The AAAI-17 Workshop on
Distributed Machine Learning
WS-17-08

Distributed Weighted Parameter Averaging for SVM Training on Big Data

Ayan Das, Raghuveer Chanda, Smriti Agrawal, Sourangshu Bhattacharya
Dept. of Computer Science and Engineering.
Indian Institute of Technology, Kharagpur, India
Email: {ayandas84, raghuveer.chanda, smritiagrawal.iitkgp, sourangshu}@gmail.com

Abstract

Two popular approaches for distributed training of SVMs on big data are parameter averaging and ADMM. Parameter averaging is efficient but suffers from loss of accuracy with increase in number of partitions, while ADMM in the feature space is accurate but suffers from slow convergence. In this paper, we report a hybrid approach called weighted parameter averaging (WPA), which optimizes the regularized hinge loss with respect to weights on parameters. The problem is shown to be same as solving SVM in a projected space. We also demonstrate an $O(\frac{1}{N})$ stability bound on final hypothesis given by WPA, using novel proof techniques. Experimental results on a variety of toy and real world datasets show that our approach is significantly more accurate than parameter averaging for high number of partitions. It is also seen the proposed method enjoys much faster convergence compared to ADMM in feature space.

1 Introduction

With the growing popularity of Big Data platforms like Hadoop (Apache Software Foundation 2016) for various machine learning and data analytics applications (Mann et al. 2009; Zinkevich et al. 2010), distributed training of Support Vector Machines (SVMs)(Cortes and Vapnik 1995) on Big Data platforms have become increasingly important. Big data platforms such as Hadoop (Apache Software Foundation 2016) provide simple programming abstraction (Map Reduce), scalability and fault tolerance at the cost of distributed iterative computation being slow and expensive (Mann et al. 2009). Thus, there is a need for SVM training algorithms which are efficient both in terms of the number of iterations and volume of data communicated per iteration.

The problem of distributed training of support vector machines (SVM) (Forero, Cano, and Giannakis 2010) in particular, and distributed regularized loss minimization (RLM) in general (Boyd et al. 2011; Mann et al. 2009), has received a lot of attention in the recent times. Here, the training data is partitioned into M-nodes, each having L datapoints. Parameter averaging (PA), also called "mixture weights" (Mann et al. 2009) or "parallelized SGD" (Zinkevich et al. 2010), suggests solving an appropriate RLM problem on data in each node, and use average of the resultant parameters. Hence, a single distributed iteration is needed. However, as shown in this paper, the accuracy of this approach reduces with increase in number of partitions. Another interesting result described in (Mann et al. 2009) is a bound of $O(\frac{1}{ML})$ on the stability of the final hypothesis, which results in a bound on deviation from optimizer of generalization error.

Another popular approach for distributed RLM is *alternating direction method of multipliers* (ADMM) (Boyd et al. 2011; Forero, Cano, and Giannakis 2010). This approach tries to achieve consensus between parameters at different nodes while optimizing the objective function. It achieves optimal performance irrespective of the number of partitions. However, this approach needs many distributed iterations. Also, number of parameters to be communicated among machines per iteration is same as the dimension of the problem. This can be $\sim$ millions for some practical datasets, e.g. webspam (Chang and Lin 2011).

In this paper, we propose a hybrid approach which uses weighted parameter averaging and learns the weights in a distributed manner from the data.In particular, we derived a novel SVM-like formulation for learning the weights of the weighted parameter averaging (WPA) model. The dual of WPA turns out to be same as SVM dual, with data projected in a lower dimensional space. We propose an ADMM (Boyd et al. 2011) based distributed algorithm (DWPA), and an accelerated version (DWPAacc), for learning the weights.

Another contribution is a $O(\frac{1}{ML})$ bound on the stability of final hypothesis leading to a bound on deviation from optimizer of generalization error. This requires a novel proof technique as both the original parameters and the weights are solutions to optimization problems (section 2.4). Empirically, we show that accuracy of parameter averaging degrades with increase in the number of partitions. Experimental results on real world datasets show that DWPA and DWPAacc achieve better accuracies than PA as the number of partitions increase, while requiring lower number of iterations and time per iteration compared to ADMM.

2 Distributed Weighted Parameter Averaging (DWPA)

In this section, we describe the distributed SVM training problem, the proposed solution approach and a distributed algorithm. We describe a bound on stability of the final hy-

pothesis in section 2.4. Note that, we focus on the distributed SVM problem for simplicity. The techniques described here are applicable to other distributed regularized risk minimization problems.

