Exact Passive-Aggressive Algorithm for Multiclass Classification Using Support Class

Shin Matsushima* Nobuyuki Shimizu† Kazuhiro Yoshida‡ Takashi Ninomiya§
Hiroshi Nakagawa¶

Abstract

The Passive Aggressive framework [1] is a principled approach to online linear classification that advocates minimal weight updates i.e., the least required so that the current training instance is correctly classified. While the PA framework allows integration with different loss functions, it is yet to be combined with a multiclass loss function that penalizes every class with a score higher than the true class. We call the method of training the classifier with this loss function the *Support Class Passive Aggressive Algorithm*. In order to obtain a weight update formula, we solve a quadratic optimization problem by using multiple constraints and arrive at a closed-form solution. This lets us obtain a simple but effective algorithm that updates the classifier against multiple classes for which an instance is likely to be mistaken. We call them the *support classes*. Experiments demonstrated that our method improves the traditional PA algorithms.

1 Introduction

The multiclass classification problem in supervised learning involves building a classifier that categorizes an input to one class from a pool of more than two classes, a setting widely found in many real-world situations. We restricted ourselves to learning a multiclass linear classifier, a model with a linear decision boundary parameterized by a weight vector. The most widely known methods for training a linear classifier include multiclass SVM and multiclass logistic regression. These two methods are known as batch learners, which calculate one optimal solution that discriminatively separates the classes *over the whole data*. Batch-learners are widely used and studied; however, in reality, the optimization problem for a large quantity of data often requires huge amounts of computation time and memory, to the extent that it may be impossible to find a solution at reasonable computational cost. In contrast, online learners calculate how the present weight vector can be modified *with each reception of an input*. This strategy of online learning enables us to apply a learner to larger data on the fly, which makes it very attractive in situations such as those in natural language processing (NLP). Most familiar online learners such as the perceptron can actually run at smaller computational costs than batch learner, however it is typically not accurate as batch learners. To narrow the performance gap, we propose the use of a new loss function in an online learning framework called Passive-Aggressive.

The PA algorithm [1] is a principled method of training a linear classifier online and it can be applied to many tasks including multiclass classification, ranking, and multitask learning once a loss function is given for them. Moreover, the algorithm guarantees termination in the separable case and an upper bound for the sum of the loss function (or *regret*) over the course of processing the whole data set.

The multiclass PA algorithm in the existing literature is actually a departure from the principle underlying the PA framework, which dictates that we should update the weight vector just enough to correctly classify the received input. In short, the multiclass PA algorithm, and other learners such as perceptron, ignores the structure of multiple classes and reduces the problem to a binary classification. To address this issue, our method, which we call the Support class Passive-Aggressive algorithm or SPA algorithm, first determines the set of classes that we call *support class* when an input is received. We then update the weight vector with respect to all support classes. Properly distinguishing a support class enables us to update the decision boundary so that we can correctly classify the received input with minimal change from the previous parameters. We induce this SPA update by solving the optimization problem as stated by the principle behind the PA framework, using the Lagrange multipliers of the nonlinear programming methodology. Moreover, we analyze convergence in the case of separable data and the upper bound of loss function in strict situations.

The contribution of this paper is

*The University of Tokyo, masin@r.dl.itc.u-tokyo.ac.jp
†The University of Tokyo, shimizu@r.dl.itc.u-tokyo.ac.jp
‡no affiliation, kyoshiddha@gmail.com
§The University of Tokyo, ninomi@r.dl.itc.u-tokyo.ac.jp
¶The University of Tokyo, n3@dl.itc.u-tokyo.ac.jp

- We propose a novel online multiclass linear classifiers, SPA, SPA-I and SPA-II, that improve accuracy of PA in one pass online situation. We deduce SPA algorithms by solving strict optimization problems following the principle underlying the PA framework.

- We demostrate that the proposed algorithms have competitive accuracy to standard batch learners in batch situation. They have much lower computational cost than batch learners do.

- We prove the termination of SPA in the separable case and a loss bound in another special case.

This paper is organized as follows. In Sec. 2, we review the PA framework and the existing application of PA to multiclass classification. In Sec. 3, we derive the update rule as a closed-form solution to a quadratic optimization problem and the algorithm that selects the support classes. In Sec. 4, we prove that SPA has the same bounds as PA, but these bounds are tighter than that of PA. In Sec. 5, a series of experiments demonstrates the SPA's empirical benefits. Finally, we conclude in Sec. 6 with a discussion of future work.

2 Passive-Aggressive Algorithm for Multiclass Classification

First, let us introduce some definitions and notations. Online learning of K-class classification proceeds as follows: in the i-th round of learning, (1) the learner is given an instance, $\mathbf{x}_i \in X \subset \mathbb{R}^D$, (2) makes a prediction, $p_i \in Y = \{1, \cdots, K\}$, according to a discriminative function, $h^{(i)} \in \mathcal{H}$, where $\mathcal{H}$ is a set of feasible discriminative functions, (3) receives the true class, $y_i \in Y$, and chooses a new discriminative function, $h^{(i+1)}$, according to the (dis)agreement between the prediction, p_i, and the true label, y_i. In this paper, we restrict the discriminative function of the learner to multiclass linear classifiers represented by K weight vectors of D dimension $\{\mathbf{w}_u\}_{u\in Y}$; in other words, a discriminative function, $h^{(i)}$, is represented with vectors $\mathbf{w}_u^{(i)} (u = 1 \cdots K)$ with the formula,

$$h^{(i)}(\mathbf{x}) = \underset{u=1,\cdots K}{\arg\max} \left(\mathbf{w}_u^{(i)} \cdot \mathbf{x}\right).$$

We call $\mathbf{w}_u^{(i)} \cdot \mathbf{x}$ the *score* of class u (at $\mathbf{x}$). Then, the discriminative function returns a class whose score is the highest as the predicted label, p_i, when the classifier received the instance, $\mathbf{x}_i$. In the Passive-Aggressive framework, the learner receives the true class, y_i, after predicting class p_i and suffers a loss according to a loss function, $\ell(\mathbf{w}^{(i)}; (\mathbf{x}_i, y_i))$,where $\mathbf{w}^{(i)} = \{\mathbf{w}_u^{(i)}\}_{u\in Y}$. The loss function is formally a map onto non-negative number whose domain is $\mathcal{H} \times (X \times Y)$.

When the learner suffers any loss, the PA algorithms require the discriminative function to be updated with new weight vectors by solving the optimization problem

$$\text{(2.1)} \quad \mathbf{w}^{(i+1)} = \underset{\mathbf{w}}{\arg\min} \quad \frac{1}{2}\sum_{u=1}^{K} \left\|\mathbf{w}_u - \mathbf{w}_u^{(i)}\right\|^2 \quad \text{s.t.} \quad \ell(\mathbf{w}; (\mathbf{x}_i, y_i)) = 0$$

Here, the optimal solution, $\mathbf{w}^{(i+1)}$, is the nearest point in the space of weight vectors that would suffer zero loss against the current input. Specifically, updating the discriminative function leaves the weight vectors unchanged if the loss of the learner is zero against the current input, $\mathbf{x}_i$.

As we may simply define a suitable loss function and employ the PA framework to obtain an online algorithm, the PA algorithms can be used for various tasks. Of these is the task of multiclass classification [1], and it gives the basic definition of the multiclass loss function as

$$\ell(\mathbf{w}^{(i)}; (\mathbf{x}_i, y_i)) = \max_{u\neq y_i} \left[1 - \left(\mathbf{w}_{y_i}^{(i)} \cdot \mathbf{x}_i - \mathbf{w}_u^{(i)} \cdot \mathbf{x}_i\right)\right]_+,$$

where $[\bullet]_+$ is threshold function $\max(\bullet, 0)$.

In the loss function, $(\mathbf{w}_{y_i} \cdot \mathbf{x}_i - \mathbf{w}_u \cdot \mathbf{x}_i)$ is the difference between the score of y_i and that of u, and it represents the degree of a classifier's tendency to prefer y_i to u. This term is the *margin* from class y_i to another class u. If the margin is positive, the algorithm will make a prediction that agrees with the true given label. However, the definition of the loss function simply claims being positive is insufficient for the loss of the learner to be zero. To confidently make a prediction, the loss function claims that the margin should be larger than 1 or else the learner should suffer a positive loss.

