Passive and Aggressive Algorithm with Class Mean Information

Yu-Lin Tsai, Yi-Chen Tseng, Yuh-Jye Lee and Hsing-Kuo Pao
Department of Computer Science and Information Engineering
National Taiwan University of Science and Technology, Taipei 106, Taiwan

I. Introduction

The online perceptron [1], [2] algorithm is a mistake-driven procedure which updates the current classifier whenever the new arriving data is misclassified. The algorithm can be interpreted as a stochastic gradient descent method which has been successfully applied to the learning task with large-scale data set [3], [4], [5]. How to decide the learning rate becomes an important issue in this type learning algorithm [6], [7], [8], [9], [10], [11]. The *passive and aggressive* algorithm proposed an updating scheme to determine the new updated classifier [12], [13], [14]. It suggests that the new classifier should not only classify the new arriving data correctly but also as close to the current classifier as possible. A closed form of updating rule was derived even taking the loss function of new arriving data into account.

Based on the success of PA algorithm [13], [14], we propose a new updating rule that will take the class mean information into account. The intuition behind our proposed method is that under the linearly separable assumption, the difference vector of positive mean and negative mean will suggests a good proximal classifier [15]. We incorporate this observation by adding a term $\|\mathbf{w}-\tilde{\mathbf{m}}\|_2^2$ into our new objective function where $\tilde{\mathbf{m}}$ is the difference between positive class mean and negative class mean of accumulated training data until now. As the PA algorithm did, a closed form of updating rule can be derived. Thus, our proposed method can have the same computational advantage with PA algorithm. The preliminary numerical results show that our proposed method is less sensitive to the input order of training data. Besides, the number of mistakes made in a single pass is less than the PA algorithm.

II. PA Algorithm with Class Means

Online learning considers an input of streaming of examples $(\mathbf{x}^t, y_t)$ in consecutive trials. In this study, we confined ourselves within the linear hypothesis space $\mathcal{H}$, that is defined by $\mathcal{H} = \{h_{\mathbf{w}} | h_{\mathbf{w}}(\mathbf{x}) = \text{sgn}(\langle \mathbf{x}, \mathbf{w}\rangle), \mathbf{w} \in \mathbb{R}^{\mathbf{n}}\}$. In the t trial, we predict the label of $\mathbf{x}^t$ based on the current classifier $\mathbf{w}^t$. If the output consists of the label y_t, we do nothing; otherwise, we update the current classifier $\mathbf{w}^t$ to improve its *performance* on the example $\mathbf{x}^t$. The *performance* of the classifier can be measured by a *loss function* $\ell(\mathbf{w}; (\mathbf{x}^t, y_t))$. If we can correctly predict $\mathbf{x}^t$ by $\mathbf{w}$, $\ell(\mathbf{w}; (\mathbf{x}^t, y_t)) = 0$. The ultimate learning goal is to minimize the cumulative loss in the whole consecutive trials. The passive and aggressive algorithm realizes this concept and enforces a criterion that the new classifier should not jump to far away from current classifier. Combining these two purposes together, the PA algorithm solves the minimization problem as follows:

$$\min_{\mathbf{w}\in\mathbb{R}^n} \ \frac{1}{2}\|\mathbf{w}-\mathbf{w}^t\|_2^2 + C\ell(\mathbf{w}; (\mathbf{x}^t, y_t)), \tag{1}$$

where $C > 0$ weights the importance of the loss function. We note that if the current classifier $\mathbf{w}^t$ correctly predicts the label of $\mathbf{x}^t$ it will be the optimal solution of (1). The closed update forms of PA algorithm for the squares loss, the hinge loss and the hard margin loss have been derived. This makes PA algorithm has been successfully applied to many real applications.