2.1 Background

Given a training dataset $S = \{(\mathbf{x}_i, y_i) : i = 1, \cdots, ML, y_i \in \{-1, +1\}, \mathbf{x_i} \in \mathbf{R^d}\}$, the linear SVM problem (Cortes and Vapnik 1995) is given by:

$$\min_{\mathbf{w}} \lambda \|\mathbf{w}\|_2^2 + \frac{1}{m} \sum_{i=1}^{ML} loss(\mathbf{w}; (\mathbf{x}_i, y_i)), \tag{1}$$

where, λ is the regularization parameter and the hinge loss is defined as $loss(\mathbf{w}; (\mathbf{x}_i, y_i)) = max(0, 1 - y_i \mathbf{w}^T \mathbf{x}_i)$. The separating hyperplane is given by the equation $\mathbf{w}^T\mathbf{x} + b = 0$. Here we include the bias b within $\mathbf{w}$ by making the following transformation, $\mathbf{w} = \left[\mathbf{w}^T, b\right]^T$ and $\mathbf{x}_i = \left[\mathbf{x}_i^T, 1\right]^T$.

The above SVM problem can be posed to be solved in a distributed manner, which is interesting when the volume of training data is too large to be effectively stored and processed on a single computer.

Let the dataset which is partitioned into M partitions be $(\mathcal{S}_m,\ m = 1, \ldots, M)$, each having L datapoints. Hence, $\mathcal{S} = \mathcal{S}_1 \cup, \ldots, \cup \mathcal{S}_M$, where $\mathcal{S}_m = \{(\mathbf{x}_{ml}, y_{ml})\}, l = 1, \ldots, L$. Under this setting, the SVM problem (Eqn 1), can be stated as:

$$\min_{\mathbf{w}_m, \mathbf{z}} \sum_{m=1}^{M} \sum_{l=1}^{L} loss(\mathbf{w}_m; (\mathbf{x}_{ml}, y_{ml})) + r(\mathbf{z})$$
$$\text{s.t.} \mathbf{w}_m - \mathbf{z} = 0, m = 1, \cdots, M,\ l = 1, \ldots, L$$

where $loss()$ is as described above and $r(\mathbf{z}) = \lambda \|\mathbf{z}\|^2$.This problem is solved in (Boyd et al. 2011) using ADMM (see section 2.3).

Another method for solving distributed RLM problems, called parameter averaging (PA), was proposed by Mann et. al. (Mann et al. 2009), in the context of conditional maximum entropy model.

$$\hat{\mathbf{w}}_m = \operatorname*{argmin}_{\mathbf{w}} \frac{1}{L} \sum_{l=1}^{L} loss(\mathbf{w}; \mathbf{x}_{ml}, y_{ml}) + \lambda \|\mathbf{w}\|^2$$
$$m = 1, \ldots, M$$

be the standard SVM solution obtained by training on partition $\mathcal{S}_m$. Mann et al. (Mann et al. 2009) suggests the approximate final parameter to be the arithmetic mean of the parameters learnt on individual partitions, $(\hat{\mathbf{w}}_m)$. Hence:

$$\mathbf{w}_{PA} = \frac{1}{M} \sum_{m=1}^{M} \hat{\mathbf{w}}_m$$

Zinekevich et al. (Zinkevich et al. 2010) have also suggested a similar approach where $\hat{\mathbf{w}}_m$'s are learnt using SGD. We tried out this approach for SVM. Note that assumptions regarding differentiability of loss function made in (Boyd et al. 2011) can be relaxed in case of convex loss function with an appropriate definition of bregmann divergence using subgradients (see (Mohri, Rostamizadeh, and Talwalkar 2012), section 2.4). The results (reported in section 3) show that the method fails to perform well as the number of partitions increase. This drawback of the above mentioned approach motivated us to propose the weighted parameter averaging method described in the next section.

2.2 Weighted parameter averaging (WPA)

The parameter averaging method uses uniform weight of $\frac{1}{M}$ for each of the M components. One can conceive a more general setting where the final hypothesis is a weighted sum of the parameters obtained on each partition: $\mathbf{w} = \sum_{m=1}^{M} \beta_m \hat{\mathbf{w}}_\mathbf{m}$, where $\hat{\mathbf{w}}_\mathbf{m}$ are as defined above and $\beta_m \in \mathbb{R}, m = 1, \ldots, M$. Thus, $\boldsymbol{\beta} = [\beta_1, \cdots, \beta_M]^T = [\frac{1}{M}, \ldots, \frac{1}{M}]$ achieves the PA setting. Note that Mann et al. (Mann et al. 2009) proposed $\boldsymbol{\beta}$ to be in a simplex. However, no scheme was suggested for learning an appropriate $\boldsymbol{\beta}$.