We can see that there is no need to update the weight vectors when the loss function is zero, because its optimality is obvious. Otherwise, we need to determine the new weight vector by solving the optimization problem above. We rewrite the problem as

$$\text{(2.2)} \quad \underset{\mathbf{w}}{\arg\min} \quad \frac{1}{2}\sum_{u=1}^{K} \left\|\mathbf{w}_u - \mathbf{w}_u^{(i)}\right\|^2 \quad \text{s.t.} \quad (\mathbf{w}_{y_i} \cdot \mathbf{x}_i - \mathbf{w}_u \cdot \mathbf{x}_i) \geq 1 \quad (\forall u \neq y_i).$$

Due to the difficulty involved in exactly solving the update, Crammer et al. [1] aimed to solve a relaxed, less constrained problem that removed all the constraints except that for the predicted class

$$\text{(2.3)} \qquad \arg\min_{\mathbf{w}} \quad \frac{1}{2}\sum_{u=1}^{K}\left\|\mathbf{w}_u - \mathbf{w}_u^{(i)}\right\|^2$$
$$\text{s.t.} \quad (\mathbf{w}_{y_i}\cdot\mathbf{x}_i - \mathbf{w}_{p_i}\cdot\mathbf{x}_i) \geq 1.$$

With a single constraint, solving a quadratic optimization problem is quite simple. As the objective function is convex, when the minimal point (i.e., $\mathbf{w}_u^{(i)}$) does not satisfy the constraint (or the loss function equals zero), the optimal solution must be on the boundary. Then, we can solve the problem easily by using the method of Lagrange multipliers. With a Lagrange multiplier, τ, we can compute the stationary points of the Lagrangian as

$$\frac{\partial}{\partial \mathbf{w}_v}\left[\frac{1}{2}\sum_{v=1}^{K}\left\|\mathbf{w}_v - \mathbf{w}_v^{(i)}\right\|^2 + \tau\left\{1 - (\mathbf{w}_{y_i}\cdot\mathbf{x}_i - \mathbf{w}_u\cdot\mathbf{x}_i)\right\}\right] = \mathbf{0}.$$

Every equation can be rewritten as

$$\begin{aligned}
\mathbf{w}_v^{(i+1)} &= \mathbf{w}_v^{(i)} && (v \neq p_i, y_i)\\
\mathbf{w}_v^{(i+1)} &= \mathbf{w}_v^{(i)} + \tau\mathbf{x}_i && (v = y_i)\\
\mathbf{w}_v^{(i+1)} &= \mathbf{w}_v^{(i)} - \tau\mathbf{x}_i && (v = p_i).
\end{aligned}$$

With further calculations we obtain

$$\tau = \frac{1 - \left(\mathbf{w}_{y_i}^{(i)}\cdot\mathbf{x}_i - \mathbf{w}_u^{(i)}\cdot\mathbf{x}_i\right)}{2\left\|\mathbf{x}_i\right\|^2}.$$

This is the traditional PA algorithm for multiclass classification.

Crammer et al. also discussed that the original PA algorithm could lead to drastic weight-vector changes in the wrong direction, e.g., when the training data included noisy instances. To cope with such problems they proposed two schemes that adopted robust update strategies. These schemes were simply obtained by introducing slack variables to the original constraints, like the soft-margin version of SVMs [2]. This replaced the optimization problem (Eq. (2.1)) by using one of the two formulations

$$\text{(PA} - \text{I)} \quad \arg\min_{\mathbf{w}} \quad \frac{1}{2}\sum_{u=1}^{K}\left\|\mathbf{w}_u - \mathbf{w}_u^{(i)}\right\|^2 + C\xi$$
$$\text{s.t.} \quad (\mathbf{w}_{y_i}\cdot\mathbf{x}_i - \mathbf{w}_{p_i}\cdot\mathbf{x}_i) \geq 1 - \xi\,,\ \xi \geq 0$$
$$\text{(PA} - \text{II)} \quad \arg\min_{\mathbf{w}} \quad \frac{1}{2}\sum_{u=1}^{K}\left\|\mathbf{w}_u - \mathbf{w}_u^{(i)}\right\|^2 + C\xi^2$$
$$\text{s.t.} \quad (\mathbf{w}_{y_i}\cdot\mathbf{x}_i - \mathbf{w}_{p_i}\cdot\mathbf{x}_i) \geq 1 - \xi,$$

where C is a positive parameter that governs the degree of influence of slack variables.

For both optimizations, the larger the C, the smaller the relaxation of the original margin constraint, resulting in a solution that is close to the original problem. These modified versions of the PA optimization problems are, again, easily solved, and the algorithms they obtained, which they called PA-I and PA-II, also preserve the simple closed form updates of the original version.

$$\text{(PA} - \text{I)} \quad \tau = \min\left\{\frac{C}{2}, \frac{1 - \left(\mathbf{w}_{y_i}^{(i)}\cdot\mathbf{x}_i - \mathbf{w}_u^{(i)}\cdot\mathbf{x}_i\right)}{2\left\|\mathbf{x}_i\right\|^2}\right\}$$

$$\text{(PA} - \text{II)} \quad \tau = \frac{1}{2}\left\{\frac{1 - \left(\mathbf{w}_{y_i}^{(i)}\cdot\mathbf{x}_i - \mathbf{w}_u^{(i)}\cdot\mathbf{x}_i\right)}{\left\|\mathbf{x}_i\right\|^2 + \frac{1}{2C}}\right\}$$

We have omited a detailed derivation of these schemes.

3 Closed-form PA Update for Multiclass Loss Function

While relaxing the constraints allowed Crammers et al. [1] to deduce a simple update scheme for the original multiclass PA algorithm, the resulting loss from relaxation was so loose that the updated classifier may still fail to predict the current input correctly immediately after the update. In this sense, the traditional multiclass PA diverges from the principle underlying the PA framework. In order to properly address this issue, we need to provide the exact solution to the unrelaxed optimization problem (2.2), to which the general PA scheme is strictly applied. While another attempt at the unrelaxed version of the multiclass PA problem by Crammer et al. [3] exists, unfortunately their solution is again not exact. MIRA [4] is another online framework that results in an update similar to the one we propose; however it suffers from a loose mistake bound. MIRA also defines a different objective function that does not result in a closed-form update. To the best of our knowledge, the exact solution has somehow escaped the attention of researchers. In this section, we will close this gap and provide the exact solution.

Let us turn our attention back to the optimization problem (2.2). The loss function that we would like to minimize is the hinge loss between class y_i and class u with the largest margin. The minimal value of the loss function, i.e., 0, is only achieved when the true class has a score higher than any other classes, so that the discriminative function would correctly classify the current input. This problem is a quadratic optimization problem; however, compared to the simplified problem (2.4), the number of constraints increases to $K - 1$.

Instead of resorting to a convex optimization solver, we attempt to obtain a closed-form solution to ensure efficiency and exactness.

From the perspective of quadratic optimization analysis, the optimization problem (2.2) has a favorable property. Since the constraints are all linear and the optimization problem obviously has an interior feasible solution, the problem satisfies Slater's condition. The optimal solution thus necessarily and sufficiently satisfies the KKT conditions [5]. This implies that we have only to find the KKT vector to satisfy the KKT conditions, which in turn will emerge as the optimal solution. The KKT conditions for this problem are

$$\frac{\partial}{\partial \mathbf{w}_v}\left[\frac{1}{2}\sum_{u=1}^{K}\left\|\mathbf{w}_u-\mathbf{w}_u^{(i)}\right\|^2+\sum_{u\neq y_i}\tau_u\left\{1-(\mathbf{w}_{y_i}\cdot\mathbf{x}_i-\mathbf{w}_u\cdot\mathbf{x})\right\}\right]=\mathbf{0}\quad(\forall v),$$

$$\tau_u\left\{1-(\mathbf{w}_{y_i}\cdot\mathbf{x}_i-\mathbf{w}_u\cdot\mathbf{x}_i)\right\}=0\quad(\forall u\neq y_i), \tag{3.4}$$

$$\tau_u\geq 0\quad(\forall u\neq y_i),$$

$$1-(\mathbf{w}_{y_i}\cdot\mathbf{x}_i-\mathbf{w}_u\cdot\mathbf{x}_i)\leq 0\quad(\forall u\neq y_i). \tag{3.5}$$

assuming that τ_u makes up the remaining KKT vectors (or dual variables). The first equation explicitly implies the update rule for every $\mathbf{w}_v$:

$$\begin{aligned}\mathbf{w}_v&=\mathbf{w}_v^{(i)}+\sum_{u\neq y_i}\tau_u\mathbf{x}_i &&(v=y_i)\\ \mathbf{w}_v&=\mathbf{w}_v^{(i)}-\tau_v\mathbf{x}_i &&(v\neq y_i),\end{aligned} \tag{3.6}$$

In these equations, τ_v provides magnitudes of data $\mathbf{x}_i$ to add to the weight vector $\mathbf{w}_v$. We then call τ_v the stepsize for each class. This is the update scheme for the PA algorithm for the strict multiclass loss function.

