Online Distributed Passive-Aggressive Algorithm for Structured Learning

Jiayi Zhao, Xipeng Qiu*, Zhao Liu, and Xuanjing Huang

School of Computer Science, Fudan University, China

Abstract. The training phase is time-consuming for structured learning, especially for supper-tagging tasks. In this paper, we propose an online distributed Passive-Aggression (PA) by averaging parameters for parallel training, which can reduce the training time significantly. We also give theoretic analysis for its convergence. Experimental results show that our method can accelerate the training process significantly with comparable or even better accuracy.

1 Introduction

Structured learning [1] becomes a popular research topic recently. In the situation of structured learning, the labels of the data are not independent from each other, instead they form a mutually associated structure.

A cumbersome problem of structured learning is that the training time is too long, since structured learning algorithms are usually polynomial complexity in the number of labels. The larger the label set is, the more complex the algorithm will be. Traditional methods reduce the search space of the structural labels by pruning to improve computation efficiency [2]. However, these methods have a more or less performance loss.

In this paper, we propose an online distributed Passive-Aggression (PA) algorithm. We construct a parallel version of standard PA algorithm via weights averaging strategy. We also give a theoretic proof of the convergence of the training process and show that the distributed algorithm gives the same cumulative loss upper bound as the standard PA algorithm.

Our experiments show that the parallel framework gets the comparable or even better accuracy than the standard PA algorithm [3] with less training time. If the standard PA algorithm suffers from a scalability problem in both memory space and computational time when the size of a dataset is too large, our distributed PA algorithm will overcome these difficulties with less space and time using a parallel strategy and just a few machine nodes.

The rest of the paper is organized as follows. We begin by briefly reviewing the necessary background in the Passive-Aggressive algorithm in Section 2. Then we describe in detail our implementation of the distributed PA algorithm in Section 3 and give a theoretic analysis in Section 4. Finally, we report our results and analysis of experiments.

* Corresponding author, Email: xpqiu@fudan.edu.cn

2 Online Passive-Aggressive Algorithm

The online algorithms do not define an object function on the entire sample set because they need not obtain all samples at once. They update the parameters only depending on the observing sample. Perceptron algorithm [4] is a famous online algorithm, which is a simple but efficient and is used extensively in structured learning [5]. In contrast to perceptron, it is guided by the margin maximum idea.

Online Passive-Aggressive algorithm is a margin based online learning algorithm for various prediction tasks [3]. The difference between this algorithm and perceptron is that PA uses the margin of the samples to update the current classifier. Online PA algorithm updates the weights of features by solving a constrained optimization problem. It requires that the updated weights must stay as close as possible to the previous weights and on the other hand the updated weights correctly classify the current example with a sufficiently high margin. [6] applied this algorithm to the dependency parsing task which is a typical structured learning problem.

Given a series of samples $(\mathbf{x_t}, \mathbf{y_t})$ denoted as $\mathcal{T}$, in each sample every $\mathbf{x_t}$ has a corresponding label $\mathbf{y_t} \in \mathcal{Y}$, $\mathcal{Y}$ is the set of all labels any $\mathbf{x_t}$ can has. Define $\mathbf{\Phi}(\mathbf{x}, \mathbf{y}) \in \{0,1\}^d$ as a feature vector of d dimensions on the sample $(\mathbf{x}, \mathbf{y})$. The value in each dimension of the $\mathbf{\Phi}(\mathbf{x}, \mathbf{y})$ is a $0-1$ value which indicates whether this feature occurs in the current sample. $\mathbf{w} \in \mathbb{R}^d$ is the parameter vector of the classifier and each dimension of it is the weight of one feature. In order to learn a proper weight vector $\mathbf{w}$ online PA algorithm adopts an iterative method until the convergence of $\mathbf{w}$ or the number of loops exceeds the predefined maximum iteration number.