Our aim is to find the optimal set of weights $\boldsymbol{\beta}$ which attains the lowest regularized loss.

Let $\hat{\mathbf{W}} = [\hat{\mathbf{w}_1}, \cdots, \hat{\mathbf{w}_M}]$, so that $\mathbf{w} = \hat{\mathbf{W}}\boldsymbol{\beta}$. Substituting $\mathbf{w}$ in Eqn. 1, the regularized loss minimization problem becomes:

$$\min_{\boldsymbol{\beta}, \xi} \lambda \|\hat{\mathbf{W}}\boldsymbol{\beta}\|^2 + \frac{1}{ML} \sum_{m=1}^{M} \sum_{i=1}^{l} \xi_{mi} \tag{2}$$
$$\textbf{subject to: } y_{mi}(\boldsymbol{\beta}^T \hat{\mathbf{W}}^T \mathbf{x}_{mi}) \geq 1 - \xi_{mi}, \quad \forall i, m$$
$$\xi_{mi} \geq 0, \quad \forall m = 1, \ldots, M,\ i = 1, \ldots, l$$

Note that, here the optimization is only w.r.t. $\boldsymbol{\beta}$ and $\xi_{m,i}$. $\hat{\boldsymbol{W}}$ is a pre-computed parameter. Next we can derive the dual formulation by writing the lagrangian and eliminating the primal variables. The Lagrangian is given by:

$$\mathcal{L}(\boldsymbol{\beta}, \xi_{mi}, \alpha_{mi}, \mu_{mi}) = \lambda \|\hat{\mathbf{W}}\boldsymbol{\beta}\|^2 + \frac{1}{ML} \sum_{m,i} \xi_{mi}$$
$$+ \sum_{m,i} \alpha_{mi}(y_{mi}(\boldsymbol{\beta}^T W^T \mathbf{x}_{mi}) - 1 + \xi_{mi}) - \sum_{m,i} \mu_{mi} \xi_{mi}$$

Differentiating the Lagrangian w.r.t. $\boldsymbol{\beta}$ and equating to zero, we get:

$$\boldsymbol{\beta} = \frac{1}{2\lambda} (\hat{\mathbf{W}}^T \hat{\mathbf{W}})^{-1} (\sum_{m,i} \alpha_{mi} y_{mi} \hat{\mathbf{W}}^T \mathbf{x}_{mi})$$

Differentiating $\mathcal{L}$ w.r.t. ξ_{mi} and equating to zero, $\forall i \in 1, \cdots, L$ and $\forall m \in 1, \cdots, M$, implies $\frac{1}{ML} - \alpha_{mi} - \mu_{mi} = 0$. Since $\mu_{mi} \geq 0$ and $\alpha_{mi} \geq 0$, $0 \leq \alpha_{mi} \leq \frac{1}{ML}$. Substituting the value of $\boldsymbol{\beta}$ in the Lagrangian $\mathcal{L}$, we get the dual problem:

$$\min_{\boldsymbol{\alpha}} \mathcal{L}(\boldsymbol{\alpha}) = \sum_{m,i} \alpha_{mi} - \frac{1}{4\lambda}$$
$$\sum_{m,i} \sum_{m',j} \alpha_{mi} \alpha_{m'j} y_{mi} y_{m'j} (\mathbf{x}_{mi}^T \hat{\mathbf{W}} (\hat{\mathbf{W}}^T \hat{\mathbf{W}})^{-1} \hat{\mathbf{W}}^T \mathbf{x}_{m'j})$$
$$\textbf{subject to: } 0 \leq \alpha_{mi} \leq \frac{1}{ML}$$
$$\forall i \in 1, \cdots, L, m \in 1, \cdots, M$$

Note that this is equivalent to solving SVM using the projected datapoint $(\mathcal{H}\mathbf{x}_{mi}, y_{mi})$, instead of $(\mathbf{x}_{mi}, y_{mi})$, where $\mathcal{H} = \hat{\mathbf{W}}(\hat{\mathbf{W}}^T\hat{\mathbf{W}})^{-1}\hat{\mathbf{W}}^T$, which is the projection on column space of $\hat{\mathbf{W}}$. Hence the performance of the method is expected to depend on size and orientation of the column space of $\hat{\mathbf{W}}$. Next, we describe distributed algorithms for learning $\boldsymbol{\beta}$.

2.3 Distributed algorithms for WPA using ADMM

In the distributed setting, we assume the presence of a central (master) computer which stores and updates the final hypothesis. The partitions of training set $\mathcal{S}_1, \ldots, \mathcal{S}_M$ are distributed to M (slave) computers, where the local optimizations are performed. The master needs to communicate to slaves and vice versa. However, no communication between slaves is necessary. Thus, the underlying networks has a star topology, which is also easily implemented using Big data platforms like Hadoop (Apache Software Foundation 2016).