We extend the PA algorithm's idea. The new updated classifier should be close to a *proximal model* that provides the information about the accumulated examples until now. Under the linearly separable assumption, the difference vector of positive mean and negative mean will suggests a good proximal classifier[15]. We incorporate this observation into our optimization model. Thus, we have

$$\min_{\mathbf{w}\in\mathbb{R}^n} \ \frac{1}{2}\|\mathbf{w}-\mathbf{w}^t\|_2^2 + \ \frac{\gamma}{2}\|\mathbf{w}-\tilde{\mathbf{m}}\|_2^2 + C\ell(\mathbf{w}; (\mathbf{x}^t, y_t)), \tag{2}$$

where $C > 0$ and $\gamma > 0$ are weighted parameters. The $\tilde{\mathbf{m}} = \mathbf{m}_+ - \mathbf{m}_-$ which is the difference between positive mean and negative mean of accumulated examples until now. It is very easy to update the exactly $\tilde{\mathbf{m}}$ of accumulated examples without keeping all examples in the pass. We simplify the minimization problem (2) and have an equivalent minimization problem as follows:

$$\min_{\mathbf{w}\in\mathbb{R}^n} \ \frac{1+\gamma}{2}\|\mathbf{w}\|_2^2 - \langle \mathbf{w}^t + \gamma\tilde{\mathbf{m}}, \mathbf{w}\rangle + C\ell(\mathbf{w}; (\mathbf{x}^t, y_t)). \tag{3}$$

Similar to the PA algorithm, we derive the close form of updating rules for the hard margin loss (PAm), the hinge loss (PAm-1) and the squares loss (PAm-2) as follows,

$$\mathbf{w}^{t+1} = \frac{1}{1+\gamma}(\mathbf{w}^t + \gamma\tilde{\mathbf{m}} + \alpha y_t \mathbf{x}^t)$$

where α is defined as

$$\alpha = \begin{cases} \frac{loss+\gamma(1-y_t\langle\tilde{\mathbf{m}},\mathbf{w}^t\rangle)}{\|\mathbf{w}^t\|_2^2} & \text{for PAm} \\ \min\{C, \frac{loss+\gamma(1-y_t\langle\tilde{\mathbf{m}},\mathbf{w}^t\rangle)}{\|\mathbf{w}^t\|_2^2}\} & \text{for PAm-1} \\ \frac{loss+\gamma(1-y_t\langle\tilde{\mathbf{m}},\mathbf{w}^t\rangle)}{\|\mathbf{w}^t\|_2^2+\frac{1+\gamma}{2C}} & \text{for PAm-2} \end{cases}$$

and the

$$loss = 1 - y_t\langle \mathbf{x}^t, \mathbf{w}^t\rangle.$$

The details of our proposed method is summarized in Algorithm 1.

Algorithm 1: PAm Algorithm (single pass)

```
/* m:samples, n:dimensions,
   (X, Y):training set                                   */
Input: X ∈ R^{m×n}, Y ∈ R^m, C > 0, γ > 0
Output: w
begin
   /* initialization                                     */
   w^1 = 0, m_+ = 0, m_- = 0, Pm = 0, Nm = 0;
   for t = 1 : m do
      if y_t > 0 then
         m_+ = (Pm * m_+ + w^t) / (Pm + 1);
      else
         m_- = (Nm * m_- + w^t) / (Nm + 1);
      end
      loss = 1 − y_t ⟨x^t, w^t⟩;
      if loss > 0 then
         /* aggressive                                   */
         m̃ = m_+ − m_-;
         α =
            (loss + γ(1 − y_t ⟨m̃, w^t⟩)) / ||x^t||_2^2                          for PAm
            min{C, (loss + γ(1 − y_t ⟨m̃, w^t⟩)) / ||x^t||_2^2}                  for PAm-1
            (loss + γ(1 − y_t ⟨m̃, w^t⟩)) / (||x^t||_2^2 + (1+γ)/(2C))           for PAm-2
         w^{t+1} = 1/(1+γ) (w^t + γ m̃ + α y_t w^t);
      else
         /* passive                                      */
         w^{t+1} = w^t;
      end
   end
end
```

If we are allowed to maintain a fixed size queue $\mathbf{Q_w}$ so that we can remember many latest new classifiers. We can replace $\mathbf{w}^t$ by $\bar{\mathbf{w}}_q$, where $\bar{\mathbf{w}}_q$ is the *average* classifier in the queue $\mathbf{Q_w}$. Intuitively, we will have more stable model than PAm does. We describe the details in Algorithm 2.