On round t, at first the classifier predicts the labels $\hat{\mathbf{y}}_t$ for a sample $\mathbf{x_t}$:

$$\hat{\mathbf{y}}_t = \arg\max_{\mathbf{z}}(\mathbf{w} \cdot \Phi(\mathbf{x}_t, \mathbf{z})), \tag{1}$$

After the prediction, the algorithm receives the correct set of relevant labels $\mathbf{Y}_t$ and we define the margin is:

$$\gamma(\mathbf{w}_t; (\mathbf{x}_t, \mathbf{Y}_t)) = \min_{r \in \mathbf{Y}_t} \mathbf{w}_t \cdot \Phi(\mathbf{x}_t, r) - \max_{s \notin \mathbf{Y}_t} \mathbf{w}_t \cdot \Phi(\mathbf{x}_t, s), \tag{2}$$

From above the margin is positive if and only if all of the relevant labels are ranked higher than all of the irrelevant labels. However online PA algorithm is not satisfied by a mere positive margin as it requires the margin of every predication to be at least a loss function $L(\mathbf{y}_t, \hat{\mathbf{y}}_t)$. The hinge-loss function which represents the loss made when the classifier gives such a prediction is defined as:

$$\ell(\mathbf{w}; (\mathbf{x}_t, \mathbf{y}_t)) = \begin{cases} 0, & \gamma(\mathbf{w}; (\mathbf{x}_t, \mathbf{y}_t)) > L(\mathbf{y}_t, \hat{\mathbf{y}}_t) \\ L(\mathbf{y}_t, \hat{\mathbf{y}}_t) - \gamma(\mathbf{w}; (\mathbf{x}_t, \mathbf{y}_t)), & \text{otherwise} \end{cases} \tag{3}$$

On round t online PA algorithm sets the new weight vector $\mathbf{w}_{t+1}$ to be the solution to the following constrained optimization problem.

$$\begin{aligned} \mathbf{w}_{t+1} &= \arg\min_{\mathbf{w} \in \mathbb{R}^d} \frac{1}{2}||\mathbf{w} - \mathbf{w}_t||^2 + \mathcal{C} \cdot \xi, \\ &\textbf{s.t. } \ell(\mathbf{w}; (\mathbf{x}_t, \mathbf{y}_t)) <= \xi \textbf{ and } \xi >= 0 \end{aligned} \tag{4}$$

where $\mathcal{C}$ is a positive parameter which controls the influence of the slack term on the objective function.

Satisfying the single constraint in the optimization problem above is equivalent to satisfying the following set of linear constraints.

$$\begin{aligned} \mathbf{w}_t \cdot \Phi(\mathbf{x}_t, \mathbf{y}_t) - \mathbf{w}_t \cdot \Phi(\mathbf{x}_t, \mathbf{y})) \geqslant L(\mathbf{y}_t, \mathbf{y}) - \xi \\ \forall \mathbf{y} \in \{\mathcal{Y} \backslash y_t\} \end{aligned} \tag{5}$$

then the large enough margin between the correct labels and the other labels is guaranteed. Solving this constraint problem leads to a parameters update formula,

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \tau_t(\Phi(\mathbf{x}_t, \mathbf{y}_t) - \Phi(\mathbf{x}_t, \hat{\mathbf{y}}_t)). \tag{6}$$

here,

$$\tau_t = \min \left\{ C, \frac{\ell(\mathbf{w}; (\mathbf{x}_t, \mathbf{y}_t))}{||\Phi(\mathbf{x}_t, \mathbf{y}_t) - \Phi(\mathbf{x}_t, \hat{\mathbf{y}}_t)||^2} \right\} \tag{7}$$

Here C is a positive parameter which controls the influence of the slack term on the objective function. In order to prevent noise samples to make the τ_t too large to depart from the classification face, so τ_t is forced to be smaller than C.

input : training data set: $(\mathbf{x}_n, \mathbf{y}_n), n = 1, \cdots, N$, and parameters: $\mathcal{C}, K$
output: $\mathbf{w}$

