Robust Distributed Training of Linear Classifiers Based on Divergence Minimization Principle

Junpei Komiyama, Hidekazu Oiwa, and Hiroshi Nakagawa

The University of Tokyo,
7-3-1, Hongo, Bunkyo-ku, Tokyo, 113-0033, Japan
junpei@komiyama.info
oiwa@r.dl.itc.u-tokyo.ac.jp
nakagawa@dl.itc.u-tokyo.ac.jp

Abstract. We study a distributed training of a linear classifier in which the data is separated into many shards and each worker only has access to its own shard. The goal of this distributed training is to utilize the data of all shards to obtain a well-performing linear classifier. The iterative parameter mixture (IPM) framework (Mann et al., 2009) is a state-of-the-art distributed learning framework that has a strong theoretical guarantee when the data is clean. However, contamination on shards, which sometimes arises in real world environments, largely deteriorates the performances of the distributed training. To remedy the negative effect of the contamination, we propose a divergence minimization principle for the weight determination in IPM. From this principle, we can naturally derive the Beta-IPM scheme, which leverages the power of robust estimation based on the beta divergence. A mistake/loss bound analysis indicates the advantage of our Beta-IPM in contaminated environments. Experiments with various datasets revealed that, even when 80% of the shards are contaminated, Beta-IPM can suppress the influence of the contamination.

1 Introduction

A linear classifier is one of the most fundamental concepts in the field of machine learning. Online learning algorithms [20,5,6] are able to train linear classifiers effectively. An online algorithm sequentially processes data points, and thus, it requires all data to be accessible from a single machine. While the training on a single machine is of its own importance, training in distributed environments has attracted increasing interest [1,10,16]. In such environments, data is divided up into disjoint sets of shards and each worker has access to only one shard.

Iterative Parameter Mixture (IPM) [16,17] is a state-of-the-art distributed training framework, which involves a master node and worker nodes. Advantages of IPM lies in its communication efficiency and simplicity: in each epoch, each worker trains a model in parallel on his own shard, and the master mixes the training results (Fig. 1).

IPM implicitly assumes that each shard is noiseless. However, it is not always the case: there can be some adversarially or randomly labelled data in some distributed learning scenarios. For example, web mail systems possibly involve some users who

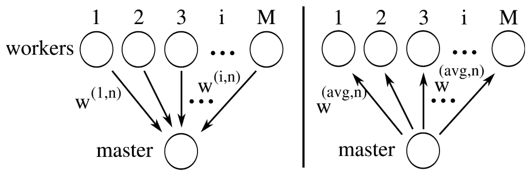

Fig. 1. Illustration of the communication in IPM

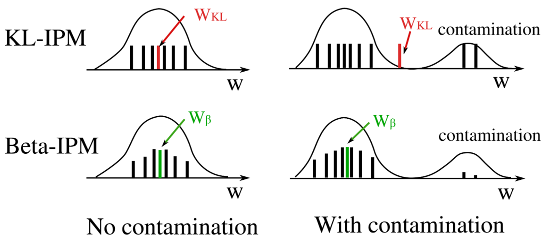

Fig. 2. Illustrative example of KL-IPM and Beta-IPM. Each horizontal line represents a parameter space, and each vertical line represents a parameter returned by a worker, the height of which is proportional to the mixing weight. While KL-IPM equally weights all the parameter vectors, Beta-IPM adaptively weights each parameter as described later.

adversarially labels spams and non-spams, or incorrect data formats [9] lead to corrupted classification result. Let us call such flawed data contamination. We verified that, the performance of the trained classifier is deteriorated by contamination.

Meanwhile, IPM has freedom in how to weight the individual workers' results. If some shards are known to be contaminated, we can avoid the effect of these shards by setting their weights to zero. However, it is unlikely that there will be prior knowledge about which shards are contaminated, and thus, the strategy we should take is to weight seemingly contaminated results less on the basis of their statistical anomalousness.

With this in mind, we propose a weight determination principle based on a divergence minimization criterion. This criterion reinterprets the most straightforward choice, which is to weight each worker equally, as the minimization of the Kullback-Leibler divergence (KL-IPM). On the other hand, the beta divergence, which is the extension of the KL divergence, provides robust inference against contamination. We propose the weight determination formula by minimizing the beta divergence (Beta-IPM). Moreover, We prove a mistake/loss bound of IPM. This theoretical result shows that, by weighting less heavily to contaminated shards with Beta-IPM we can suppress the upper bound of losses over training. The difference between KL-IPM and Beta-IPM is illustrated in Fig. 2. Finally, an empirical evaluation on various datasets confirms that Beta-IPM remedies the effect of contamination.