Using this update, the weight vector could potentially be updated for every class, while the traditional PA performs an update against predicted class p_i and answer class y_i. For example, if the score of the correct label is not high enough and the margin to all other classes is smaller than 1, then the scores of most classes must be updated to ensure enough margins. However, if there exists only a small number of competitive classes that distracts the classifier, then the other classes may not take part in the update operation.

As a result, the algorithm chooses in each round a set of classes whose score must be updated. We call such a set the *support class* set. In the next section, we show that the support class set can be determined exactly and efficiently by using a simple enumeration algorithm. We call the algorithm obtained by using the above explained exact solution to optimization problem (2.2) the *Support-Class Passive Aggressive Algorithm* (SPA) in this paper.

To complete the update rule, we then need to determine the values of τ_u. From the second KKT condition, either $\tau_u=0$ or $1-(\mathbf{w}_{y_i}\cdot\mathbf{x}_i-\mathbf{w}_u\cdot\mathbf{x}_i)=0$ must hold true for every class label u except the true class, y_i. The classes for which $\tau_u=0$ can be omitted from the summation in Eq. (3.6) play no role in updating the weight vector. The remaining u, with positive τ_u, must be a support class. A support class must satisfy

$$\mathbf{w}_{y_i}\cdot\mathbf{x}_i-\mathbf{w}_u\cdot\mathbf{x}_i=1. \tag{3.7}$$

This property is crucial to our algorithm. It means that the margin of every support class u must be equal to 1 after the weight vector is updated. We determine the stepsizes for the support class using this condition in the next subsections. We resolve the following two questions in turn: (A) Which class belongs to the set of support classes? (B) What is the stepsize for each support class?

3.1 Stepsizes for support classes

We solve the latter problem in the previous subsection first, i.e., to determine the Lagrange multiplier, τ_v, assuming that the support class is already known. Here we denote the support class as S. Then, the classes for which $u\notin S$ can be omitted from the summation in Eq. (3.6) play no role in updating the weight vector. Therefore, by plugging Eq. (3.6) into the marginal condition (Eq. (3.7)) we obtain

$$\left(\mathbf{w}_{y_i}^{(i)}+\sum_{u\in S}\tau_u\mathbf{x}_i\right)\cdot\mathbf{x}_i-\left(\mathbf{w}_v^{(i)}-\tau_v\mathbf{x}_i\right)\cdot\mathbf{x}_i=1\quad\forall v\in S.$$

Then, the equation can be rewritten as

$$\sum_{u\in S}\tau_u\left\|\mathbf{x}_i\right\|^2+\tau_v\left\|\mathbf{x}_i\right\|^2=1-\left(\mathbf{w}_{y_i}^{(i)}\cdot\mathbf{x}_i-\mathbf{w}_v^{(i)}\cdot\mathbf{x}_i\right). \tag{3.8}$$

As Eq. (3.8) is true for each $v\in S$, we have $|S|$ equations. After summing all $|S|$ equations, we obtain

$$(|S|+1)\sum_{u\in S}\tau_u=\sum_{u\in S}\frac{1-\left(\mathbf{w}_{y_i}^{(i)}\cdot\mathbf{x}_i-\mathbf{w}_u^{(i)}\cdot\mathbf{x}_i\right)}{\left\|\mathbf{x}_i\right\|^2}. \tag{3.9}$$

Thus, the closed form of a Lagrange multiplier is given by substituting Eq. (3.9) into Eq. (3.8),

$$\begin{aligned}\tau_v\;=\;&\frac{1-\left(\mathbf{w}_{y_i}^{(i)}\cdot\mathbf{x}_i-\mathbf{w}_v^{(i)}\cdot\mathbf{x}_i\right)}{\left\|\mathbf{x}_i\right\|^2}\\ &-\sum_{u\in S}\frac{1-\left(\mathbf{w}_{y_i}^{(i)}\cdot\mathbf{x}_i-\mathbf{w}_u^{(i)}\cdot\mathbf{x}_i\right)}{\left\|\mathbf{x}_i\right\|^2(|S|+1)}.\end{aligned}$$

Because every τ_v must be positive, this equation can be rewritten as

$$\tau_v = \frac{1}{\|\mathbf{x}_i\|^2}\left(\ell_v - \frac{1}{|S|+1}\sum_{u\in S}\ell_u\right), \tag{3.10}$$

where we define $\ell_v = \left[1 - \left(\mathbf{w}_{y_i}^{(i)} \cdot \mathbf{x}_i - \mathbf{w}_u^{(i)} \cdot \mathbf{x}_i\right)\right]_+$. Since every ℓ_v can be computed with $\mathbf{w}^{(i)}$ and $(\mathbf{x}_i, y_i)$, we can easily determine the stepsizes and obtain a complete update rule if the support classes are known.

3.2 Determining support classes To determine the set of support classes, we can derive the following theorem, which states a necessary and sufficient condition for being a support class.

THEOREM 3.1. *Assume that $S \neq \varnothing$. Let $\sigma(k)$ be the k-th class when sorted in descending order of ℓ_v. The necessary and sufficient condition for $\sigma(k)$ to be a support class is*

$$\sum_{j=1}^{k-1}\ell_{\sigma(j)} < k\ell_{\sigma(k)}. \tag{3.11}$$

We define $\sum_{j=1}^{0} \bullet$ as zero.

Proof. First, we focus on the KKT conditions in Eq. (3.5) for the non-supported class. We can rewrite this equation by substituting Eq. (3.6) as

$$\sum_{u\in S}\tau_u \|\mathbf{x}_i\|^2 \geq 1 - \left(\mathbf{w}_v^{(i)} \cdot \mathbf{x}_i - \mathbf{w}_{y_i}^{(i)} \cdot \mathbf{x}_i\right) \quad \forall v \notin S,$$

for v is a non-support class and τ_v equals zero. By using the definition of ℓ_v and Eq. (3.9), we can rewrite this equation as

$$\sum_{u\in S}\frac{\ell_u}{|S|+1} \geq \ell_v \quad \forall v \notin S. \tag{3.12}$$

For a support class, on the other hand, as the following holds true for Eq. (3.10):

$$\sum_{u\in S}\frac{\ell_u}{|S|+1} < \ell_v, \ \ \forall v \in S, \tag{3.13}$$

their Lagrange multiplier must be positive. From Eqs. (3.12) and (3.13), the support classes are the set of classes for which the value of ℓ_u is larger than some given value. The next step for determining the support class is then straightforward.

Let $\sigma(k)$ be the k-th class when sorted in descending order of ℓ_v.

(Sufficiency) If it is assumed that $\ell_{\sigma(k)}$ satisfies Eq.(3.11), then

$$\begin{aligned}\sum_{j=1}^{k-1}\ell_{\sigma(j)} < k\ell_{\sigma(k)} \quad &\Rightarrow \quad \sum_{j=1}^{|S|}\ell_{\sigma(j)} < (|S|+1)\ell_{\sigma(k)}\\ &\Leftrightarrow \quad \sum_{j=1}^{|S|}\frac{\ell_{\sigma(j)}}{|S|+1} < \ell_{\sigma(k)}\end{aligned}$$

The second inequality follows because ℓ_u is in descending order. This means k corresponds to a label of some support classes (Eq. (3.13)).

(Necessity) If it is assumed that ℓ_k does not satisfy Eq. (3.11), then

$$\begin{aligned}\sum_{j=1}^{k-1}\ell_{\sigma(j)} \geq k\ell_{\sigma(k)} \quad &\Leftrightarrow \quad \sum_{j=1}^{k}\ell_{\sigma(j)} \geq (k+1)\ell_{\sigma(k)}\\ &\Rightarrow \quad \sum_{j=1}^{k}\frac{\ell_{\sigma(j)}}{k+1} \geq \ell_{\sigma(k)} \geq \ell_{\sigma(k+1)}\end{aligned}$$

Therefore, any j larger than k does not satisfy Eq. (3.13). this means $|S| < k$ and thus k does not correspond to a label of a support class.

