbf{k}_p^T (\mathbf{K}^{-1} \mathbf{k}_p) \right). \quad (17)$$

Considering there are B+1 SVs, evaluation of (17) requires $O(B^3)$ time for each budget maintenance step. As an efficient approximation, we propose a simplified solution that always projects the smallest SV, $p = \arg\min_{j \in I_{t+1}} \| \boldsymbol{\alpha}_j \|^2 \, k(\mathbf{x}_j, \mathbf{x}_j)$. Then, the computation is reduced to $O(B^2)$. We should also note that the space requirement of projection is $O(B^2)$ needed to store the kernel matrix and its inverse.

Unlike the recently proposed projection method for multi-class perceptron (Orabona et al., 2009) that projects an SV only onto the SVs assigned to the same class, our method solves more general case by projecting an SV onto all the remaining SVs and thus results in smaller weight degradation.

4.3 Budget maintenance through Merging

Problem (14) can also be solved by merging two SVs to a newly created one. The justification is as follows. For the i-th class weight, if $\Phi(\mathbf{x}_m)$ and $\Phi(\mathbf{x}_n)$ are replaced by $M^{(i)} = \left(\alpha_m^{(i)} \Phi(\mathbf{x}_m) + \alpha_n^{(i)} \Phi(\mathbf{x}_n)\right) / (\alpha_m^{(i)} + \alpha_n^{(i)})$ (assuming $\alpha_m^{(i)} + \alpha_n^{(i)} \neq 0$) and the coefficient of $M^{(i)}$ is set to $\alpha_m^{(i)} + \alpha_n^{(i)}$, then the weight remains unchanged. The difficulty is that $M^{(i)}$ cannot be used directly because the pre-image of $M^{(i)}$ may not exist. Therefore we need to approximate $M^{(i)}$ by image $\Phi(\mathbf{z})$ of some input space vector $\mathbf{z}$. Considering the multi-class problem, $\mathbf{z}$ can be found as

$$\min_{\mathbf{z}} \sum_{i \in Y} \| M^{(i)} - \Phi(\mathbf{z}) \|^2 . \quad (18)$$

Let us assume the Gaussian kernel $k(\mathbf{x}, \mathbf{x}') = \exp(-\sigma \|\mathbf{x} - \mathbf{x}'\|^2)$ is used. Problem (18) can then be reduced to

$$\max_{\mathbf{z}} \sum_{i \in Y} (M^{(i)})^T \Phi(\mathbf{z}) . \quad (19)$$

Setting the gradient of (19) with respect to $\mathbf{z}$ to zero leads to solution

$$\mathbf{z} = h \mathbf{x}_m + (1 - h) \mathbf{x}_n , \quad (20)$$

where $h = \dfrac{\sum_{i \in Y} \frac{1}{(\alpha_m^{(i)} + \alpha_n^{(i)})} \alpha_m^{(i)} \exp(-\sigma \| \mathbf{x}_m - \mathbf{z} \|^2)}{\sum_{i \in Y} \frac{1}{(\alpha_m^{(i)} + \alpha_n^{(i)})} \sum_{k=m,n} \alpha_k^{(i)} \exp(-\sigma \| \mathbf{x}_k - \mathbf{z} \|^2)}$.

Eq. (20) indicates that $\mathbf{z}$ lies on the line connecting $\mathbf{x}_m$ and $\mathbf{x}_n$. Plugging (20) into (19), merging problem is simplified to finding

$$\max_{h \in \mathbb{R}} \left(\sum_{i \in Y} \frac{\alpha_m^{(i)}}{\alpha_m^{(i)} + \alpha_n^{(i)}} \right) k_{mn}^{(1-h)^2} + \left(\sum_{i \in Y} \frac{\alpha_n^{(i)}}{\alpha_m^{(i)} + \alpha_n^{(i)}} \right) k_{mn}^{h^2} . \quad (21)$$

We can use any efficient line search method (e.g. golden search) to find the optimal h. After that, the optimal $\mathbf{z}$ can be calculated using (20).