Let $\boldsymbol{\gamma}_m$, for $m = 1, \cdots, M$ be the weight values at the M different nodes and $\boldsymbol{\beta}$ be the value of the weights at the central server. The formulation given in Eqn. 2 can be written as:

$$\min_{\gamma_m, \beta} \frac{1}{ML} \sum_{m=1}^{M} \sum_{l=1}^{L} loss(\hat{W}\boldsymbol{\gamma}_m; \mathbf{x}_{ml}, y_{ml}) + r(\boldsymbol{\beta})$$
$$\text{s.t.} \boldsymbol{\gamma}_m - \boldsymbol{\beta} = 0, m = 1, \cdots, M,$$

where $r(\boldsymbol{\beta}) = \lambda\|\hat{\mathbf{W}}\boldsymbol{\beta}\|^2$.The augmented lagrangian for the above problem is:

$$L(\boldsymbol{\gamma}_m, \boldsymbol{\beta}, \boldsymbol{\lambda}) = \frac{1}{ML} \sum_{m=1}^{M} \sum_{l=1}^{L} loss(\hat{W}\boldsymbol{\gamma}_m; \mathbf{x}_{ml}, y_{ml}) +$$
$$r(\boldsymbol{\beta}) + \sum_{i=1}^{M} \frac{\rho}{2}\|\boldsymbol{\gamma}_m - \boldsymbol{\beta}\|^2 + \sum_{i=1}^{M} \boldsymbol{\psi}_m^T(\boldsymbol{\gamma}_m - \boldsymbol{\beta}),$$

where $\boldsymbol{\psi}_m$ is the lagrange multiplier vector corresponding to m^{th} constraint.

Let $\mathbf{A}_m \in \boldsymbol{R}^{L\times d} = -\mathbf{diag}(\mathbf{y}_m)\mathbf{X}_m\hat{W}$. Using results from (Boyd et al. 2011), the ADMM updates for solving the above problem can derived as:

$$\boldsymbol{\gamma}_m^{k+1} := \operatorname*{argmin}_{\boldsymbol{\gamma}}(loss(\mathbf{A}_i\boldsymbol{\gamma}) + (\rho/2)\|\boldsymbol{\gamma}_m l - \boldsymbol{\beta}^k + \mathbf{u}_m^k\|_2^2)$$
$$\boldsymbol{\beta}^{k+1} := \operatorname*{argmin}_{\boldsymbol{\beta}}(r(\boldsymbol{\beta}) + (M\rho/2)\|\boldsymbol{\beta} - \overline{\boldsymbol{\gamma}}^{k+1} - \overline{\mathbf{u}}^k\|_2^2)$$
$$\mathbf{u}_m^{k+1} = \mathbf{u}_m^k + \boldsymbol{\gamma}_m^{k+1} - \boldsymbol{\beta}^{k+1}.$$

where, $\mathbf{u}_m = \frac{1}{\rho}\boldsymbol{\psi}_m$, $\overline{\boldsymbol{\gamma}} = \frac{1}{M}\sum_{m=1}^{M}\boldsymbol{\gamma}_m$ and $\overline{\mathbf{u}} = \frac{1}{M}\sum_{m=1}^{M}\mathbf{u}_m$ and the superscript k denotes the iteration counts. Algorithm 1 describes the full procedure.

A heuristic called *overrelaxation* (Boyd et al. 2011) is often used for improving the convergence rate of ADMM. For overrelaxation, the updates for $\boldsymbol{\beta}^k$ (line 1 and $\mathbf{u}_m^k$ (line 1) are obtained by replacing $\overline{\boldsymbol{\gamma}}^k$ with $\hat{\boldsymbol{\gamma}}_m^k = \alpha \times \boldsymbol{\gamma}_m^k + (1-\alpha) \times \boldsymbol{\beta}^{k-1}$, in algorithm 1. We implemented this heuristic for both DSVM and DWPA. We call them accelarated DSVM (DSVMacc) and accelarated DWPA (DWPAacc).