Using this theorem, we can immediately and efficiently determine the support classes. We may simply sort classes according to their ℓ_u in descending order, and the top J classes in the sorted list are the support classes, with J being the largest k that satisfies the theorem. This allows us to algorithmically determine the support classes. Thus, the SPA algorithm can finally be listed in Tab. 1. To summarize the update algorithm, we first determine the set of support classes by using the stated algorithm, then obtain the value of τ_u for $u \in S$ with Eq. (3.10), and finally update the weight vectors with Eq. (3.6). Binary classification can be applied to this algorithm. However, in this case, the SPA algorithm becomes identical to PA algorithm as their optimization problems are also identical.

3.3 Extension of PA-I and PA-II Following similar arguments, we can also derive new algorithms based on PA-I and PA-II with support classes. We call these SPA-I and SPA-II. We have omited the detailed derivation because it can be obtained by directly following the derivation of SPA. The optimization problems for both algorithms compared with the SPA algorithm and PA algorithms are defined in Tab. 2. Also the update rule of their optimal solutions is the same as that in Eq.(3.6). To obtain the executable algorithms, replace the SPA's stepsize setting (A) and conditions for searching the support classes (B) with those of SPA-I and SPA-II in Tab. 2.

	PA	SPA
Optimization	$\min_{\mathbf{w}} \frac{1}{2}\sum_{u=1}^{K}\left\|\mathbf{w}_u - \mathbf{w}_u^{(i)}\right\|^2$ s.t. $\mathbf{w}_{y_i}\cdot\mathbf{x}_i - \mathbf{w}_{p_i}\cdot\mathbf{x}_i \geq 1,$	$\min_{\mathbf{w}} \frac{1}{2}\sum_{u=1}^{K}\left\|\mathbf{w}_u - \mathbf{w}_u^{(i)}\right\|^2$ s.t. $\forall u \neq y_i, \mathbf{w}_{y_i}\cdot\mathbf{x}_i - \mathbf{w}_u\cdot\mathbf{x}_i \geq 1$
Support class (A)	$k = 1$	$\sum_{j=1}^{k-1}\ell_{\sigma(j)} < k\ell_{\sigma(k)}$
Stepsize (B)	$\tau_v := \frac{[\ell_v]_+}{2\|\mathbf{x}_i\|^2}$	$\tau_v := \|\mathbf{x}_i\|^{-2}\left[\ell_v - \sum_{u\in S}\frac{\ell_u}{\|S\|+1}\right]_+$
	PA-I	**SPA-I**
Optimization	$\min_{\mathbf{w}} \frac{1}{2}\sum_{u=1}^{K}\left\|\mathbf{w}_u - \mathbf{w}_u^{(i)}\right\|^2 + C\xi$ s.t. $\mathbf{w}_{y_i}\cdot\mathbf{x}_i - \mathbf{w}_{p_i}\cdot\mathbf{x}_i \geq 1-\xi, \xi \geq 0$	$\min_{\mathbf{w}} \frac{1}{2}\sum_{u=1}^{K}\left\|\mathbf{w}_u - \mathbf{w}_u^{(i)}\right\|^2 + C\xi$ s.t. $\forall u \neq y_i, \mathbf{w}_{y_i}\cdot\mathbf{x}_i - \mathbf{w}_u\cdot\mathbf{x}_i \geq 1-\xi, \xi \geq 0$
Support class (A)	$k = 1$	$\sum_{j=1}^{k-1}\ell_{\sigma(j)} < \min\left\{\frac{(k+1)C}{\|\mathbf{x}_i\|^2} - \ell_{\sigma(k)}, k\ell_{\sigma(k)}\right\}$
Stepsize (B)	$\tau_v := \min\left(\frac{C}{2}, \frac{[\ell_v]_+}{2\|\mathbf{x}_i\|^2}\right)$	$\tau_v := \|\mathbf{x}_i\|^{-2}\left[\ell_v - \max\left\{\sum_{u\in S}\frac{\ell_u}{\|S\|} + \frac{C}{\|S\|}\|\mathbf{x}_i\|^2, \sum_{u\in S}\frac{\ell_u}{\|S\|+1}\right\}\right]_+$
	PA-II	**SPA-II**
Optimization	$\min_{\mathbf{w}} \frac{1}{2}\sum_{u=1}^{K}\left\|\mathbf{w}_u - \mathbf{w}_u^{(i)}\right\|^2 + C\xi^2$ s.t. $\mathbf{w}_{y_i}\cdot\mathbf{x}_i - \mathbf{w}_{p_i}\cdot\mathbf{x}_i \geq 1-\xi$	$\min_{\mathbf{w}} \frac{1}{2}\sum_{u=1}^{K}\left\|\mathbf{w}_u - \mathbf{w}_u^{(i)}\right\|^2 + C\xi^2$ s.t. $\forall u \neq y_i, \mathbf{w}_{y_i}\cdot\mathbf{x}_i - \mathbf{w}_u\cdot\mathbf{x}_i \geq 1-\xi$
Support class (A)	$k = 1$	$\sum_{j=1}^{k-1}\ell_{\sigma(j)} < \left(\frac{k\|\mathbf{x}_i\|^2 + \frac{k-1}{2C}}{\|\mathbf{x}_i\|^2 + \frac{1}{2C}}\right)\ell_{\sigma(k)}$
Stepsize (B)	$\tau_v := \frac{1}{2}\left(\frac{[\ell_v]_+}{\|\mathbf{x}_i\|^2 + \frac{1}{2C}}\right)$	$\tau_v := \|\mathbf{x}_i\|^{-2}\left[\ell_v - \left(\frac{\|\mathbf{x}_i\|^2 + \frac{1}{2C}}{(\|S\|+1)\|\mathbf{x}_i\|^2 + \frac{1}{2C}\|S\|}\right)\sum_{u\in S}\ell_u\right]_+$

Table 2: Summary of PA and SPA algorithms

```
procedure $SPA$
  foreach $i = 1, \cdots, N$ do
    while $v \in Y \setminus \{y_i\}$ do
      $\ell_v := \left[1 - \left(\mathbf{w}_v^{(i)}\cdot\mathbf{x}_i - \mathbf{w}_{y_i}^{(i)}\cdot\mathbf{x}_i\right)\right]_+$
    end while
    Let $\sigma$ be a ranking function
    $\sigma : \{1, \cdots, K-1\} \to Y\setminus\{y_i\}$ s.t. $j \leq j' \to \ell_{\sigma(j)} \geq \ell_{\sigma(j')}$
    $S := \varnothing$
    $k := 1$
    while $\sum_{j=1}^{k-1}\ell_{\sigma(j)} < k\ell_{\sigma(k)}$ do    $\cdots$(A)
      $S := S \cup \{\sigma(k)\}$
      $k := k + 1$
    end while
    while $v \in S$ do
      $\tau_v := \|\mathbf{x}_i\|^{-2}\left[\ell_v - \sum_{u\in S}\frac{\ell_u}{|S|+1}\right]_+$    $\cdots$(B)
      $\mathbf{w}_{y_i}^{(i+1)} := \mathbf{w}_{y_i}^{(i)} + \tau_v\mathbf{x}_i$
      $\mathbf{w}_v^{(i+1)} := \mathbf{w}_v^{(i)} - \tau_v\mathbf{x}_i$
    end while
  end foreach
```

Table 1: Support Class Passive-Aggressive Algorithm

Note that the support classes for SPA-I and II can easily be identified because evaluating the support condition of a class is possible by simply looking up classes with larger losses than that of itself.

4 Bound Analysis

Because the SPA algorithm strictly follows the principle underlying PA, most bound analyses in [1] can be proven analogically. We proved that a mistake bound was stricter than the former algorithm. We first prove a loss bound for the SPA algorithm in the separable case. This type of bound was previously presented by Herbster [6] and is analogous to the classic mistake bound for the perceptron algorithm due to [7]. Without loss of generality, we can assume that $\mathbf{u}$ is scaled such that $\min_{v\in Y}(\mathbf{u}_{y_i}\cdot\mathbf{x}_i - \mathbf{u}_v\cdot\mathbf{x}_i) = 1$ and therefore $\mathbf{u}$ attains a loss of zero on all T examples in the sequence. With the vector, $\mathbf{u}$, at our disposal, we prove the following bound on the cumulative squared loss of PA. As a result, the mistake bound of the SPA is stricter than that of PA in [1]. However, we have not proven the bounds of SPA-I and SPA-II.