2 Related Work

2.1 Distributed Training of Linear Models

Distributed training frameworks for linear models have been studied in the literature. Asynchronous updates are sets of models in which all workers simultaneously operate on shared parameters. There is a long line of work related to asynchronous updates from the 1980s [22] onwards [24,12]. A problem of the shared memory resource it that, it does not scale with many shards because the communication cost is proportional to the number of updates.

The distributed gradient method [4] is a distributed extension of the gradient descent method, which optimizes some smooth function by taking steps proportional to the negative gradient. In the distributed gradient method, individual workers compute partial gradients based on their shards, which are then summed to make an update.

In the IPM method, each worker operates independently and shares parameters after each worker finishes an iteration. A master mixes the parameters of the workers with weights whose sum is normalized. IPM is used in many linear and regression models, such as logistic regression [16], structured perceptron [17], etc. McDonald et al. [17] proved that IPM with perceptron has mistake bound in a linearly separable case (i.e., the case in which every data point is correctly classified by some classifier). Moreover, Hall et al. [14] empirically compared the asynchronous updates, the distributed gradient method and IPM using large-scale click-through data. The results show that IPM combined with the perceptron [20] performed the best.

2.2 Robust Training Against Flawed Data

Detection of spam and malicious activities is an important problem in the highly distributed web industry [18,7]. In addition, poorly formatted data [19,9] causes a considerable problem in distributed training. Despite significant efforts made at removing such flawed data, there still is a need for robust models. Robust models are used in many fields, including multi-task learning [13,23] and sensor networks [3].

The problem of contamination in distributed data can be broken down into two cases: the first case is when the contamination is scattered across every shard, and the second case is when some shards are clean while others are contaminated. Studies on the robustness of online learning algorithms [5,6,15] have mainly dealt with the first case, which considers a single data repository affected by noise. In this case, the clean data are hard to distinguish from the noise. Instead, we consider the second case and show that a significant improvement is possible by putting less importance on statistically extraordinary shards which are likely to be wrongly labeled or corrupted.

We note that, Daumé III et al. [8] proposed a distributed learning algorithm with adversarially distributed data. Their definition of adversarially distributed data is different from our adversarial noise: while they considered separable data with an adversary who can generate an arbitrary imbalance among shards, we consider an adversarial attacker that can harm the separability assumption by maliciously labelling.

Algorithm 1. Iterative Parameter Mixture (IPM)

```
1: Shards: $\mathcal{T}_1, ..., \mathcal{T}_M$, $\mathbf{w}^{(\mathrm{avg},0)} = \mathbf{0}$
2: for $n = 1, ..., N$ do
3:     $\mathbf{w}^{(i,n)} = \mathrm{SingleIterationTrain}(\mathcal{T}_i, \mathbf{w}^{(\mathrm{avg},n-1)})$
4:     $\mathbf{w}^{(\mathrm{avg},n)} = \sum_i \alpha_{i,n} \mathbf{w}^{(i,n)}$
5: end for
```

3 Problem Setup

We consider a binary classification. Let $\mathcal{X} \in \mathbb{R}^d$ be the input space and $\mathcal{Y} = \{-1, 1\}$ be the output space. A data point is defined as an input-output tuple $(\mathbf{x}, y) \in \mathcal{X} \times \mathcal{Y}$. A linear classifier with parameter vector $\mathbf{w}$ predicts an output as $\hat{y} = \mathrm{sign}(\mathbf{w} \cdot \mathbf{x})$. The goal of our distributed training framework is to find the parameter vector $\mathbf{w}$ which explains the whole data most.