Algorithm 1 Distributed Weighted Parameter Averaging (DWPA)

input : Partitioned datasets $\mathcal{S}_m$, SVM parameter learnt for each partition $\hat{\mathbf{w}}_m, \forall m = 1, \cdots, M$
output : Optimal weight vector $\boldsymbol{\beta}$

1: Initialize $\boldsymbol{\beta} = \mathbf{1}, \boldsymbol{\gamma}_m = \mathbf{1}, \mathbf{u}_m = \mathbf{1}, \forall m \in \{1, \cdots, M\}$
2: **while** $k < T$ **do**
3: {*Executed on slaves*}
4: **for** $m \leftarrow 1$ to M **do**
5: $\boldsymbol{\gamma}_m^k := \operatorname{argmin}_{\boldsymbol{\gamma}_m}(1^T(A_m\boldsymbol{\gamma}_m + 1)_+ + \rho/2\|\boldsymbol{\gamma}_m^{k-1} - \boldsymbol{\beta}^{k-1} - \mathbf{u}_m^{k-1}\|_2^2)$
6: **end for**
7: {*Executed on master*}
8: $\boldsymbol{\beta}^k := \frac{1}{2\lambda}(\hat{W}^T\hat{W} + M\rho I_m)^{-1}M\rho(\overline{\boldsymbol{\gamma}}^k + \overline{\mathbf{u}}^{k-1})$
9: **for** $m \leftarrow 1$ Tto M **do**
10: $\mathbf{u}_m^k = \mathbf{u}_m^{k-1} + \boldsymbol{\gamma}_m^k - \boldsymbol{\beta}^k$
11: **end for**
12: **end while**

2.4 Bound on stability of WPA

In this section, we derive a bound of $\mathcal{O}(\frac{1}{ML})$ on stability of the final hypothesis returned by WPA algorithm described in Eqn. 2. A similar bound was derived by Mann et al. (Mann et al. 2009) on the stability of PA. This leads to a $O(\frac{1}{ML})$ bound on deviation from optimizer of generalization error. Let $S = \{S_1, \cdots, S_M\}$ and $S' = \{S'_1, \cdots, S'_M\}$ be two datasets with M partitions and L datapoints per partition, differing in only one datapoint. Hence, $S_m = \{z_{m1}, \cdots, z_{mL}\}, S'_m = \{z'_{m1}, \cdots, z'_{mL}\}$ where $z_{ml} = (\mathbf{x}_{ml}, y_{ml}), z'_{ml} = (\mathbf{x}'_{ml}, y'_{ml})$. Further, $S_1 = S'_1, \cdots, S_{M-1} = S'_{M-1}$, and S_M and S'_M differs at single point z_{ML} and z'_{ML}. Also, let $\|\mathbf{x}\| \leq R, \forall \mathbf{x}$.

Let $\hat{\mathbf{W}} = [\hat{\mathbf{w}}_{S_1}, \cdots, \hat{\mathbf{w}}_{S_M}]$ and $\hat{W}' = [\hat{\mathbf{w}}_{S'_1}, \cdots, \hat{\mathbf{w}}_{S'_M}]$ where, $\hat{\mathbf{w}}_{S_i} = \operatorname{argmin}_{\mathbf{w}} \lambda\|\mathbf{w}\|^2 + \frac{1}{L}\sum_{i\in S_i} max(0, 1 - y\mathbf{w}^T\mathbf{x})$. We assume $\|\hat{\mathbf{W}}\|_F = \|\hat{\mathbf{W}}'\|_F = 1$. Hence, $\|\hat{\mathbf{w}}_m\|^2 = \|\hat{\mathbf{w}}'_m\|^2 = \frac{1}{M}, \forall m \in \{1, \cdots, M\}$.

We also define the following quantities:

$$\boldsymbol{\beta} = \operatorname*{argmin}_{\boldsymbol{\beta}} \lambda\|\hat{\mathbf{W}}\boldsymbol{\beta}\|^2 + \frac{1}{ML}\sum_{i=1}^{M}\sum_{z\in S_i} max(0, 1 - y(\hat{\mathbf{W}}\boldsymbol{\beta})^T\mathbf{x})$$

$$\boldsymbol{\beta}' = \operatorname*{argmin}_{\boldsymbol{\beta}} \lambda\|\hat{\mathbf{W}}'\boldsymbol{\beta}\|^2 + \frac{1}{ML}\sum_{i=1}^{M}\sum_{z'\in S'_i} max(0, 1 - y'(\hat{\mathbf{W}}\boldsymbol{\beta})^T\mathbf{x}')$$

$$\tilde{\boldsymbol{\beta}} = \operatorname*{argmin}_{\boldsymbol{\beta}} \lambda\|\hat{\mathbf{W}}'\boldsymbol{\beta}\|^2 + \frac{1}{ML}\sum_{i=1}^{M}\sum_{z\in S_i} max(0, 1 - y(\hat{\mathbf{W}}'\boldsymbol{\beta})^T\mathbf{x})$$

Also, let $\mathbf{w} = \hat{\mathbf{W}}\boldsymbol{\beta}$, $\mathbf{w}' = \hat{\mathbf{W}}'\boldsymbol{\beta}'$ and $\tilde{\mathbf{w}} = \hat{\mathbf{W}}'\tilde{\boldsymbol{\beta}}$.