Initialize: $\mathbf{cw} \leftarrow 0, \mathbf{w}_0 \leftarrow 0$;
for $k = 0 \cdots K-1$ **do**
  for $t = 0 \cdots T-1$ **do**
    receive an example $(\mathbf{x}_t, \mathbf{y}_t)$;
    predict: $\hat{\mathbf{y}}_t = \arg\max_{\mathbf{z} \neq \mathbf{y}_t} \langle \mathbf{w}_t, \Phi(\mathbf{x}_t, \mathbf{z}) \rangle$;
    calculate $\ell(\mathbf{w}; (\mathbf{x}, \mathbf{y}))$;
    update $\mathbf{w}_{t+1}$ with Eq.(6);
  end
  $\mathbf{cw} = \mathbf{cw} + \mathbf{w}_T$;
end
return $\mathbf{w}$;

Algorithm 1: Online Passive-Aggressive Algorithm

We assume that there exists some $\mathbf{u} \in \mathbb{R}^d$ such that $y_t(\mathbf{u} \cdot \mathbf{x}_t) > 0$ for all $t \in 1...T$. In the other word $\mathcal{T}$ are absolutely separable because such $\mathbf{u}$ exists. So from Eq. 3 for all samples $\mathbf{x}_t$ we can get $\ell(\mathbf{u}, \mathbf{x}_t) = 0$. Assume that for all feature function $\Phi, (\mathbf{x}, \mathbf{y})$ $||\Phi(\mathbf{x}, \mathbf{y})||^2 \leqslant R/2$ is satisfied. Then [3] proves the cumulative squared loss of PA algorithm on this sequence of examples is bounded by

$$\sum_{t=1}^{T} \ell_t^2 \leq R^2 ||\mathbf{u}||^2 \tag{8}$$

3 Distributed Implementation of Online Passive-Aggressive Algorithm

Since the online algorithms update the parameters only depending on the observing sample, there is a straight-forward way for distributed training.

Firstly we divide the training set $\mathcal{T}$ into S disjoint pieces $\mathcal{T} = \{\mathcal{T}_1, ..., \mathcal{T}_S\}$ randomly. After dividing samples, all pieces are evenly assigned to different cores[1]. Each core trains the parameters with PA algorithm on its assigned samples. After all cores finish training, a summarization system averages the S parameter vectors to get a new parameter vector and send it to each core for the next training iteration. The above process is repeated until the parameters converge.

This distributed online algorithm can be easily transplanted to the popular MapReduce [7] framework. The Mapper process trains parameters with each piece of samples using the Passive-Aggressive algorithm and transfer these parameters to the Reducer. The Reducer is responsible for gathering these parameters, averaging them and finally sending them to each Mapper.

Algorithm 2 is the pseudo code of the distributed Passive-Aggressive algorithm. In Algorithm 2, $\mu_{i,n}$ is a distribution which averages the parameters vector returned by each classifier. $\mathbf{w}^{(avg,n-1)}$ is the initial parameters vector on round n.

3.1 Parameters Averaging Strategy

An unsolved problem is how to mix the parameters in each iteration. In other words, we need assign a proper value of $\mu_{i,n}$ for each piece i.

Here, we discuss two different parameters averaging strategies.

One is the uniform averaging strategy, called "uniform mixing".

Suppose that the sample set is randomly split into S portions, we set $\mu_{i,n}$ to a uniform distribution,

$$\mu_{i,n} = \frac{1}{S}. \tag{9}$$

Thus, the newer $\mathbf{w}^{(avg,n)}$ is defined as

$$\mathbf{w}^{(avg,n)} = \frac{\sum_{i=1}^{S} \mathbf{w}^{(i,n)}}{S}. \tag{10}$$

Another is the weighted averaging strategy, called "error mixing". In "uniform mixing" strategy, if a machine or core is arranged to deal with a more difficultly-treated sample portion than others, the parameters obtained from it should be more important. The updating speed will be dragged down due to uniform averaging strategy. Therefore, we wish the touchy parts to contribute more in parameters updating. In the other word, we want to increase the influence of the parameters trained from those pieces with more errors. So we use the following formula to average the parameters from all portions.