In distributed training, the training data is divided into M non-overlapping subsets (shards), and a shard is assigned to each worker. There also is a master who integrates the results of workers. Training based on IPM (Algorithm 1) goes as follows. In each epoch $n = 1, 2, ..., N$, each worker i independently does a single iteration of training its own parameters $\mathbf{w}^{(i,n)}$, which are then sent to the master. The master waits until all workers finish their training before it computes a mixed parameter $\mathbf{w}^{(\mathrm{avg},n)}$, which is a weighted sum of the trainers' parameters. The weight $\alpha_{i,n}$ of each worker i in each epoch n can be chosen arbitrarily as long as it is normalized (i.e. $\sum_{i=1}^{M} \alpha_{i,n} = 1$). Later in Section 4 we propose weight determination formulas that we call KL-IPM and Beta-IPM. At the end of the epoch, the mixed parameters $\mathbf{w}^{(\mathrm{avg},n)}$ are sent back to the workers, who in the next epoch start the new single iteration training based on the received mixed parameter. After N epochs have been completed, the master outputs the final parameter vector (a linear classifier).

3.1 IPM Combined with Online Algorithms

We use online learning algorithms in single iteration training ("SingleIterationTrain" in Algorithm 1). We specifically deal with the perceptron [20] and the Passive Aggressive (PA) method [5]. Section 5 describes that, IPM combined with perceptron and PA is able to extend the theoretical guarantee of these single-machine online algorithms.

4 Divergence Minimization Principle

In this section, we describe our main proposal, which is how to determine the weights based on the divergence minimization principle. Section 4.1 describes our statistical assumptions and the divergence minimization principle. Section 4.2 describes the KL and beta divergences, and Section 4.3 shows KL-IPM and Beta-IPM formulas. Section 4.4 demonstrates the behavior of KL-IPM and Beta-IPM with a simple example.

4.1 Statistical Assumption And Divergence Minimization Principle

The statistical assumption is as follows: the parameters returned by the workers in each epoch n should be drawn from a Gaussian distribution Q_n. Our proposal is that, in each epoch n the mixed parameter vector $\mathbf{w}^{(\text{avg},n)}$ is determined to be the mean of the Gaussian Q_n. However, the parameters are actually drawn from P_n, which possibly contains contamination. Namely, $\mathbf{w}^{(i,n)} \sim P_n$. The mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ of Q_n are determined in such a way as to minimize the divergence:

$$\arg\min_{\boldsymbol{\mu},\boldsymbol{\Sigma}} D(P_n||Q_n(\boldsymbol{\mu},\boldsymbol{\Sigma})), \tag{1}$$

where D is the divergence between P_n and Q_n. If we use a robust divergence that suppresses the influence of contamination, we are able to estimate the true $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$.

4.2 KL and Beta Divergences

The KL divergence is the most basic measure that indicates the deviation of a distribution from another distribution. The KL divergence between two probability distributions P and Q on $\mathbb{R}^d$ is defined as

$$D_{KL}(P||Q) = \int P(\mathbf{w}) \log \frac{P(\mathbf{w})}{Q(\mathbf{w})} d\mathbf{w}, \tag{2}$$

which is non-negative and equal to zero if and only if $P = Q$ almost everywhere. While the KL divergence is of fundamental importance in information theory, it is not robust to the contamination of outliers.

The beta divergence, which was introduced by [2] and [11], is parameterized by a real parameter $\beta > 0$. The beta divergence between P and Q is defined as

$$D_\beta(P||Q) = \int \left\{ P(\mathbf{w}) \frac{P^\beta(\mathbf{w}) - Q^\beta(\mathbf{w})}{\beta} \right\} - \frac{P^{\beta+1}(\mathbf{w}) - Q^{\beta+1}(\mathbf{w})}{\beta+1} d\mathbf{w}. \tag{3}$$

When $\beta \to 0$, the beta divergence is consistently defined as $\lim_{\beta\to 0} D_\beta(P||Q) = D_{KL}(P||Q)$. Therefore, the beta divergence can be considered as an extension of the KL divergence. One of the main motivations of investigating the beta divergence is to devise a robust inference against contamination. That is, the beta divergence between two distributions P and Q remains undisturbed by some fraction of the contamination in P. β is a trade-off parameter. The bigger β is, the more robust and less computationally effective the divergence becomes.

4.3 KL-IPM and Beta-IPM

KL-IPM: KL-IPM is a weight determination formula in IPM that equally weights each worker. Namely,

$$\alpha_{i,n} = \frac{1}{M}. \tag{4}$$

However, if flawed data contaminate some of the shards, the performance of KL-IPM deteriorates. To remedy this problem, we derive Beta-IPM that minimizes the beta divergence.