THEOREM 4.1. *Assume a separable case that there ex-*

ists $\mathbf{u}$ *such that loss* $\ell(\mathbf{u};(\mathbf{x}_i,y_i))$ *is zero through the whole data and* $\|\mathbf{x}_i\| \le R$ *for all* i*, The number of misclassifications, #miss, made by the SPA algorithm is upper bounded as in the following representation.*

$$\#miss < 2R^2\sum_{v=1}^{K}\|\mathbf{u}_v\|^2.$$

Proof. First, we define $\Delta^{(i)}$ as

$$\Delta^{(i)} \triangleq \sum_{v=1}^{K}\left\|\mathbf{w}_v^{(i)}-\mathbf{u}_v\right\|^2-\sum_{v=1}^{K}\left\|\mathbf{w}_v^{(i+1)}-\mathbf{u}_v\right\|^2$$

Here, the upper index denotes the number of the instance along which the discriminative function is updated. Then,

$$\Delta^{(i)} = \sum_{v=1}^{K}\left\|\mathbf{w}_v^{(i)}-\mathbf{u}_v\right\|^2-\sum_{v=1}^{K}\left\|\mathbf{w}_v^{(i)}-\mathbf{u}_v-\tau_v^{(i)}\mathbf{x}_i\right\|^2.$$

Here, we define $\tau_u^{(i)}$ as the stepsize of class u for the instance $(\mathbf{x}_i,y_i)$ at $u\neq y_i$ and $\tau_{y_i}^{(i)}$ as $-\sum_{u\neq y_i}\tau_u^{(i)}$. Then,

(4.14)
$$\begin{aligned}\Delta^{(i)} &= 2\sum_{v=1}^{K}\left(\mathbf{w}_v^{(i)}-\mathbf{u}_v\right)\cdot\tau_v^{(i)}\mathbf{x}_i-\sum_{v=1}^{K}\left\|\tau_v^{(i)}\mathbf{x}_i\right\|^2\\ &= 2\sum_{v=1}^{K}\tau_v^{(i)}\left(\mathbf{w}_v^{(i)}-\mathbf{u}_v\right)\cdot\mathbf{x}_i-\|\mathbf{x}_i\|^2\sum_{v=1}^{K}\left(\tau_v^{(i)}\right)^2\\ &= 2\sum_{v\neq y_i}\tau_v^{(i)}\left(\left(\mathbf{w}_v^{(i)}-\mathbf{w}_{y_i}^{(i)}\right)\cdot\mathbf{x}_i-\left(\mathbf{u}_v^{(i)}-\mathbf{u}_{y_i}^{(i)}\right)\cdot\mathbf{x}_i\right)\\ &\qquad-\|\mathbf{x}_i\|^2\left(\sum_{v\neq y_i}\left(\tau_v^{(i)}\right)^2+\left(\sum_{v\neq y_i}\tau_v^{(i)}\right)^2\right)\\ &= -\sum_{v\neq y_i}\tau_v^{(i)}\left(-2\ell_v^{(i)}+2\ell_v^{*(i)}+\|\mathbf{x}_i\|^2\tau_v^{(i)}\right)\\ &\qquad-\|\mathbf{x}_i\|^2\left(\sum_{v\neq y_i}\tau_v^{(i)}\right)^2,\end{aligned}$$

where $\ell_v^{(i)} = \left[1-\left(\mathbf{w}_{y_i}^{(i)}\cdot\mathbf{x}_i-\mathbf{w}_v^{(i)}\cdot\mathbf{x}_i\right)\right]_+$ and $\ell_v^{*(i)} = \left[1-\left(\mathbf{u}_{y_i}\cdot\mathbf{x}_i-\mathbf{u}_v\cdot\mathbf{x}_i\right)\right]_+$. Let $S^{(i)}$ be the set of support classes in the i-th round of learning. For all $v\in S^{(i)}$ we can see from Eq. (3.10) that

$$\|\mathbf{x}_i\|^2\tau_v^{(i)} = \ell_v^{(i)}-\frac{1}{\left|S^{(i)}\right|+1}\sum_{u\in S^{(i)}}\ell_u^{(i)}$$

and

$$\|\mathbf{x}_i\|^2\sum_{v\in S^{(i)}}\tau_v^{(i)} = \frac{1}{\left|S^{(i)}\right|+1}\sum_{u\in S^{(i)}}\ell_u^{(i)}$$

For the last representation of $\Delta^{(i)}$ in Eq.(4.14) can ignore all $v\notin S^{(i)}$ in the sum, we can see by substituting these equations that it becomes

(4.15)
$$\begin{aligned}\Delta^{(i)} = \frac{1}{\|\mathbf{x}_i\|^2}\sum_{v\in S^{(i)}}&\left(\ell_v^{(i)}-\frac{1}{\left|S^{(i)}\right|+1}\sum_{u\in S^{(i)}}\ell_u^{(i)}\right)\\ &\left(\ell_v^{(i)}+\frac{1}{\left|S^{(i)}\right|+1}\sum_{u\in S^{(i)}}\ell_u^{(i)}-2\ell_v^{*(i)}\right)\\ &-\frac{1}{\|\mathbf{x}_i\|^2}\frac{1}{\left(\left|S^{(i)}\right|+1\right)^2}\left(\sum_{u\in S^{(i)}}\ell_u^{(i)}\right)^2.\end{aligned}$$

From the assumption, we see that $\ell_v^{*(i)}=0$ for all i and $v\neq y_i$. Then,

$$\begin{aligned}\Delta^{(i)} &= \frac{1}{\|\mathbf{x}_i\|^2}\sum_{v\in S^{(i)}}\left(\ell_v^{(i)}\right)^2\\ &\qquad-\frac{1}{\|\mathbf{x}_i\|^2}\frac{|S^{(i)}|}{\left(|S^{(i)}|+1\right)^2}\left(\sum_{u\in S^{(i)}}\ell_u^{(i)}\right)^2\\ &\qquad-\frac{1}{\|\mathbf{x}_i\|^2}\frac{1}{\left(|S^{(i)}|+1\right)^2}\left(\sum_{u\in S^{(i)}}\ell_u^{(i)}\right)^2\\ &= \frac{\|\mathbf{x}_i\|^{-2}}{\left|S^{(i)}\right|+1}\sum_{v\in S^{(i)}}\left(\ell_v^{(i)}\right)^2\\ &\quad+\frac{\|\mathbf{x}_i\|^{-2}}{\left|S^{(i)}\right|+1}\left(\left|S^{(i)}\right|\sum_{v\in S^{(i)}}\left(\ell_v^{(i)}\right)^2-\left(\sum_{u\in S^{(i)}}\ell_u^{(i)}\right)^2\right)\\ &\geq \frac{\|\mathbf{x}_i\|^{-2}}{\left|S^{(i)}\right|+1}\sum_{v\in S^{(i)}}\left(\ell_v^{(i)}\right)^2.\end{aligned}$$

The last inequality follows as

$$\begin{aligned}&n\sum_{i=1}^{n}a_i^2-\left(\sum_{i=1}^{n}a_i\right)^2\\ &= (n-1)\sum_{i=1}^{n}a_i^2-2\sum_{i=1}^{n}\sum_{j=i+1}^{n}a_ia_j\\ &= \sum_{i=1}^{n}\sum_{j=i+1}^{n}(a_i-a_j)^2\geq 0.\end{aligned}$$

We next use the following lemma to further evaluate $\Delta^{(i)}$.

LEMMA 4.1. *Assuming* $\mathbf{w}^{(i)}$ *was updated using the SPA and the previous definition of* $\ell_v^{(i)}$,

$$\ell_v^{(i)} \geq \frac{1}{2}\,\ell^{(i)} \qquad (\forall v \in S^{(i)})$$

where $\ell^{(i)} = \ell(\mathbf{w}^{(i)}, (\mathbf{x}_i, y_i))$.

Proof. Because $\ell_{\sigma(1)}^{(i)} = \ell^{(i)}$, this is sufficiently holds if $\sum_{j=1}^{k} \ell_{\sigma(j)}^{(i)} \geq \frac{(k+1)}{2}\ell^{(i)}$ for any $\sigma(k) \in S^{(i)}$. Here, $\sigma(k)$ is the same notation as above. Thus, we prove these inequalities. If $k = 1$, that obviously holds. Else, if $\sum_{j=1}^{k} \ell_{\sigma(j)} \geq \frac{(k+1)}{2}\ell^{(i)}$ and $\sigma(k+1) \in S^{(i)}$, then $\sum_{j=1}^{k+1} \ell_{\sigma(j)} = \sum_{j=1}^{k} \ell_{\sigma(j)} + \ell_{\sigma(k+1)} > \sum_{j=1}^{k} \ell_{\sigma(j)} + \frac{1}{k+1}\sum_{j=1}^{k} \ell_{\sigma(j)} \geq (1 + \frac{1}{k+1})\frac{(k+1)}{2}\ell^{(i)} = \frac{\{(k+1)+1\}}{2}\ell^{(i)}$. Therefore, the inequalities hold by mathematical induction.