After obtaining the optimal solution $\mathbf{z}$, the optimal coefficients $\boldsymbol{\alpha}_{\mathbf{z}} = [\alpha_z^{(1)} \ldots \alpha_z^{(c)}]$ of $\mathbf{z}$ for approximating $\alpha_m^{(i)} \Phi(\mathbf{x}_m) + \alpha_n^{(i)} \Phi(\mathbf{x}_n)$ are obtained by solving the following optimization problem

$$\min_{\boldsymbol{\alpha}_z} \sum_{i \in Y} \| \alpha_m^{(i)} \Phi(\mathbf{x}_m) + \alpha_n^{(i)} \Phi(\mathbf{x}_n) - \alpha_z^{(i)} \Phi(\mathbf{z}) \|^2 . \quad (22)$$

The optimal solution of (22) is

$$(\alpha_z^{(i)})^* = \alpha_m^{(i)} k(\mathbf{x}_m, \mathbf{z}) + \alpha_n^{(i)} k(\mathbf{x}_n, \mathbf{z}), \ \forall i \in Y .$$

The total cost of finding the optimal merging for the n-th and m-th SV is $O(1)$. The remaining question is what pair of SVs leads to the small weight degradation. The optimal solution can be found by performing merging of all $B(B-1)/2$ pairs of SVs that would require $O(B^2)$ time. To simplify the computation, we use the same approximation method as in projection (Section 4.2) by fixing m as the SV with the smallest value of $\| \boldsymbol{\alpha}_m \|^2$. Thus the computation is reduced to $O(B)$. We should observe that the space requirement is also $O(B)$ because there is no need to store the kernel matrix.

5. Experiments

In this section we evaluate the proposed algorithms on several large datasets whose properties are summarized in the first column of Table 1. *Checkerboard* is generated. *Letter*, *USPS, Covertype* and *Waveform* are standard UCI Repository benchmarks. Attributes in all data sets were scaled to mean 0 and standard deviation 1. In the experiment, we evaluated budget maintenance methods for BPegasos^M explained in Section 4. We used budgets of B = 100 and 500 and set $\lambda = 10^{-4}$. In particular, we compared the proposed projection and merging methods with the random removal method (Cesa-Bianchi & Gentile, 2006) and the method that removes the SV with the smallest coefficient (Cheng et al., 2007). The results of the non-budgeted Pegasos^M are also reported to

Table 1. Testing accuracy comparison. The lower script in the Pegasos^{M} column is #examples being trained before early stopped.

DATA SET (N, DIM, |Y|)	PEGASOS^{M} (#SV)	*B*	PRJTRN++	BP^{M}+RAND	BP^{M}+SMAL	BP^{M}+PROJ	BP^{M}+MRG
USPS	94.2±0.3	*100*	81.1±3.2	78.3±1.5	78.6±4.0	90.5±0.4	**92.0±0.2**
(7K, 256, 10)	(4.2K)	*500*	92.0±0.5	88.5±0.6	89.4±0.6	92.4±0.4	**93.9±0.3**
LETTER	95.7±0.1	*100*	46.3±1.8	39.9±1.8	41.7±0.7	**76.3±0.9**	72.0±1.3
(16K, 16, 26)	(10K)	*500*	76.6±1.1	68.1±0.8	68.5±1.1	87.3±0.6	**89.5±0.3**
COVERTYPE	81.1±0.1	*100*	62.5±3.1	58.1±0.7	57.5±2.4	70.1±0.6	**72.0±0.2**
(0.5M, 54, 7)	($41K_{72K}$)	*500*	67.3±2.9	61.6±0.7	62.3±1.1	74.9±0.2	**76.8±0.2**
WAVEFORM	86.1±0.6	*100*	80.7±0.8	79.1±1.1	79.1±2.9	85.0±0.4	**85.9±0.6**
(2M, 21, 3)	($53K_{140K}$)	*500*	83.5±0.5	82.7±0.6	82.9±1.0	86.1±0.2	**86.9±0.1**
CHECKERB	99.2± 0.1	*100*	96.9±0.4	83.6±1.0	83.9±1.1	98.2±0.3	**99.5±0.1**
(10M, 2, 2)	($63K_{540K}$)	*500*	98.2±0.3	90.3±0.4	90.9±0.5	99.0±0.2	**99.8±0.0**