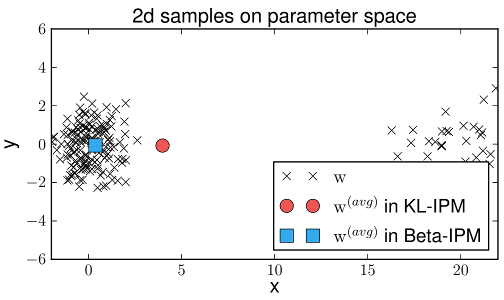

Fig. 3. Illustration of a two-dimensional example. Each of the 100 crosses represents the parameters $\mathbf{w} \sim P$. The 80 crosses are from the true distribution (Gaussian with $\boldsymbol{\mu} = (0,0)^\top$ and $\boldsymbol{\Sigma} = \mathrm{diag}(1,1)$). The other 20 crosses are contamination and generated from a false distribution (Gaussian with $\boldsymbol{\mu} = (20,0)^\top$ and $\boldsymbol{\Sigma} = \mathrm{diag}(2,2)$). The large red circle is the simple mean of all parameters, determined by KL-IPM (Equation (4)). The large blue square is the mixed parameter determined by Beta-IPM with $\beta = 0.1$ (Equation (6)).

Beta-IPM: Let $\boldsymbol{\mu}_c$ and $\boldsymbol{\Sigma}_c$ are respectively the empirical mean and covariance of the parameter vectors $\{\mathbf{w}^{(i,n)}\}$ defined as

$$\boldsymbol{\mu}_c = \frac{1}{M}\sum_{i=1}^{M} \mathbf{w}^{(i,n)}, \text{ and } \boldsymbol{\Sigma}_c = \frac{1}{M}\sum_{i=1}^{M} (\mathbf{w}^{(i,n)} - \boldsymbol{\mu}_c)(\mathbf{w}^{(i,n)} - \boldsymbol{\mu}_c)^\top. \tag{5}$$

Beta-IPM is defined as a weight determination formula in IPM that in each epoch n chooses weight $\alpha_{i,n}$ as follows:

$$\alpha_{i,n} = \frac{\exp S(\mathbf{w}^{(i,n)}|\boldsymbol{\mu}_c, \frac{1}{\beta}\boldsymbol{\Sigma}_c)}{\sum_{j=1}^{M} \exp S(\mathbf{w}^{(j,n)}|\boldsymbol{\mu}_c, \frac{1}{\beta}\boldsymbol{\Sigma}_c)}, \tag{6}$$

where $S(\mathbf{w}^{(i,n)}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = -(1/2)(\mathbf{w}^{(i,n)} - \boldsymbol{\mu})^\top \Sigma^{-1}(\mathbf{w}^{(i,n)} - \boldsymbol{\mu})$ is the exponent part of the Gaussian. Namely, each weight of a shard is determined by the distance of the parameter vector from the mean. Beta-IPM is parameterized by $\beta \geq 0$ and is equivalent to KL-IPM when $\beta \to 0$ because the covariance $(1/\beta)\boldsymbol{\Sigma}_c$ in (6) becomes infinitely large. The KL-IPM and Beta-IPM formulas above are derived in Appendix A.1. Note that the problem of minimizing the beta divergence is non-convex, so we have made some approximations in order to derive (6).

4.4 Example of KL-IPM and Beta-IPM

Fig. 3 is a two-dimensional example that displays the behaviors of KL-IPM and Beta-IPM. While KL-IPM equally weights each parameter vector, Beta-IPM weights the vector farther from the mean less, and in this way it suppresses the influence of contamination. As a result, the mixed parameter vector chosen by Beta-IPM is closer to the true center (=$(0,0)^\top$) than that by KL-IPM.

5 Mistake / Loss Bound in IPM

This section provides a theoretical viewpoint for the weight determination by Beta-IPM. We first discuss the separable mistake bound of IPM with a single iteration perceptron (IPM-perceptron) in Section 5.1, then goes to the corresponding loss bound of IPM with a single iteration PA (IPM-PA) in Section 5.2. With these bounds, we discuss about Beta-IPM as a suppressor of weights in contaminated shards in Section 5.3.