Algorithm 2: PAm-W Algorithm (keep many latest new classifiers in a queue)

```
/* m:samples, n: dimensions,
   (X, Y):training set                                   */
Input: X ∈ R^{m×n}, Y ∈ R^m, C > 0, γ > 0
Output: w
begin
   /* initialization                                     */
   w^1 = 0, m_+ = 0, m_- = 0, Q_w = 0, Pm = 0,
   Nm = 0;
   for t = 1 : m do
      if y_t > 0 then
         m_+ = (Pm * m_+ + w^t) / (Pm + 1);
      else
         m_- = (Nm * m_- + w^t) / (Nm + 1);
      end
      loss = 1 − y_t ⟨x^t, w^t⟩;
      if loss > 0 then
         if | Q_w | is not full then
            Q_w = [Q_w, w^t]
         else
            /* Q_w is full, FIFO
               replacement                              */
            Q_w = [Q_w(2 : end), w^t]
         end
         /* aggressive                                   */
         m̃ = m_+ − m_-;
         α =
            (loss + γ(1 − y_t ⟨m̃, w̄_q⟩)) / ||x^t||_2^2                          for PAmW
            min{C, (loss + γ(1 − y_t ⟨m̃, w̄_q⟩)) / ||x^t||_2^2}                  for PAmW-1
            (loss + γ(1 − y_t ⟨m̃, w̄_q⟩)) / (||x^t||_2^2 + (1+γ)/(2C))           for PAmW-2
         w^{t+1} = 1/(1+γ) (w̄_q + γ m̃ + α y_t w̄_q);
      else
         /* passive                                      */
         w^{t+1} = w^t;
      end
   end
end
```

III. Numerical Result

In order to demonstrate the robustness and efficiency of our methods, we conduct several experiments and compare the results with conventional PA algorithm. The datasets that we used in the numerical tests come from two sources, LIBSVM library site [1] and the Pascal Large Scale Learning Challenge in 2008 [2]. The first source provides 4 moderate size datasets. We implemented our algorithms in MATLAB and all experiments were run on a personal computer consisting of a 3.0 GHz processor and 4 gigabytes RAM. Table I summarizes the sizes of benchmark datasets.

In the first part of our experiment, we would like to investigate the effect of the input order of examples. We run a single pass for PA algorithm and PAm algorithm ten times with different input order at each time. We report the average CPU seconds and the average number of updates in a single pass in Table II and Table III respectively. PAm family has much less number of update thus can be finished a single pass with a shorter CPU time. We note that the less number

TABLE I
THE STATISTICS FOR 4 MEDIUM-SIZE AND 3 LARGE-SCALE DATA SETS.

Dataset	Training	Testing	Features
svmguide1	3089	4000	4
w3a	4912	44837	300
ijcnn1	35000	91701	22
adult	32550	16281	123
alpha	400000	100000	500
delta	400000	100000	500
gamma	400000	100000	500

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

[2] http://largescale.ml.tu-berlin.de/about/.

of update indicates that less mistakes made in the learning process. The average testing error rate and standard deviation of these ten testing error rates are shown in TABLE IV and TABLE V respectively. These results show that PAm family has better performance in accuracy. Moreover, PAm family has much smaller standard deviation than PA family. That is an evidence shows that our proposed method is less sensitive to the input order. We also give the best performance of these methods in Table VI. In summary, the PAm-1, PAm algorithm with the hinge loss, outperforms in most of datasets.

TABLE II
COMPARISON OF AVERAGE RUNNING TIME

Dataset	PA	PA-1	PA-2	PAm	PAm-1	PAm-2
svmguide1	0.0248	0.0558	0.0276	**0.0215**	0.0249	0.0235
w3a	**0.0388**	0.0627	0.0461	0.0600	0.0673	0.0666
ijcnn1	0.2466	0.4669	0.2689	**0.2138**	0.2447	0.2182
adult	0.3286	0.5364	0.4002	0.3083	0.3224	**0.2957**
alpha	42.5299	52.2090	47.3214	**13.0613**	13.8954	13.3054
delta	25.6031	32.2362	42.4580	26.1802	14.0128	**13.4835**
gamma	23.0035	26.5207	23.1364	15.2128	**12.7841**	13.1756