establish an upper-bound on accuracy. Besides our BPegasos^{M} framework, we also evaluated the Projectron++ algorithm (Orabona et al., 2009) which is the state-of-the-art budgeted kernel perceptron algorithm. In Projectron++, an SV is projected only if model degradation is below the threshold; otherwise, budget is increased by one SV. In our experiments, we set the Projectron++ threshold such that the number of SVs is equal to *B* of BPegasos^{M} at the end of training. All algorithms used Gaussian kernels whose width σ was optimized for each combination of data set, algorithm and budget, choosing among $\{2^0/d, 2^2/d, 2^4/d, 2^6/d\}$, where *d* is data dimensionality. All the algorithms were implemented in MATLAB and the experiments were run on a 2.1GHz Dual-Core processor with 4G memory under Linux.

The accuracy of different algorithms on test data are reported in Table 1. Each result is computed using an average of 5 repetitions on different permutations of each data set. Pegasos^{M} was stopped after 3,000 seconds. From Table 1 we can observe that BPegasos^{M} with merging and projection significantly outperformed both their removal cousin and Projectron++. Merging resulted in somewhat better performance than projecting.

On *Waveform* and *Checkerboard* data, BPegasos^{M} with merging achieved even higher accuracy than the non-budgeted Pegasos^{M} that had to be stopped after 140K and 540K examples, respectively. On *Covertype* and *Letter* data, the accuracy gap between budget $B = 500$ and non-budgeted algorithms remained large and it can be explained by the complexity of these problems; for example, 60% of *Covertype* examples became SVs in Pegasos^{M} and *Letter* has 26 class labels. In both data sets, the accuracy clearly improved form $B = 100$ to 500, which indicates that extra budget is needed for comparable accuracy.

In Figure 1 and 2 the accuracy evolution curves are plotted to illustrate the anytime prediction quality. Accuracy curves of BPegasos^{M} with merging and projection closely followed and eventually surpassed that of Pegasos^{M}. BPegasos^{M} with removal did not perform particularly well.

Figure 3 shows log-log plot of the running time vs. the data stream length. Excluding the initial stage, the non-budgeted Pegasos^{M} had the fastest increase in training time, confirming the expected $O(N^2)$ runtime. On the budget side, the runtime time of BPegasos^{M} with merging and projection increases linearly, with merging having better constant, as expected (merging has constant $O(B)$ and projection $O(B^2)$).

Considering accuracy, runtime, and memory expenditure, BPegasos^{M} with merging is the clear winner on all 5 benchmark datasets.