$$\mathbf{w}^{(avg,n)} = \sum_{i=1}^{S} \frac{\delta^{i,n}}{\sum_{k=1}^{S} \delta^{i,n}} \mathbf{w}^{(i,n)}, \tag{11}$$

[1] The cores can be located in different computing nodes.

```
// The Client function
// Here 1 ≤ t ≤ |T_i|
Cient (T_i : {(x_t, y_t)}, w^(avg,n−1))
begin
    Initialize: w^(0) = w^(avg,n−1), k = 0;
    for t = 1 ··· |T_i| do
        ŷ_t = argmax_{y∈Y} (w · Φ(x_t, y));
        if ŷ_t ≠ y_t then
            w^(k+1) = w^(k) + τ_t (Φ(x_t, y_t) − Φ(x_t, ŷ_t));
            k = k + 1;
        end
    end
    w^(i,n) = w^(k);
    output: w^(i,n)
end
// The Server function
// Here Σ_{i=1} μ_{i,n} = 1
Server ({w^(i,n), 1 ≤ i ≤ |S|})
begin
    w^(avg,n) = Σ_{i=1} μ_{i,n} w^(i,n);
    output: w^(avg,n)
end
```

Algorithm 2: Distributed Passive-Aggressive Algorithm

where $\delta^{i,n}$ is defined as the number of wrong classified samples in the ith training portion on the nth round.

4 Theoretical Analysis

We also need to estimate the cumulative loss produced by the distributed Passive-Aggression algorithm.

For every feature function $\Phi(\mathbf{x}, \mathbf{y})$ assume that $||\Phi(\mathbf{x}, \mathbf{y})||^2 \leq R/2$ is satisfied and introduce a notation $\Delta\Phi = \Phi(\mathbf{x}, \mathbf{y}) - \Phi(\mathbf{x}, \hat{\mathbf{y}})$.

The loss function is $\ell(\mathbf{w}) = L(\mathbf{y}, \hat{\mathbf{y}}) - (\mathbf{w} \cdot \Delta\Phi)$ following the sections above. If the final classification hyperplane produced by the classifier is $\mathbf{u} \in \mathbb{R}^d$ then we denote the result loss function as $\ell^* = \ell(\mathbf{u}) = L(\mathbf{y}, \hat{\mathbf{y}}) - (\mathbf{u} \cdot \Delta\Phi)$. We define Δ_t as the difference between the square of the distance $\mathbf{w}_t$ apart from $\mathbf{u}$ and the square of the distance $\mathbf{w}_{t+1}$

apart from $\mathbf{u}$ on round t,

$$\begin{aligned}\Delta_t &= ||\mathbf{w}_t - \mathbf{u}||^2 - ||\mathbf{w}_{t+1} - \mathbf{u}||^2 \\ &= ||\mathbf{w}_t - \mathbf{u}||^2 - ||\mathbf{w}_t + \tau_t \Delta\Phi - \mathbf{u}||^2 \\ &= -2\tau_t (\mathbf{w}_t - \mathbf{u}) \cdot \Delta\Phi - \tau_t^2 ||\Delta\Phi||^2 \\ &\geq 2\tau_t (\ell_t - L(\mathbf{y}, \hat{\mathbf{y}}) - (\ell_t^* - L(\mathbf{y}, \hat{\mathbf{y}}))) - \tau_t^2 ||\Delta\Phi||^2 \\ &= \tau_t \left(2\ell_t - \tau_t ||\Delta\Phi||^2 - 2\ell_t^*\right)\end{aligned} \tag{12}$$

If the sample set $\mathcal{T} = \{(\mathbf{x}_1, \mathbf{y}_1), ... (\mathbf{x}_t, \mathbf{y}_t)\}$ is linearly separable, we will get $\ell^* = 0$ is satisfied for any sample. Then

$$\Delta_t \geq \tau_t (2l_t - \tau_t ||\Delta\Phi||^2) \; = \; l_t^2 / ||\Delta\Phi||^2 \; \geq \; l_t^2 / R^2 \tag{13}$$

from above,

$$\ell_t^2 / R^2 \leq ||\mathbf{w}_t - \mathbf{u}||^2 - ||\mathbf{w}_{t+1} - \mathbf{u}||^2 \tag{14}$$

So the upper bound of the loss of a sample is determined by the difference of the parameters and the final classification face $\mathbf{u}$ before and after each round.