We are interested in deriving a bound on $\|\mathbf{w}-\mathbf{w}'\|$, which decompose as: $\|\mathbf{w}-\mathbf{w}'\| \leq \|\mathbf{w}-\tilde{\mathbf{w}}\| + \|\tilde{\mathbf{w}}-\mathbf{w}'\|$. Intuitively, the first term captures the change from $\hat{\mathbf{W}}$ to $\hat{\mathbf{W}}'$ and second term captures change in dataset. We proved that $\|\tilde{\mathbf{w}}-\mathbf{w}'\|$ is $\mathcal{O}(\frac{1}{ML})$ by showing bounds on $\|\mathbf{w}-\tilde{\mathbf{w}}\|$ which require bounds on $\|\boldsymbol{\beta}-\tilde{\boldsymbol{\beta}}\|$ and $\|\hat{\mathbf{W}}-\hat{\mathbf{W}}'\|$.

3 Experimental Results

In this section, we experimentally analyze and compare the methods proposed, *distributed weighted parameter averaging* (DWPA) and accelerated DWPA (DWPAacc) described in section 2.3, with *parameter averaging* (PA) (Mann et al. 2009), Distributed SVM (DSVM) using ADMM, and accelerated DSVM (DSVMacc) (Boyd et al. 2011).

For our experimentation all the algorithms have been implemented in Scala using Apache Spark libraries on a 20 node CDH 5.1 cluster. Each node is a HP Proliant server with;

- 2 hex-core processors
- 128 GB RAM
- 4 TB Hard Disk
- OS: CentOS 6.5

We used real world datasets (described in table 1) for our experiments. Real world datasets were obtained from LIBSVM website (Chang and Lin 2011). Samples for real world datasets were selected randomly. The datasets were selected to have various ranges of feature count and sparsity.

3.1 Comparison of Accuracies

In this section,we compare accuracies obtained by various algorithms on real world datasets. Figure 1 reports test set accuracies for PA, WPA and SVM on three real world datasets with varying size of partitions. It is clear that performance of PA degrades dramatically as the number of partition increases..

We also observe that performance of WPA improves with increase in number of partitions. This is due to fact that dimension of space on which $\boldsymbol{x}_{ml}$'s are projected using $\mathcal{H}$ (section 2.2) increases, thus reducing the information loss caused by projection. Finally, as expected WPA performs slightly worse than DSVM.

3.2 Convergence Analysis and Time comparison

In this section, we compare the convergence properties of DSVM, DSVMacc, DWPA, and DWPAacc. In Figure 2, we show variation of primal residual (disagreement between parameters on various partitions) with iterations. It is clear that DWPA and DWPAacc show much lesser disagreement compared to DSVM and DSVMacc, thus showing faster convergence.

In Figure 3,we show the variation of test set accuracy with iterations. The same behaviour is apparent here, with testset accuracy of DWPA and DWPAacc converging much faster than DSVM and DSVMacc. One of the reasons is also that DWPA has an obvious good starting point of $\boldsymbol{\beta} = [\frac{1}{M}, \ldots, \frac{1}{M}]$ corresponding to PA.

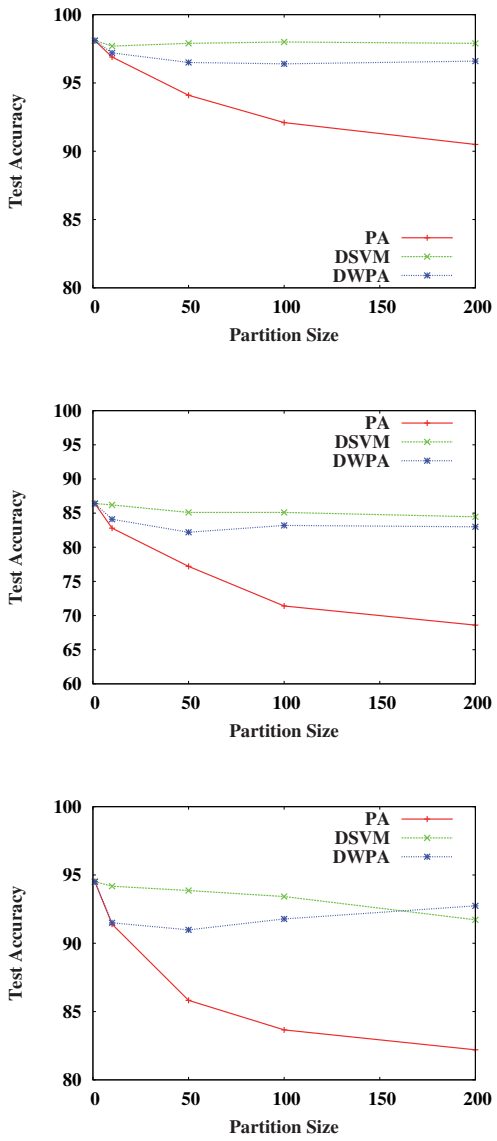

Figure 1: Variation of accuracy with number of partitions. From top *gisette(first), epsilon(second)* and *real-sim(third)* for partition size 1, 10, 50, 100 and 200