TABLE III
COMPARISON OF AVERAGE NUMBER OF UPDATES

Dataset	PA	PA-1	PA-2	PAm	PAm-1	PAm-2
svmguide1	1053.7	1069.0	1231.9	737.9	**728.3**	774.4
w3a	542.7	562.3	734.4	384.9	**337.8**	406.7
ijcnn1	10620.4	10616.8	11912.8	**4525.6**	5367.1	5120.3
adult	13020.6	12378.2	17343.3	6436.3	**5485.0**	5565.3
alpha	261730.5	261493.7	262013.2	**120078.5**	129455.3	121121.6
delta	226541.7	226506.0	226853.9	103098.9	88980.7	**88953.2**
gamma	216626.6	216670.0	217098.8	96354.3	82639.5	**82601.6**

TABLE IV
COMPARISON OF AVERAGE TESTING ERROR RATE

Dataset	PA	PA-1	PA-2	PAm	PAm-1	PAm-2
svmguide1	0.0788	0.0830	0.0731	0.0778	0.0716	**0.0712**
w3a	0.0230	0.0239	0.0220	0.0219	**0.0211**	0.0220
ijcnn1	0.1055	0.0938	0.0912	0.0975	**0.0803**	0.0855
adult	0.2013	0.1654	0.1715	0.1788	**0.1606**	0.1629
alpha	0.3008	0.3520	0.3367	0.2638	**0.2490**	0.2638
delta	0.2936	0.2993	0.2937	0.2713	**0.2161**	0.2177
gamma	0.2886	0.2845	0.2800	0.2408	**0.2016**	0.2021

TABLE V
COMPARISON OF STANDARD DEVIATION TESTING ERROR RATE

Dataset	PA	PA-1	PA-2	PAm	PAm-1	PAm-2
svmguide1	0.0447	0.0644	0.0599	0.0437	0.0362	**0.0238**
w3a	0.0048	0.0057	0.0054	0.0019	**0.0015**	0.0037
ijcnn1	0.0327	0.0152	0.0095	0.0280	**0.0063**	0.0094
adult	0.0329	0.0070	0.0102	0.0141	**0.0062**	0.0079
alpha	0.0613	0.0921	0.0715	**0.0260**	0.0093	0.0274
delta	0.0860	0.0525	0.0535	0.0708	**0.0004**	0.0022
gamma	0.0817	0.0555	0.0588	0.0517	**0.0004**	0.0009

In the second part of experiment, we investigate the convergence of PA algorithm and PAm algorithm. We run 20 epochs for each methods and record the classifier when an epoch is complete. For the PAm family, the proximal classifier will not be changed once the first epoch is finished. We compute the cosine similarity between two consecutive epoch classifiers. We plot the results for `svmguide1` and `adult` datasets in Fig. 1. From these results, we observe that PAm-1 will have almost identical classifiers after 3 to 5 epochs. We also observe the convergency of our proposed methods from the angle of error rate as well as the number of update within one epoch. The baseline of error rate is provided by the smooth support vector machine (SSVM[16]) which is a batch learning algorithm. PAm-1 will approach to SSVM's results after 3 to 5 epochs on `gamma` and `delta` datasets. We summarize these results in Fig. 2. In the batch learning framework, if the given dataset is linear non-separable, certain samples must be misclassified. We record the number of update in each epoch and plot the results in Fig. 3. We find that the PA and PAm go through enough passes the updating times will be almost fixed. However, the PAm will be fixed to the smaller updating times than PA. In the other words, the numbers of misclassified samples on PAm are less than PA after enough passes.