Now consider the cumulative loss after weighted average on round n,

$$\sum_{i=1}^{S} \mu_{i,n} \sum_{t_i=1}^{T_i} \frac{l_{t_i,n}^2}{R^2} \quad \leq \quad ||\mathbf{w}^{(avg,n-1)} \; - \; \mathbf{u}||^2 \; - \; \sum_{i=1}^{S} \mu_{i,n} ||\mathbf{w}^{(i,n)} \; - \; \mathbf{u}||^2 \tag{15}$$

According to the algorithm 2 we know $\mathbf{w}^{(avg,n)} = \sum_{i=1}^{S} \mu_{i,n} \mathbf{w}^{(i,n)}$ and use Jensen's inequality. Δ^2 is denoted as $||\mathbf{w}^{(avg,n)} - \mathbf{u}||^2$,

$$\begin{aligned}\mathbf{\Delta^2} &= ||\sum_{i=1}^{S} \mu_{i,n} \mathbf{w}^{(i,n)} - \mathbf{u}||^2 \\ &\leq \sum_{i=1}^{S} \mu_{i,n} ||\mathbf{w}^{(i,n)}||^2 - 2 \sum_{i=1}^{S} \mu_{i,n} \mathbf{u} \cdot \mathbf{w}^{(i,n)} + ||\mathbf{u}||^2 \\ &= \sum_{i=1}^{S} \mu_{i,n} ||\mathbf{w}^{(i,n)} - \mathbf{u}||^2\end{aligned} \tag{16}$$

then

$$||\mathbf{w}^{(avg,n)} - \mathbf{u}||^2 \leq \sum_{i=1}^{S} \mu_{i,n} ||\mathbf{w}^{(i,n)} - \mathbf{u}||^2 \tag{17}$$

So the cumulative loss in N iterations is

$$\begin{aligned}&\sum_{n=1}^{N} \sum_{i=1}^{S} \mu_{i,n} \sum_{t_i=1}^{T_i} \frac{l_{t_i,n}^2}{R^2} \\ &\leq ||\mathbf{w}^{(avg,0)} - \mathbf{u}||^2 - \sum_{i=1}^{S} \mu_{i,n} ||\mathbf{w}^{(i,N)} - \mathbf{u}||^2 \\ &\leq ||\mathbf{w}^{(avg,0)} - \mathbf{u}||^2 \\ &= ||\mathbf{u}||^2\end{aligned} \tag{18}$$

so

$$\sum_{n=1}^{N}\sum_{i=1}^{S}\mu_{i,n}\sum_{t_i=1}^{T_i}l_{t_i,n}^2 \leq R^2||\mathbf{u}||^2 \tag{19}$$

Refer to the section 2 we can find that the online distributed Passive-aggression algorithm has the same cumulative loss upper bound.

5 Experiments

We verify the efficiency of our method with joint segmentation and POS-tagging problem, which is a common task in natural language processing (NLP). Structured learning methods are usually adopted to solve this problem nowadays but suffers from long training time.

The construction of our distributed algorithm is based on the FudanNLP toolkit [8] for natural language processing.

We use POS tagging task with CTB corpus from the SIGHAN Bakeoff [9]. There are 37 kinds of POS tags in this corpus. We use cross-label method for this task. The cross-label consists of two parts: the first part is the label of segmentation, and the second part is the label of POS tag. We use four segmentation labels ("B", "M", "E", "S") to represent the beginning, middle, end of a word and a single character word respectively. For example, a label "B-NN" means the beginning of a noun word. After processing the corpus, we get 108 different cross-labels.

Table 1 shows all the feature templates used in our task. These templates are very common for the segmentation and POS-tagging task.