From this lemma, we can see that the square of every class can be upper bounded by 1/4 when the classifier misclassifies the instance. Then, $\sum_{v\in S^{(i)}} (\ell_v^{(i)})^2 \geq \frac{|S^{(i)}|+3}{4}$ by taking into account that $\ell^{(i)} \geq 1$. Then, we continue to evaluate the lower bound of $\Delta^{(i)}$ as

$$\begin{aligned}
\Delta^{(i)} &\geq \frac{1}{\|\mathbf{x}_i\|^2}\frac{1}{\left|S^{(i)}\right|+1}\sum_{v\in S^{(i)}} \left(\ell_v^{(i)}\right)^2 \\
&\geq \frac{1}{\|\mathbf{x}_i\|^2}\frac{1}{\left|S^{(i)}\right|+1}\frac{\left|S^{(i)}\right|+3}{4}\left[\!\left[h^{(i)}(\mathbf{x}_i) \neq y_i \right]\!\right] \\
&\geq \frac{1}{2R^2}\left[\!\left[h^{(i)}(\mathbf{x}_i) \neq y_i \right]\!\right].
\end{aligned}$$

However, the upper bound of the sum of $\Delta^{(i)}$ can be obtained as

$$\begin{aligned}
\sum_{i=1}^{N} \Delta^{(i)} &= \sum_{i=1}^{N}\sum_{v=1}^{K}\left(\left\|\mathbf{w}_v^{(i)} - \mathbf{u}_v\right\|^2 - \left\|\mathbf{w}_v^{(i+1)} - \mathbf{u}_v\right\|^2\right) \\
&= \sum_{v=1}^{K}\left(\left\|\mathbf{w}_v^{(1)} - \mathbf{u}_v\right\|^2 - \left\|\mathbf{w}_v^{(N)} - \mathbf{u}_v\right\|^2\right) \\
&\leq \sum_{v=1}^{K}\|\mathbf{u}_v\|^2
\end{aligned}$$

Thus, their inequality implies

$$\begin{aligned}
&\sum_{i=1}^{N}\frac{1}{2R^2}\left[\!\left[h^{(i)}(\mathbf{x}_i) \neq y_i \right]\!\right] \leq \sum_{i=1}^{N}\Delta^{(i)} \leq \sum_{v=1}^{K}\|\mathbf{u}_v\|^2 \\
\Leftrightarrow\quad &\#miss \leq 2R^2\sum_{v=1}^{K}\|\mathbf{u}_v\|^2
\end{aligned}$$

Next, we do not have a separability assumption. Then, the loss bound is found to be the following theorem.

THEOREM 4.2. *In the case that* $\|\mathbf{x}_i\| = R$ *for all* i, *for any* $\mathbf{u} = \{\mathbf{u}_v\}_{v\in Y}$, *the loss is upper bounded by*

$$\sum_{i=1}^{N}\ell^{(i)} < 2\left(R\sqrt{\sum_{v=1}^{K}\|\mathbf{u}_v\|^2} + \sqrt{2\sum_{i=1}^{N}(\ell^{*(i)})^2}\right)^2,$$

where $\ell^{(i)} = \ell(\mathbf{w}^{(i)}, (\mathbf{x}_i, y_i))$ *and* $\ell^{*(i)} = \ell(\mathbf{u}, (\mathbf{x}_i, y_i))$.

Proof. From the proof of the latter theorem, we can see that

$$\begin{aligned}
\Delta^{(i)} = &\frac{1}{\|\mathbf{x}_i\|^2}\sum_{v\in S^{(i)}}\left(\ell_v^{(i)} - \frac{1}{\left|S^{(i)}\right|+1}\sum_{u\in S^{(i)}}\ell_u^{(i)}\right) \\
&\left(\ell_v^{(i)} + \frac{1}{\left|S^{(i)}\right|+1}\sum_{u\in S^{(i)}}\ell_u^{(i)} - 2\ell_v^{*(i)}\right) \\
&-\frac{1}{\|\mathbf{x}_i\|^2}\frac{1}{\left(\left|S^{(i)}\right|+1\right)^2}\left(\sum_{u\in S^{(i)}}\ell_u^{(i)}\right)^2 \\
\geq &\frac{\|\mathbf{x}_i\|^{-2}}{\left|S^{(i)}\right|+1}\sum_{v\in S^{(i)}}\left(\ell_v^{(i)}\right)^2 - 2\sum_{v\in S^{(i)}}\ell_v^{*(i)}\tau_v^{(i)} \\
\geq &\frac{1}{\|\mathbf{x}_i\|^2}\frac{1}{\left|S^{(i)}\right|+1}\frac{\left|S^{(i)}\right|+3}{4}(\ell^{(i)})^2 \\
- &\ 2\ell^{*(i)}\frac{1}{\|\mathbf{x}_i\|^2}\left(\frac{1}{\left|S^{(i)}\right|+1}\sum_{u\in S^{(i)}}\ell_u^{(i)}\right) \\
= &\left(\frac{\left|S^{(i)}\right|+3}{4\left|S^{(i)}\right|+4}\ell^{(i)} - \ell^{*(i)}\right)\frac{\ell^{(i)}}{\|\mathbf{x}_i\|^2}.
\end{aligned}$$

Then, we obtain

$$\begin{aligned}
&R^2\sum_{v=1}^{K}\|\mathbf{u}_v\|^2 \geq R^2\sum_{i=1}^{N}\Delta^{(i)} \geq \frac{L^2}{2} - LL^* \\
\Rightarrow\quad &2R^2\sum_{v=1}^{K}\|\mathbf{u}_v\|^2 + (L^*)^2 \geq (L - L^*)^2,
\end{aligned}$$

where $L = \sqrt{\sum_{i=1}^{N}(\ell^{(i)})^2}$ and $L^* = \sqrt{\sum_{i=1}^{N}(\ell^{*(i)})^2}$. For this equation to hold, L must be less than $L^* + \sqrt{2R^2\sum_{v=1}^{K}\|\mathbf{u}_v\|^2 + (L^*)^2}$. Then, the equation of the theorem taking into account that $\sqrt{2R^2\sum_{v=1}^{K}\|\mathbf{u}_v\|^2 + (L^*)^2} \leq \sqrt{2R^2\sum_{v=1}^{K}\|\mathbf{u}_v\|^2} + L^*$.

	# of classes	# of data	# of features	classes
sb-8-1	8	4,000	16,282	autos, christian, crypt, hockey, mac, mideast, space, windows
sb-7-1	7	3,500	13,997	atheism, autos, crypt, forsale, hockey, space, windows
ob-8-1	8	4,000	13,890	autos, baseball, graphics, mac, med, motor, politics, space
ob-7-1	7	3,500	12,878	autos, baseball, electronics, med, motor, politics, space
ol-8-1	8	2,388	9,971	autos, baseball, graphics, mac, med, motor, politics, space
ol-7-1	7	2,586	9,605	autos, baseball, electronics, med, motor, politics, space
USPS	10	7,291	256	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Reuters	20	7,800	34,488	acq, alum, cocoa, coffee, copper, cpi, crude, earn, gnp, gold, grain, interest, jobs, money-fx, money-supply, reserves, rubber, ship, sugar, trade
News20	20	15,935	60,345	comp.graphics, comp.os.ms-windows.misc, comp.sys.ibm.pc.hardware, comp.sys.mac.hardware, comp.windows.x, rec.autos, rec.motorcycles, rec.sport.baseball, rec.sport.hockey, sci.crypt, sci.electronics, sci.med, sci.space, misc.forsale, talk.politics.misc, talk.politics.guns, talk.politics.mideast, talk.religion.misc, alt.atheism, soc.religion.christian

Table 3: Test set specifications

5 Experiments

5.1 Dataset specification We evaluated the performance of our SPA, SPA-I, and SPA-II algorithms with the 20 Newsgroups corpus in the Machine Learning Group Datasets,[1] the USPS data set in the LIBSVM data sets,[2], and the Reuters-21578 corpus.[3]

The 20 Newsgroups corpus consists of around 20,000 documents and each document is categorized into one of 20 different groups. We used six subsets of the 20 Newsgroups corpus in the Machine Learning Group Datasets. The first letter 'o' denotes "overlap" and 's' denotes "separate." The second letter 'b' denotes "balanced" and 'l' denotes "large." The middle number is the number of classes. The features were given as a bag-of-words from each document, i.e., each feature corresponded to a word frequency in the document, and therefore, the number of features was extremely high. The USPS data set is an image database for handwritten digit recognition [8]. "Reuters" consists of news articles. We preprocessed the corpus and obtained a dataset for a 20-class classification problem. The specifications for data sets are provided in Tab. 3.