3.3 Convergence time

In this section, we report the time taken to achieve the best testset accuracy for the different partitionings of the datasets using the different algorithms. Table 2 and 3 summarizes the results. Here, we assume that each data partition in already distributed to a different slave node.

Table 4 reports the average time taken in seconds by DWPA and DSVM for completing one iteration as a function

Dataset Name	Number of training instances	Number of test instances	Number of features
real-sim	3,000	5,000	20,958
gisette	6,000	1,000	5,000
webspam	320,000	33,000	16,609,143
epsilon	400,000	100,000	2,000
url	2,396,130	15,000	3,231,9614
kdda	8,407,752	510,302	20,216,830
kddb	19,264,097	748,401	29,890,095
splice-site	10,000,000	4,627,840	11,725,480

Table 1: Training and test dataset size

	webspam			
No. of partitions	ADMM	ADMMacc	DWPA	DWPAacc
10	192(93.4)	241(93.4)	199(92.06)	93(92.06)
20	271(93.4)	271(93.4)	354(91.86)	210.5(91.86)
40	469.5(93.4)	429(93.4)	408(91.97)	184(91.97)
80	748(93.4)	680(93.4)	320(92.30)	176(92.30)
	epsilon			
10	1044(89.8)	1044(89.8)	457(89.2)	417(89.2)
20	840.15(89.8)	747.08(89.8)	356(89)	296.5(89)
40	915.16(89.8)	915.16(89.8)	277.8(88.67)	268.24(88.67)
80	678(89.8)	498(89.8)	225(88.2)	203(88.2)

Table 2: Time taken in seconds (best test set accuracy) for DWPA, ADMM and accelerated versions of DWPA and ADMM(DWPAacc and ADMMacc respectively)

Dataset Name	ADMM		DWPA	
	Test Accuracy	Total time	Test Accuracy	Total time
epsilon	89.8	12.7	89.2	7.6
webspam	93.4	5.5	92.3	5.33
gisette	97	50	97.4	2
url	99.46	126	96.3	27.5
kdda	86.78	104.2	86.78	3.1
kddb	86.2	49.7	86.2	5.7
splice-site	99.58	239.5	99.58	119.7

Table 3: Comparison of running time (*min*) and testset accuracy for the two algorithms.

	DWPA		DSVM	
Number of partitions	epsilon	webspam	epsilon	webspam
10	2	2	72	7
20	3.5	3.5	43.61	5
40	2.24	4	34.78	4.5
80	2	4	12	5

Table 4: Average time per iteration(in seconds)

of number of partitions. It is clear that DWPA takes much lesser time due to much smaller number of variables in the local optimization problem (Feature dimensions for DSVM, number of partitions for DWPA). There is slight increase in time per iteration with increase in number of partitions due to increase in number of variables.

4 Conclusion and Future Work

We propose a novel approach for training SVM in a distributed manner by learning an optimal set of weights for combining the SVM parameters independently learnt on partitions of the entire dataset. Experimental results show that our method is much more accurate than *parameter averaging* and is much faster than training SVM in feature space. Moreover, our method reaches an accuracy close to that of SVM trained in feature space in a much shorter time. We propose a novel proof to show that the stability final SVM parameter learnt using DWPA is $\mathcal{O}(\frac{1}{ML})$. Also, our method requires much less network band-width as compared to DSVM when the number of features for a given dataset is