Table 1. Feature Templates

(1.1)	$c_{i-2}t_i$, $c_{i-1}t_i$, c_it_i, $c_{i+1}t_i$, $c_{i+2}t_i$
(1.2)	$c_{i-1}c_it_i$, $c_ic_{i+1}t_i$, $c_{i-1}c_{i+1}t_i$
(1.3)	$t_{i-1}t_i$

Note: c_i represents the character at position i in the sentence and t_i is for the target label of this character.

In our experiments, $23,443$ sentences are used for training and $2,079$ sentences are used for testing. We split the training set into 2, 5, 10, 20 and 50 pieces randomly and train each piece in each machine or core.

We firstly compare the different averaging strategy of parameters.

Figure 1 gives the comparison between the uniform mixing and error mixing methods. In this experiment we split the entire sample set into 2 pieces and 5 pieces respectively and inspect the training errors at each iteration.

Figure 1 shows that on each round the numbers of training errors are almost the same for both 2 pieces and 5 pieces training strategy. This is because training corpus is randomly and uniformly split so the probability of hard tagging sentences gathering in a single piece is rare. So the number of training errors of each machine is very close. The

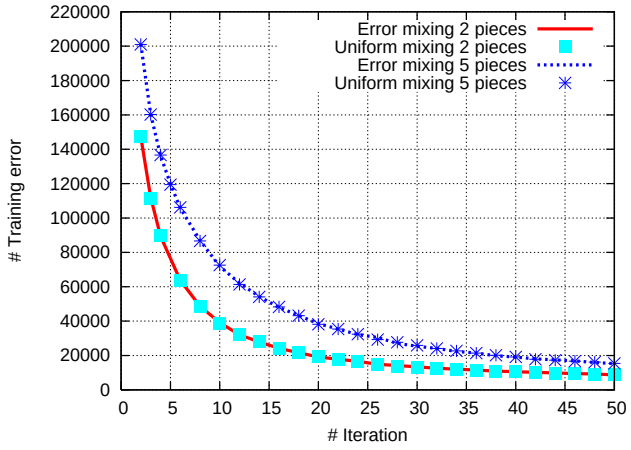

Fig. 1. Training errors with iterations. Uniform mixing just uniformly averages the weights from each machine and Error mixing assigns $\mathbf{w}^{(avg,n)} = \sum_{i=1}^{S} \frac{\delta^{i,n-1}}{\sum_{k=1}^{S} \delta^{i,n-1}} \mathbf{w}^{(i,n-1)}$ as the new weight using weighted average.

difference between the two averaging strategies is very small in this situation. However a better averaging strategy maybe speeds the process of training up.

In the later evaluation, we just use the "uniform mixing" strategy.

Table 2 gives the results of our method and the standard PA algorithm with different experimental settings. It shows that our algorithm can increase the speed of the training without accuracy loss.

The entries in the Table 2 with star '*' is the "Error Mixing" version of distributed online PA algorithm. As the result from Table 2 the distributed algorithm get approximate or even better result of the standard PA algorithm.

We also compared the performance of the training process and the convergence property of the normal online PA algorithm with our distributed implementation. Figure 2 displays relations between the training precision and the number of iterations under different pieces of samples. We set the maximum iterations to be 150. We can see that it needs the more iterations to convergence as the number of pieces increases. However, since each piece use less time for the larger number of pieces, we can reduce the training time in total.

To evaluate the effect of speed-up in training phrase, we compare the test precision as the time goes on. In this section we ignore the time of network I/O operations and only consider the accumulated CPU time. The comparisons are shown in Figure 3. Apparently, the distribution PA algorithm is faster than the original algorithm to achieve the same test precision. The parallel PA algorithm is also faster to converge with comparable precision to the normal PA algorithm. We can see that the test precision of distributed PA algorithm is a little higher.