5.2 Comparison of online learners Online learning assumes that when making a prediction, a learner immediately receives feedback on the correct label. This implies that, unlike in batch learning where training and testing phases are separate, we may evaluate the successes or failures of a learner as it receives feedback during the course of online learning. Thus, in an online setting, the performance of a learner is typically evaluated using the accumulated loss and the number of misclassifications as learning progresses. We plot the number of misclassifications, the number of updates, the accumulated loss, $\sum_{i=1}^{N} \ell(\mathbf{w}^{(i)}; (\mathbf{x}_i, y_i))$, and success rate $1 - \#miss/N$ every 200 instances for the presented SPA algorithms and existing PA algorithms. We used the USPS dataset for these experiments.

Before we go on to present the results, we would like to clarify the hyperparameter setting. If we were to assume a truly online setting, we would have no validation set to tune the hyperparameters. With this in mind, instead of tuning hyperparameters, we simply used the value indicated in [1] and fixed the hyperparameters to 0.001 in all algorithms. We evaluated the classifiers on a Xeon 3.0-GHz server with a 32 GB memory.

Fig. 1 plots the number of mistakes and updates as the learners receive inputs. We observed that the number of misses for the SPA classifiers were much lower than the PA classifiers and perceptron. The number of updates were eventually lower as well, indicating a good convergence properties. In a truly online setting, the SPA classifiers achieved better performance than other online algorithms.

Fig. 2 plots the values of accumulative loss and success rate every 200 rounds. In these results, our algorithms again demonstrated better performance. The SPA algorithm forces weight vectors to have more aggressive updates as the classifier suffers positive loss. Although intensive change in the weight vector would intuitively lead to the fractuation of scores and possibly more losses, the lower accumulated loss of the SPA algorithms indicates that their learning is efficient.

Let us next examine the accuracy of the test dataset using a 10-fold cross validation in various tasks. In these experiments, each data set was also randomly divided into 10 partitions for 10-fold cross validation and one partition was used for determining the hyperparameter

[1]http://mlg.ucd.ie/datasets

[2]http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

[3]http://www.daviddlewis.com/resources/testcollections/reuters21578/

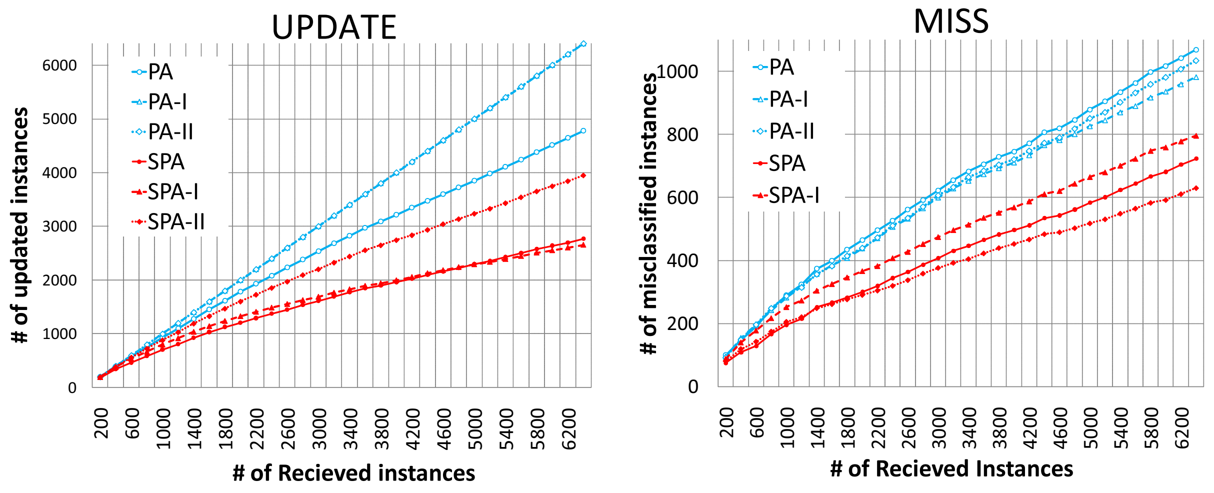

Figure 1: Updates and missclassifications number progression

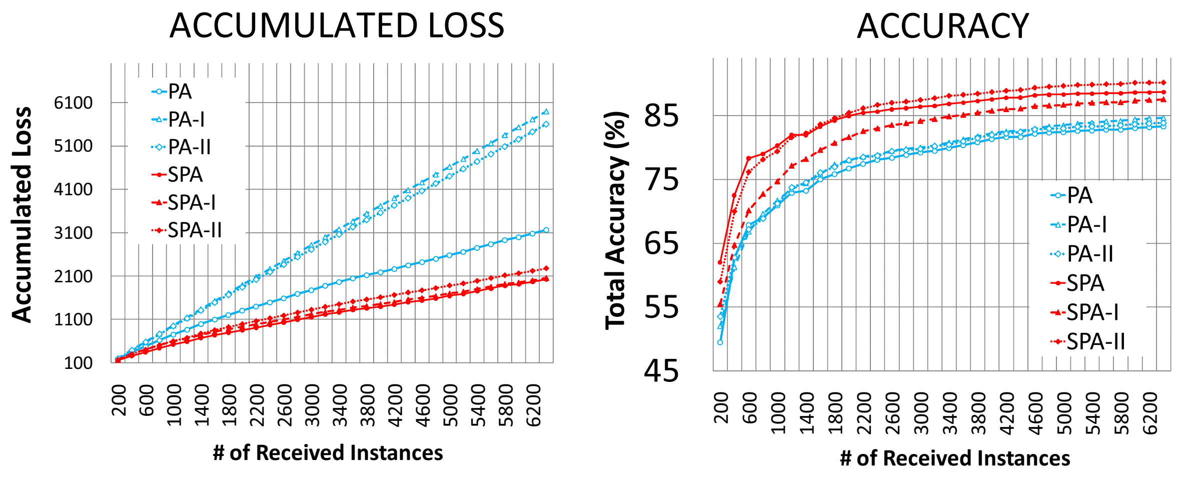

Figure 2: Accuracy and loss progression

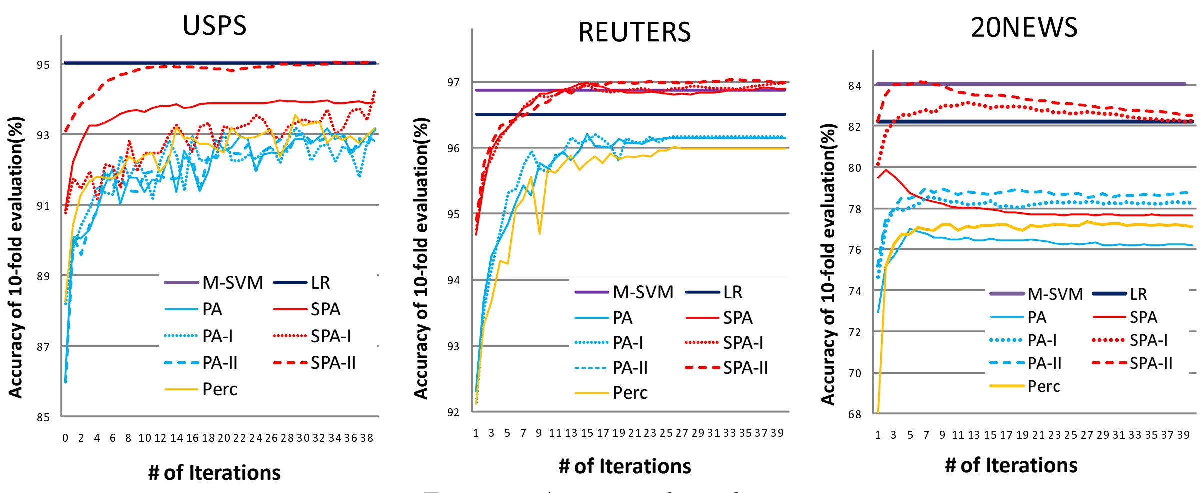

Figure 3: Accuracy through iterations

	Perc	PA	SPA	PA-I	SPA-I	PA-II	SPA-II
sb-8-1	13.08	10.35	5.28	10.48	5.20	10.43	**4.38**
sb-7-1	11.43	10.89	5.46	8.57	5.23	9.34	**4.60**
ol-8-1	16.20	18.02	9.98	16.01	7.23	15.51	**6.81**
ol-7-1	16.20	17.92	9.13	14.41	7.33	14.49	**7.29**
ob-8-1	17.08	13.55	7.20	12.55	6.88	12.50	**6.15**
ob-7-1	13.69	13.69	7.37	12.14	6.60	13.29	**6.40**
reut20	7.88	7.69	5.32	7.44	5.46	7.95	**5.29**
USPS	11.74	14.02	9.15	11.05	8.68	11.74	**6.89**
News20	31.96	27.03	20.50	24.62	18.85	25.82	**17.23**