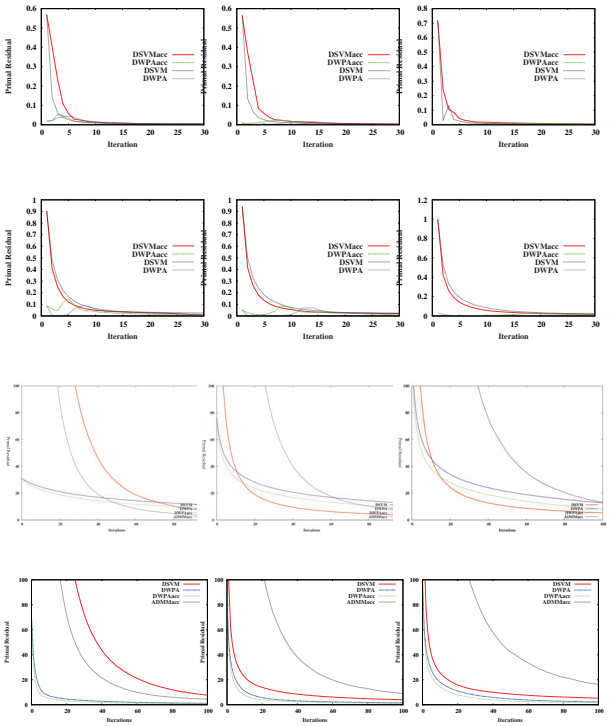

Figure 2: Convergence of primal residual for **real-sim** (first) **gisette** (second) **epsilon** (third) **webspam** (fourth)

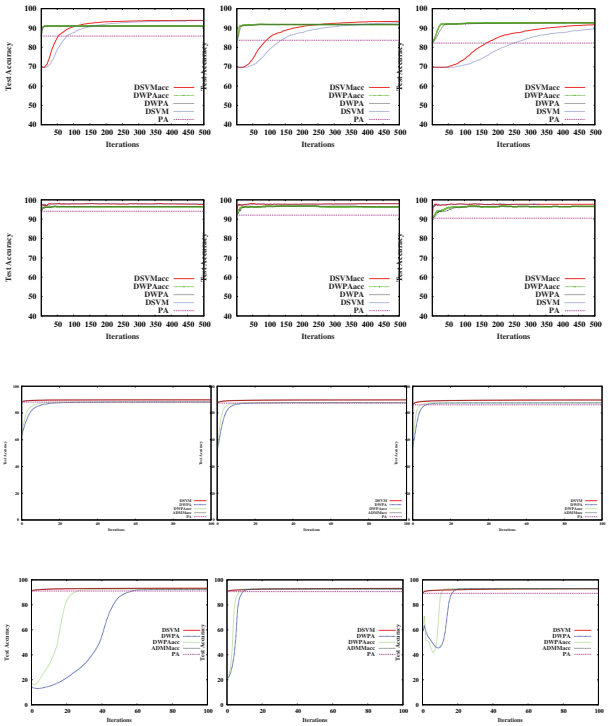

Figure 3: Convergence of test accuracy for **real-sim** (first) **gisette** (second) **epsilon** (third) **webspam** (fourth)

very large as compared to the number of partitions, which is the usual scenerio for Big Data. In future we would also like to compare with other distributed learning algorithms, such as CoCoA (Jaggi et al. 2014) and CoCoA+ (Ma et al. 2015).

References

Apache Software Foundation. 2016. Hadoop.

Boyd, S.; Parikh, N.; Chu, E.; Peleato, B.; and Eckstein, J. 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.* 3(1):1–122.

Chang, C.-C., and Lin, C.-J. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology* 2:27:1–27:27. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

Cortes, C., and Vapnik, V. 1995. Support-vector networks. *Machine Learning* 20(3):273–297.

Forero, P. A.; Cano, A.; and Giannakis, G. B. 2010. Consensus-based distributed support vector machines. *J. Mach. Learn. Res.* 11:1663–1707.

Jaggi, M.; Smith, V.; Takáč, M.; Terhorst, J.; Krishnan, S.; Hofmann, T.; and Jordan, M. I. 2014. Communication-efficient distributed dual coordinate ascent. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, NIPS'14, 3068–3076. Cambridge, MA, USA: MIT Press.

Ma, C.; Smith, V.; Jaggi, M.; Jordan, M. I.; Richtrik, P.; and Takc, M. 2015. Adding vs. Averaging in Distributed Primal-Dual Optimization. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *JMLR Proceedings*, 1973–1982. JMLR.org.

Mann, G.; McDonald, R.; Mohri, M.; Silberman, N.; and Walker, D. 2009. Efficient large-scale distributed training of conditional maximum entropy models. In Bengio, Y.; Schuurmans, D.; Lafferty, J.; Williams, C. K. I.; and Culotta, A., eds., *Advances in Neural Information Processing Systems 22*. Neural Information Processing Systems, 2010. 1231–1239.

Mohri, M.; Rostamizadeh, A.; and Talwalkar, A. 2012. *Foundations of Machine Learning*. The MIT Press.

Zinkevich, M.; Weimer, M.; Smola, A. J.; and Li, L. 2010. Parallelized stochastic gradient descent. In *NIPS*, 2595–2603.