Table 2. Evaluation of trained models

Methods	Segmentation		POS-tagging		Time(s)[†]
	Accuracy(%)	$F1$	Accuracy(%)	$F1$	
original	97.63	95.00	89.67	87.54	6679.3
2 pieces	97.54	94.95	89.39	87.36	3514.8
2 pieces[*]	97.53	94.90	89.18	87.01	3414.2
5 pieces	97.76	95.35	89.93	87.78	1839.9
5 pieces[*]	97.78	95.37	90.04	87.88	1259.4
10 pieces	97.82	95.45	90.05	87.91	761.6
20 pieces	97.81	95.44	90.12	88.10	959.9[♯]
50 pieces	97.85	95.52	90.32	88.34	654.5

[*] The method with * uses error mixing averaging strategy.
[†] This is training time omitting the time cost by data transferring.
[♯] When we split the sample set into 20 pieces the algorithm didn't convergence after running for 150 iterations.

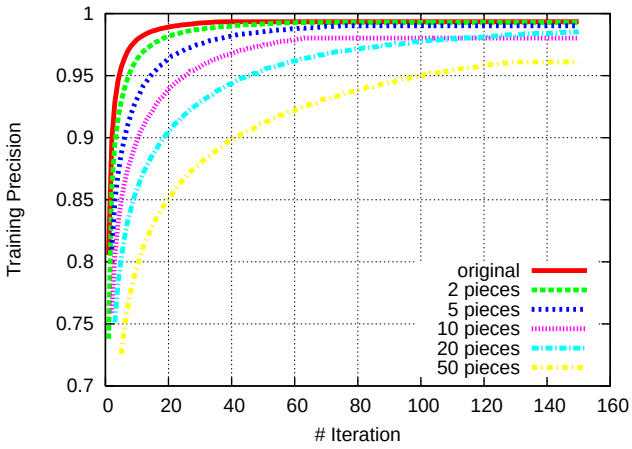

Fig. 2. Training precision with iterations.

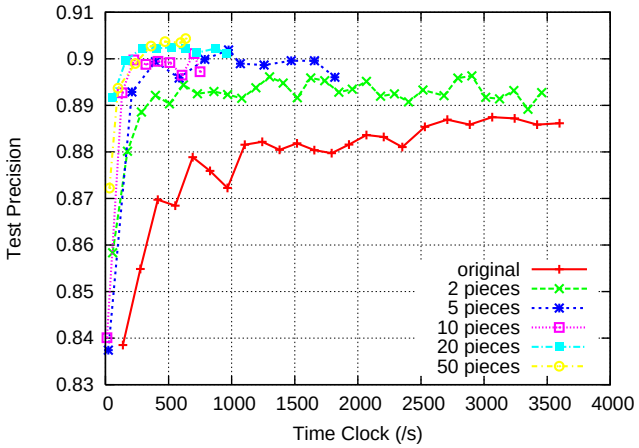

Fig. 3. Testing precision with training time

5.1 Experimental Discussion

There are some things to be worth noting from our experiments.

Firstly, the distributed online PA algorithm has a little higher testing accuracy than the standard one mentioned above. We suspect this happens for the season that the distributed method is a form of parameter averaging which has the same effect as the average perceptron [5]. Although training time is dramatically increased after we add an average strategy to the original algorithm, the test accuracy is still lower than the distributed algorithm. Maybe this happens because the original version overfits the training data.

Secondly, when the training set is split into large enough pieces, the time in the testing data is almost the same to converge and to achieve the best performance. This is shown in Figure 3 that lines in the left up corner cover each other. Further considering the network I/O waste, an unlimited split of the sample set is undesirable. The segmentation with 10 pieces is a better choice for POS tagging in our paper.

6 Related Works

The distributed machine learning is attracting increasing attention in recent years. For example, Mahout[2] is a famous package of scalable parallel machine learning libraries based on MapReduce [7] framework. The distributed computing framework also provides a new way to deal with the large computation cost in structured learning.

Chu et al.[10] develop a broadly applicable parallel programming method, which is easily applied to many different learning algorithms. They have also investigated parallel implements of many batch algorithms and modified these algorithms in the map-reduce framework. For online algorithms, parameters updating process is a serial

[2] http://mahout.apache.org/

procedure. In order to guarantee the accuracy and convergence of online algorithms we need a particular strategy to make the serial process parallel.