Table 4: Test set error rates (%) according to online situation for 10-fold cross validation

	PA	PA-I	PA-II	SPA	SPA-I	SPA-II	Perc	LR	M-SVM
USPS	5.5	5.5	5.5	5.9	5.9	5.9	1.5	53.6	68.5
Reuter	6.2	6.2	6.2	6.4	6.4	6.4	1.0	92.5	34.3
News20	15.7	15.9	15.8	16.8	17.1	17.2	7.9	405.6	75.7

Table 5: Runtime specification (sec)

(C). Tab. 4 shows the error rates of online situation for each dataset. Our algorithms were significantly more accurate than the existing PA algorithms.

5.3 Comparison of online and batch learners

Unlike batch classifiers, in which optimization takes place for an entire data set, online classifiers are unable to exploit all the data as they optimize themselves one by one. In terms of pure performance, this puts them at a competitive disadvantage compared to batch classifiers. We therefore generally expect online classifiers to perform worse than batch classifiers.

To determine how close our proposed classifiers perform in comparison to batch classifiers, we will apply an online-to-batch conversion to online classifiers. An online-to-batch conversion is typically done as follows: instead of applying an online classifier once to the whole data, we apply them iteratively over the data until performance no longer increases. As many iterations may result over a large dataset in high computational cost, an online algorithm could expend more computational resources than batch algorithms. Therefore, a required property of a classifier is to achieve reasonable accuracy with fewer iterations. We let every online algorithm iterate through the dataset 40 times. We measured the time required for learning with every algorithm, the results of which are shown in Tab. 5. Thus, as previously noted, online learners including SPA algorithms learn faster than batch learners for these datasets.

We compared the accuracy of our algorithms with two batch-learning algorithms, a multiclass logistic regression [?] (LR, also known as a log-linear model or maximum entropy model) and a multiclass support vector machine (M-SVM) [2, 9]. The hyperparameters for online learners, a Gaussian prior for "LR" and a soft margin parameter C for "M-SVM" were determined in the same way using the validation set.

Tab. 6 and Fig. 3 demonstrate the error rates of our SPA algorithms (SPA, SPA-I and SPA-II), the original PA algorithms (PA, PA-I and PA-II), and perceptron(Perc) [10, 7, 11], as well as the accuracies of LR and M-SVM. The figure inside the parentheses indicates the number of iterations.

As seen in the table, our SPA-II algorithm was the most accurate of the online learning methods. We also observed that PA-II was more accurate than PA and PA-I, and SPA-II was more accurate than SPA and SPA-I. The batch algorithms were more accurate than the dataset of the online algorithms for the USPS, but SPA-I and SPA-II outperformed the batch methods for the Reuters data set. In the 20 Newsgroups dataset, although the accuracy of PA and SPA peak in the first iteration and quickly deteriorate as the iteration continues, the other algorithms which adopt a soft margin demonstrate stable accuracies and outperform the original algorithms respectively. This indicates that this dataset includes many noisy instances.

The proposed algorithms acquired competitive peak performances compared to batch learners, taking less than a fourth as long as them. Furthermore, while other online learners tended to be unstable across different validation sets, the proposed SPA algorithms stably improved their accuracies. That implies the SPA algorithm has potential for a standard learner.

6 Related Work

Updating via multiple classes can be seen in other algorithms such as [12]; however, their attempts have been aimed at obtaining a large margin and not online learning. Apart from MIRA[4] and [3] that were previously mentioned in Section 3, a confidence weighted linear classifier[13],[14] was based on the PA framework and their method was extended to multiclass classification by [15]. While in the separable case, the binary CW classifier has a lower bound and mistake bound, the bounds in multiclass classification do not exist. We focused on a rigid extension of the framework of the PA algorithm to multiclass classification. Our extetension encompassed the mistake and loss bounds as shown in Section 4.

7 Conclusion

We presented three new PA algorithms for a strict multiclass loss function with efficient update algorithms deduced within the PA framework. The resulting update ensures that the current training instance to be correctly classified with at least a margin of one. This is accomplished by sorting the classes by their losses, then finding the support classes with a loss above a threshold, a figure calculated in closed form. Once the support

	PA	PA-I	PA-II	SPA	SPA-I	SPA-II	Perc	LR	M-SVM
USPS	6.82(34)	6.83(34)	6.82(40)	6.02(34)	5.73(39)	**4.95**(35)	6.45(30)	4.98	**4.83**
Reuters	3.80(14)	3.80(14)	3.81(15)	3.01(14)	3.03(39)	**2.96**(32)	3.99(25)	3.50	**3.18**
News20	22.98(4)	21.49(7)	21.02(8)	20.01(1)	16.84(5)	**15.83**(11)	22.63(26)	17.79	**15.94**

Table 6: Dataset peak error rate (%) by algorithm

classes are found, the update stepsize for each class and the weights are moved from the support classes to the correct class.

In the multiclass classification, a learner has to choose the class which requires an update. As a prediction depends on scores of all classes, the choice must be relative to the scores of all classes. Since this has not been done in previous studies, a principle method for doing this was not known. The proposed algorithm provides a principle method for making this choice that considers the relative scores of all classes. This method consists of the condition for a class to be a support class and the algorithm that efficiently evaluates this condition. Identifying support classes is not easy because not all classes with margin less than one are support classes. We can see that all classes with a margin less than one do not have to be updated. Our method considers the structure of multiclass classification by taking into account the relative scores of all classes.

We also proved a mistake bound that was stricter than the former algorithm. The resulting update formula for the learner also experimentally also outperformed other online algorithms and was thus demonstrated to be competitive with batch-learning algorithms.

References

[1] K.Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, 2006.

[2] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines.* Cambridge University Press, 2000.

[3] K. Crammer and Y. Singer. A new family of online algorithms for category ranking. *Jornal of Machine Learning Research*, 3:1025–1058, 2003.

[4] K. Crammer and Y. Singer. Ultraconservative online algorithms for multiclass problems. *Jornal of Machine Learning Research*, 3:951–991, 2003.

[5] Dimitri P. Bertsekas. *Nonlinear Programming.* Athena Scientific, September 1999.

[6] M. Herbster. Learning additive models online with fast evaluating kernels. In *Proc. of COLT 2001 and EuroCOLT 2001*, pages 444–460, 2001.

[7] A. B. J. Novikoff. On convergence proofs on perceptrons. In *Proc. of the Symposium on the Mathematical Theory of Automata*, volume 12, pages 615–622, 1962.

[8] J. J. Hull. A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):550–554, 1994.

[9] K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Jornal of Machine Learning Research*, 2:265–292, 2002.

[10] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[11] Y. Freund and R. E. Schapire. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296, 1999.

[12] Koby Crammer and Yoram Singer. On the learnability and design of output codes for multiclass problems. In *In Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, pages 35–46, 2000.

[13] M.Dredze, K.Crammer, and F.Pereira. Confidence-weighted linear classification. In *Proc. of ICML 2008*, pages 264–271, 2008.

[14] K.Crammer, M.Dredze, and F.Pereira. Multi-class confidence weighted algorithms. In *Proc. of NIPS 2008*, 2008.

[15] K.Crammer, M.Dredze, and A.Kulesza. Multi-class confidence weighted algorithms. In *Proc. of Emperical Methods of Natural Lunguage Processing*, pages 496–504, 2009.