5.1 Mistake bound of IPM-perceptron

The following theorem, which is proven by McDonald et al. [17], is an extension of the well-known mistake bound of the single machine perceptron to IPM,

Theorem 1. (Mistake bound of IPM-perceptron in the separable case) [Theorem 3 in [17]] *Assume all the training data is separable by a margin γ. Suppose that $||\mathbf{x}|| \leq R$ holds for any training input $\mathbf{x}$, and let $k_{i,n}$ be the number of mistakes in shard i during the nth epoch of training. For any number of epochs N, the number of mistakes during the training in the IPM-perceptron is bounded as*

$$\sum_{n=1}^{N}\sum_{i=1}^{M}\alpha_{i,n}k_{i,n} \leq \frac{R^2}{\gamma^2}. \tag{7}$$

Theorem 1 states that the IPM-perceptron with separable data has a finite number of misses, which guarantees it converges to parameters that correctly classify the entire data.

In contrast, when some fraction of the dataset is non-separable, there are no parameters that perfectly classify all the data. Yet even in this case, we can bound the mistake in terms of the loss of the possible best classifier (parameter vector) $\mathbf{u}$.

Theorem 2. (Mistake bound of IPM-perceptron in the non-separable case) *Let $k_{i,n}$ be the number of mistakes in shard i during the nth epoch of training. Furthermore, let $\mathbf{u}$ be an arbitrary normalized parameter vector $\mathbf{u} \in \mathbb{R}^n (||\mathbf{u}|| = 1)$ Let $\xi = \max\{0, \gamma - y(\mathbf{u} \cdot \mathbf{x})\}$ and $\Xi_i = \sum_{t'} \xi$, where the index t' runs over all data points in shard i. For any number of epochs N and any $\gamma \geq 0$, the following inequality holds:*

$$\sum_{n=1}^{N}\sum_{i=1}^{M}\alpha_{i,n}k_{i,n} \leq \frac{R^2}{\gamma^2} + \frac{2}{\gamma}\sum_{n=1}^{N}\sum_{i=1}^{M}\alpha_{i,n}\Xi_i. \tag{8}$$

Theorem 2 is proven by the combination of the technique for the IPM loss bound [17] and an ordinary technique for the non-separable mistake bound of perceptron. The proof of Theorem 2 is in a full version of this paper. Notice that, ξ is the distance from the margin with a data point $(\mathbf{x}, y)$, which indicates how the classification with a classifier $\mathbf{u}$ fails for this data point. Therefore, Ξ_i, the sum of ξ over shard i, can be considered as a cumulative loss if $\mathbf{u}$ is run on shard i. From inequality (8), the number of mistakes of IPM-perceptron is bounded in terms of the cumulative loss of an arbitrary vector $\mathbf{u}$.

5.2 Loss Bound of IPM-PA

Here, we describe the loss bound of IPM with the Passive Aggressive algorithm (IPM-PA). As in the case of IPM-perceptron, we can obtain separable and non-separable loss bounds. The proofs of the bounds are in Appendix A.2.

Theorem 3. (Loss bound of IPM-PA in the separable case)
Let there be a parameter vector $\mathbf{u}$ *that suffers no loss for any data point* $(\mathbf{x}, y)$ *in the training data set. Suppose that* $||\mathbf{x}|| \leq R$ *holds for any input* $\mathbf{x}$*. Then,*

$$\sum_{n=1}^{N}\sum_{i=1}^{M} \alpha_{i,n} L_{i,n} \leq ||\mathbf{u}||^2 R^2, \tag{9}$$

where $L_{i,n}$ *is the cumulative squared loss which worker* i *suffers in epoch* n*.*

Theorem 4. (Loss bound of IPM-PA in the non-separable case)
Assume $||\mathbf{x}|| = 1$ *holds for any data point. Then, for any parameter vector* $\mathbf{u}$*,*

$$\sum_{n=1}^{N}\sum_{i=1}^{M} \alpha_{i,n} L_{i,n} \leq \left(||\mathbf{u}|| + 2\sqrt{\sum_{n=1}^{N}\sum_{i=1}^{M} \alpha_{i,n} L_i^*} \right)^2 , \tag{10}$$

where L_i^* *is the cumulative squared loss of parameter vector* $\mathbf{u}$ *with data on shard* i*.*

5.3 Superiority of Beta-IPM from a Theoretical Perspective

The cumulative loss in (8) is weighted by $\alpha_{i,n}$. Suppose the shards are divided into two categories: separable shards $i = 1, ..., m$ which can be classified by $\mathbf{u}$ and non-separable shards $i = m+1, ..., M$ with no vector to correctly classify them. The smaller the weights of the non-separable shards $\alpha_{m+1}, ..., \alpha_M$ are, the smaller the weighted cumulative loss $\sum_{i=1}^{M} \alpha_{i,n} \Xi_i$ we can obtain, and this means that it is very important to reduce the weights corresponding to contaminated shards. The same argument goes with PA. In general, Beta-IPM suppresses the weights of non-separable shards as described in Section 4, and thus Beta-IPM is expected to have a smaller mistake count than KL-IPM.