Chang et al.[11] develop a parallel support vector machine (SVM)[12] algorithm which reduces memory use through performing a row-based, approximate matrix factorization.

Wolfe et al.[13] present a framework which fully distributes the EM procedure. In their implementation every node only interacts with parameters relevant to its data and sends messages to other nodes along a junction-tree topology.

Chiang et al.[14] parallel the MIRA[15] online training algorithm with some coordination mechanism among processors. However, there is no detailed theoretic discussion in their paper.

McDonald et al.[16] propose a parameters mixing strategy used in the parallel Perceptron implementation. Their work is very similar to ours, but PA and Perceptron are different online algorithms. So the theoretic proofs of the convergence are actually quite different.

7 Conclusion

In this paper we propose a distributed training strategy with the online PA algorithm for structured learning. We guarantee its convergence with the proposed averaging strategy.

Experimental results show that our method significantly reduces the time cost without decreasing the accuracy. Although the number of iterations increases, less time cost in each iteration also make the training time still less.

In the future, we wish to extend our algorithm to more complex structured learning, such as parsing.

8 Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments. This work was funded by NSFC (No.61003091 and No.61073069).

References

1. Ando, R.K., Zhang, T.: A framework for learning predictive structures from multiple tasks and unlabeled data. J. Mach. Learn. Res. **6** (December 2005) 1817–1853
2. Zhang, Y., Clark, S.: A fast decoder for joint word segmentation and POS-tagging using a single discriminative model. In: Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing. EMNLP '10, Stroudsburg, PA, USA, Association for Computational Linguistics (2010) 843–852
3. Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., Singer, Y.: Online passive-aggressive algorithms. Journal of Machine Learning Research **7** (2006) 551–585
4. Rosenblatt, F.: The perceptron: A probabilistic model for information storage and organization in the brain. Psychological review **65**(6) (1958) 386–408
5. Collins, M.: Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In: Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing. (2002)

6. McDonald, R., Pereira, F., Ribarov, K., Hajič, J.: Non-projective dependency parsing using spanning tree algorithms. Proc. of HLT-EMNLP (2005)
7. Dean, J., Ghemawat, S.: Mapreduce: simplified data processing on large clusters. Commun. ACM **51** (January 2008) 107–113
8. Qiu, X., Zhang, Q., Huang, X.: FudanNLP: A toolkit for chinese natural language processing. In: Proceedings of ACL. (2013)
9. Jin, C., Chen, X.: The fourth international Chinese language processing bakeoff: Chinese word segmentation, named entity recognition and Chinese pos tagging. In: Sixth SIGHAN Workshop on Chinese Language Processing. (2008) 69
10. Chu, C.T., Kim, S.K., Lin, Y.A., Ng, A.Y.: Map-reduce for machine learning on multicore. Architecture **19** (2007) 281
11. Chang, E.Y., Zhu, K., Wang, H., Bai, H., Li, J., Qiu, Z., Cui, H.: Psvm : Parallelizing support vector machines on distributed computers. Change **20**(2) (2007) 1–8
12. Cristianini, N., Shawe-Taylor, J.: An introduction to support Vector Machines: and other kernel-based learning methods. Cambridge Univ Pr (2000)
13. Wolfe, J., Haghighi, A., Klein, D.: Fully distributed em for very large datasets. In: Proceedings of the 25th international conference on Machine learning. ICML '08, New York, NY, USA, ACM (2008) 1184–1191
14. Chiang, D., Marton, Y., Resnik, P.: Online large-margin training of syntactic and structural translation features. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics (2008) 224–233
15. Crammer, K., Singer, Y.: Ultraconservative online algorithms for multiclass problems. Journal of Machine Learning Research **3** (2003) 951–991
16. McDonald, R., Hall, K., Mann, G.: Distributed training strategies for the structured perceptron. In: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics. HLT '10, Stroudsburg, PA, USA, Association for Computational Linguistics (2010) 456–464