TABLE VI
COMPARISON OF MINIMAL TESTING ERROR RATE

Dataset	PA	PA-1	PA-2	PAm	PAm-1	PAm-2
svmguide1	0.0458	0.0443	0.0443	0.0438	0.0430	**0.0428**
w3a	0.0192	0.0186	0.0189	0.0186	**0.0185**	**0.0185**
ijcnn1	0.0844	0.0823	0.0830	0.0792	**0.0727**	0.0765
adult	0.1666	0.1603	0.1582	0.1589	**0.1526**	0.1540
alpha	0.2446	0.2452	0.2458	0.2415	**0.2386**	0.2400
delta	0.2215	0.2381	0.2232	0.2169	**0.2155**	0.2158
gamma	0.2169	0.2092	0.2051	0.2020	**0.2010**	0.2013

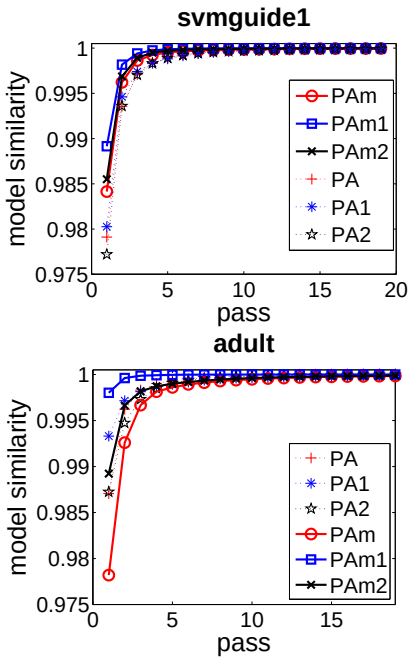

Fig. 1. Comparison of model similarity between two consecutive passes for the data sets *svmguide1* and *adult*.

IV. Summary and Future Work

We incorporate a proximal classifier with the passive and aggressive algorithm to solve the large scale binary classification problem under the online learning setting. The proximal classifier is defined by the difference of class means of the

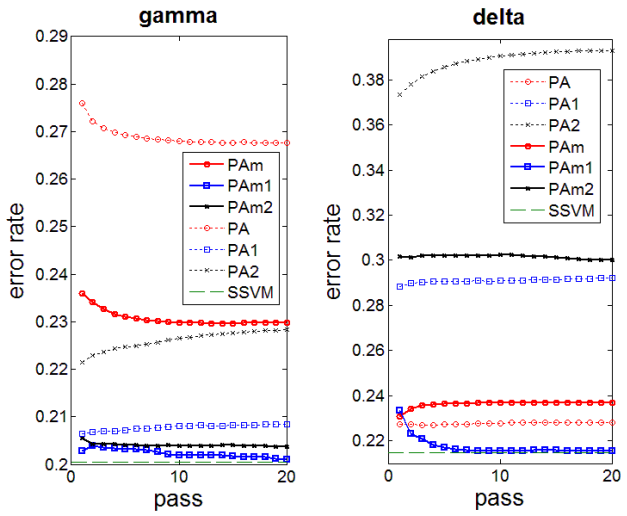

Fig. 2. Comparison of error rate convergence for data sets *gamma* and *delta* based on several PAm and PA algorithms.

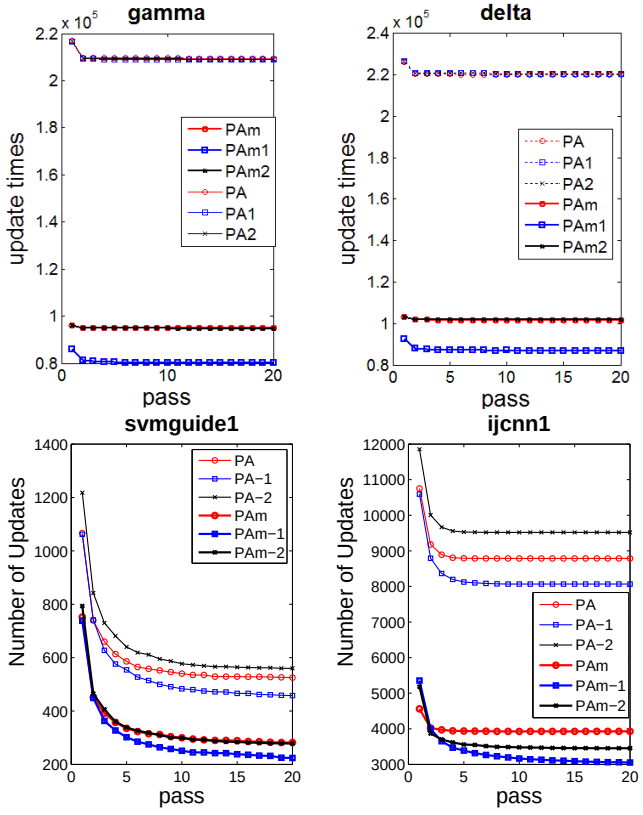

Fig. 3. Comparison of the number of updates for four data sets based on PAm and PA algorithms.