6 Empirical Evaluation

We conducted an evaluation with various datasets. The overall goal of these experiments was to study how KL-IPM and Beta-IPM behave in contaminated environments.

6.1 Setup

Our experiments involved 16 datasets (Table 1). Zeta and ocr datasets are from the Pascal large-scale learning challenge[1], and the imdb and citeseer datasets are from Paul Komarek's webpage[2]. The other datasets are from the LIBSVM dataset repository[3].

[1] `http://largescale.ml.tu-berlin.de/`
[2] `http://komarix.org/ac/ds/`
[3] `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html`

Table 1. List of the binary classification datasets evaluated. The tasks of the datasets are CI (census income prediction), DC (document categorization), HA (human answer prediction), IP (involvement prediction of person to some contents), IR (image recognition), MD (malware / suspicious contents detection), S (synthetically created problem), TC (toxicity categorization), or TD (text decoding).

	# of features	# of data points	task
ijcnn1.tr	22	49,990	TD
mushrooms	112	8,124	TC
a8a	123	22,696	CI
ocr	1,156	3,500,000	IR
epsilon	2,000	400,000	S
zeta	2,000	500,000	S
gisette	5,000	6,000	IR
real-sim	20,958	72,309	DC

	# of features	# of data points	task
rcv1	47,236	20,242	DC
citeseer	105,354	181,395	IP
imdb	685,569	167,773	IP
news20	1,355,191	19,996	DC
url	3,231,961	2,396,130	MD
webspam	16,609,143	350,000	MD
kdda	20,216,830	8,407,752	HA
kddb	29,890,095	19,264,097	HA

Data shards: For each dataset, we used 80% of the data for training and 20% for testing. Then, the training dataset is divided into 100 shards associated with workers. Algorithms are trained with the training data and evaluated in terms of the classification accuracy of the test data.

To study the proposed algorithms' robustness against contamination, we studied the clean setting (i.e. no contamination) and two following contamination settings. Note that the contaminations are only on the training shards (the test data is always clean).
Setting 1 - adversarial labels: In this setting, 30 out of 100 shards are adversarial data: the labels of the data are reversed. This setting models situations where the data in some shards are maliciously labeled.
Setting 2 - random labels: In this setting, 80 out of 100 shards are assigned random labels. Each data point in these randomly labeled shards is labeled $y_t \sim \mathrm{Bernoulli}(p)$, regardless of its true label. The ratio of positive labels, p, varies from 0.1 to 0.9 among workers. This setting models situations where data in most shards are corrupted.
Algorithms: We compared four algorithms: KL-IPM with a single-iteration perceptron or Passive Aggressive (KL-IPM-perceptron and KL-IPM-PA, respectively) and Beta-IPM with a single-iteration perceptron or Passive Aggressive (Beta-IPM-perceptron and Beta-IPM-PA, respectively). All values of β were the best among $\{10^{-1}, 10^{-2}, ..., 10^{-8}\}$. Since our research includes high-dimensional datasets, we assumed that the Gaussian in Beta-IPM was diagonal. The features with zero-variances were ignored in the weight calculation. We normalized the parameter vector of each worker by using the $l2$-norm when calculating the weights in Beta-IPM.

6.2 Results

The results for all datasets are shown in Table 2. The results of KL-IPM in the clean setting can be considered to be the possible best performance of linear classifiers in our distributed setting. As aforementioned, the performance of IPM is degraded by contamination. Note that our main interest in these experiments is the extent to which Beta-IPM can remedy the effects of contamination. First, let us compare the results of KL-IPM in

Table 2. Accuracy of the algorithms in the clean/adversarial/random settings at the 50th epoch. Boldface entries in the contamination settings are the best among the individual datasets.