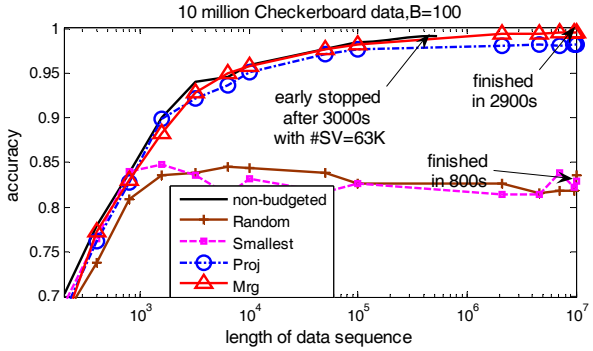

Figure 1. Accuracy evolution curve on *Checkerboard*

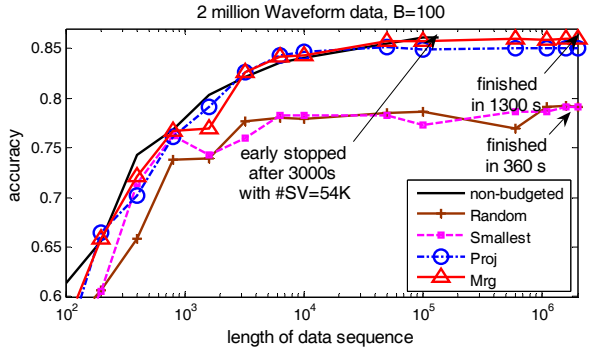

Figure 2. Accuracy evolution curve on *Waveform*

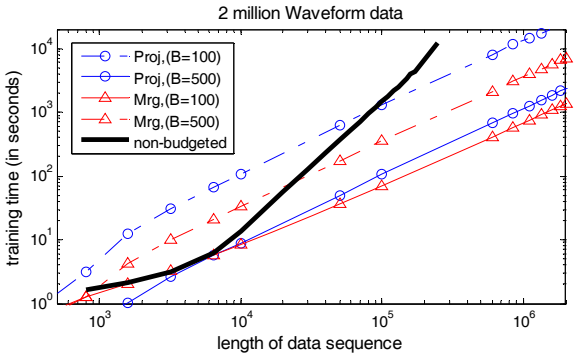

Figure 3. Training time curves

6. Conclusion

We proposed a family of kernel-based budgeted online algorithms for multi-class SVM training based on the Pegasos algorithm. We obtained theoretical guarantees for its performance that indicate that its success is clearly tied with the model degradation due to budget maintenance. Based on the analysis, three budget maintenance methods were studied. We experimentally evaluated the proposed methods in terms of accuracy and training time. The results indicate that highly accurate multi-class kernel classifiers can be trained on high throughput large data streams while having very modest memory signature.

Acknowledgements

This work was supported by the U.S. National Science Foundation Grant IIS-0546155. Koby Crammer is a Horev Fellow, supported by the Taub Foundations.

References

Cauwenberghs, G. & Poggio, T. (2000). Incremental and decremental support vector machine learning. *NIPS*, 13, 409-415.

Cesa-Bianchi, N. & Gentile, C. 2006. Tracking the best hyperplane with a simple budget Perceptron. *COLT*.

Cheng, L., Vishwanathan, S. V. N., Schuurmans, D., Wang, S. & T. Caelli. (2007). Implicit online learning with kernels. *NIPS*, 19, 249-256.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20, 273-297.

Crammer, K., Kandola, J. & Singer, Y. (2004). Online classification on a budget. *NIPS*, 16, 225-232.

Crammer, K & Singer. Y. (2001). On the algorithmic implementation of multiclass kernel-based vector machines. *JMLR*, 2, 262-292.

Csató, L., & Opper, M. (2001). Sparse representation for gaussian process models. *NIPS*, 13, 444-450.

Dekel, O., Shalev-Shwartz, S. & Singer, Y. (2008). The forgetron: A kernel-based perceptron on a budget. *SIAM Journal on Computing*, 37, 1342-1372.

Engel, Y., Mannor, S. & Meir, R. (2002). Sparse online greedy support vector regression. *ECML*.

Hsu, C.-W. & Lin, C.-J. (2002). A comparisons of methods multiclass support vector machines. *IEEE Transactions on Neural Networks*, 13, 415-425.

Nguyen, D & Ho, T. (2005). An efficient method for simplifying support vector machines. *ICML*, 617-624.

Orabona, F. Keshet, J. & Caputo, B. (2009). Bounded kernel-based online learning. *JMLR*, 10, 2643-2666.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65, 386-408.

Schökopf, B, Mika, S., Burges, C. J. C., Knirsch, P., Müler, K., Räsch, G. & Smola, A. J. (1999). Input space versus feature space in kernel-based methods. *IEEE Transactions on Neural Networks*, 10, 1000-1017.

Shalev-Shwartz, S., & Singer, Y. (2007). Logarithmic regret algorithms for strongly convex repeated games (Technical Report). The Hebrew University.

Shalev-Shwartz, S., Singer, Y., & Srebro, N. (2007). Pegasos: primal estimated sub-gradient solver for svm. *ICML*, 807-814.

Sutskever, I. (2009). A simpler unified analysis of budget perceptrons. *ICML*, 985-992.

Vucetic, S, Coric, V., & Wang, Z. (2009). Compressed Kernel Perceptrons. *DCC*, 153-162.

Wang, Z & Vucetic, S. (2009). Twin Vector Machines for Online Training on a Budget. *SDM*.

Wang, Z & Vucetic, S. (2009). Tighter Perceptron with Improved Dual Use of Cached Data for Model Representation and Validation. *IJCNN*, 2766-2771.

Wang, Z & Vucetic, S. (2010). Online Passive-Aggressive Algorithms on a Budget. *AISTATS*.

Weston, J., Bordes, A. & Bottou, L. (2005). Online (and offline) on an even tighter budget. *AISTATS*, 413-420.

Zinkevich, M. (2003). Online convex programming and generalized infinitesimal gradient ascent. *ICML*, 928-935.