accumulate examples until current trial. Similar to the PA algorithm, we derived the close forms of updating rules for three popular loss functions. We tested our proposed method on seven public available benchmark datasets. The experimental results show that our method is less sensitive to the input data order and has a less number of updating than conventional PA algorithm. Thus, we can have a better performance than PA algorithm in CPU time. Besides, the similarity of the resulting classifiers of two consecutive passes is also higher than the PA algorithm. That means we can have an approximated classifier in a single pass.

For the future work, one possible extension is to modify the Algorithm 2. We can use weighted average instead of simple average. The weights can be determined by each classifier's performance on a certain validation set. We also would like to extend the linear classifier to the nonlinear classifier by utilizing the reduced kernel trick [17], [18]. Thus, we can enlarge the hypothesis to deal with more complicated data pattern.

REFERENCES

[1] N. Cristianini and J. Shawe-Taylor. *An introduction to support Vector Machines: and other kernel-based learning methods*. Cambridge Univ Pr, 2000.

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

[3] L. Bottou and Y. Le Cun. On-line learning for very large data sets. *Applied Stochastic Models in Business and Industry*, 21(2):137–151, 2005.

[4] L. Bottou. Online learning and stochastic approximations. *On-line learning in neural networks*, pages 9–42, 1998.

[5] T. Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the twenty-first international conference on Machine learning*, page 116. ACM, 2004.

[6] N. Cesa-Bianchi, A. Conconi, and C. Gentile. A second-order perceptron algorithm. In *Computational Learning Theory*, pages 129–140. Springer, 2002.

[7] C.N. Hsu, H.S. Huang, Y.M. Chang, and Y.J. Lee. Periodic step-size adaptation in second-order gradient descent for single-pass on-line structured learning. *Machine learning*, 77(2):195–224, 2009.

[8] A. Bordes, L. Bottou, and P. Gallinari. Sgd-qn: Careful quasi-newton stochastic gradient descent. *The Journal of Machine Learning Research*, 10:1737–1754, 2009.

[9] A. Bordes, L. Bottou, P. Gallinari, J. Chang, and S.A. Smith. Erratum: SGDQN is Less Careful than Expected. *Journal of Machine Learning Research*, 11:2229–2240, 2010.

[10] SVN Vishwanathan, N.N. Schraudolph, and A.J. Smola. Step size adaptation in reproducing kernel Hilbert space. *The Journal of Machine Learning Research*, 7:1107–1133, 2006.

[11] C. Angulo and A. Català. Online Learning with kernels for Smart Adaptive Systems: a review. *Proc. European Network for Intelligent Technologies, Oulu, Finland*, 2003.

[12] M. Dredze, K. Crammer, and F. Pereira. Confidence-weighted linear classification. In *Proceedings of the 25th international conference on Machine learning*, pages 264–271. ACM, 2008.

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

[14] S. Shalev-Shwartz. *Online learning: Theory, algorithms, and applications*. PhD thesis, Hebrew University, 2007.

[15] B. Scholkopf and A.J. Smola. *Learning with kernels*, volume 64. Citeseer, 2002.

[16] Y.J. Lee and O.L. Mangasarian. SSVM: A smooth support vector machine for classification. *Computational optimization and Applications*, 20(1):5–22, 2001.

[17] Y.J. Lee and O.L. Mangasarian. RSVM: Reduced support vector machines. In *SIAM International Conference on Data Mining*, pages 00–07. Citeseer, 2001.

[18] Y.J. Lee and S.Y. Huang. Reduced support vector machines: A statistical theory. *Neural Networks, IEEE Transactions on*, 18(1):1–13, 2006.