Clean Setting	ijcnn1.tr	mushrooms	a8a	ocr	epsilon	zeta	gisette	real-sim	rcv1	citeseer	imdb	news20	url	webspam	kdda	kddb
KL-IPM-perceptron	0.913	0.999	0.845	0.763	0.899	0.628	0.947	0.968	0.960	0.976	0.981	0.953	0.986	0.990	0.881	0.886
KL-IPM-PA	0.912	0.999	0.845	0.762	0.899	0.694	0.958	0.975	0.966	0.977	0.985	0.958	0.986	0.990	0.882	0.887

Contamination Setting 1: Adversarial Labels	ijcnn1.tr	mushrooms	a8a	ocr	epsilon	zeta	gisette	real-sim	rcv1	citeseer	imdb	news20	url	webspam	kdda	kddb
KL-IPM-perceptron	**0.908**	0.937	0.838	0.760	0.881	0.582	0.854	0.824	0.762	0.976	0.980	0.703	0.983	0.987	0.743	0.759
KL-IPM-PA	**0.908**	0.983	0.837	0.760	0.886	0.651	0.912	0.904	0.871	**0.977**	**0.984**	0.844	0.983	0.987	0.689	0.712
Beta-IPM-perceptron	**0.908**	**0.998**	**0.846**	**0.763**	**0.898**	**0.665**	0.935	0.961	0.955	0.976	0.981	0.945	**0.986**	**0.991**	**0.876**	**0.882**
Beta-IPM-PA	**0.908**	0.989	**0.846**	0.762	**0.898**	0.663	**0.957**	**0.972**	**0.962**	**0.977**	**0.984**	**0.950**	**0.986**	**0.991**	0.676	0.693

Contamination Setting 2: Random Labels	ijcnn1.tr	mushrooms	a8a	ocr	epsilon	zeta	gisette	real-sim	rcv1	citeseer	imdb	news20	url	webspam	kdda	kddb
KL-IPM-perceptron	0.886	0.858	0.820	0.750	0.737	0.516	0.698	0.680	0.600	0.657	0.611	0.644	0.971	0.951	0.739	0.734
KL-IPM-PA	0.855	0.942	0.817	0.674	0.758	0.545	0.827	0.741	0.701	0.685	0.684	**0.730**	0.971	0.960	0.761	0.757
Beta-IPM-perceptron	0.911	0.980	0.825	**0.755**	0.886	**0.642**	0.888	0.948	**0.919**	0.836	0.826	0.717	0.981	**0.986**	0.853	0.833
Beta-IPM-PA	**0.913**	**0.999**	**0.830**	0.723	**0.890**	0.624	**0.942**	**0.958**	0.910	**0.916**	**0.943**	**0.730**	**0.985**	0.985	**0.868**	**0.868**

the adversarial/random settings with those in the clean setting. The contamination negatively affected the results on almost all datasets. On the random setting, where 80% of the shards are contaminated, the damage to the results tended to be more severe than that on the adversarial setting. Second, let us compare the performances of Beta-IPM and KL-IPM in the adversarial/random settings. One can see that Beta-IPM outperformed KL-IPM on almost all datasets. Indeed, one many datasets Beta-IPM performed almost as well as KL-IPM under the clean setting; this confirms that Beta-IPM can remove the influence of contamination.

Fig. 4 shows the classification results of Beta-IPM with a single iteration perceptron for several values of β. The optimal value with this dataset was $\beta = 10^{-5}$, and the accuracy with this β value showed a steady rise in epochs. The accuracy after epoch 50 was nearly 95%. With a β value smaller than the optimal one ($\beta = 10^{-6}$) and

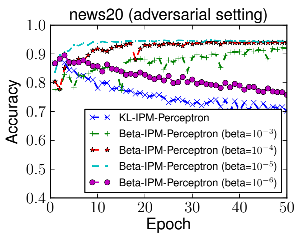

Fig. 4. Classification accuracy of Beta-IPM with a single iteration perceptron for several values of beta. The algorithms were run with news20 in the adversarial setting.

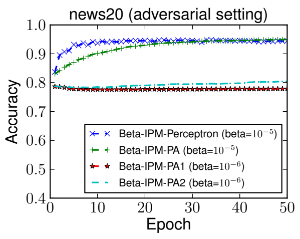

Fig. 5. Classification accuracies of Beta-IPM with various algorithms. The algorithms were run with news20 in the adversarial setting.

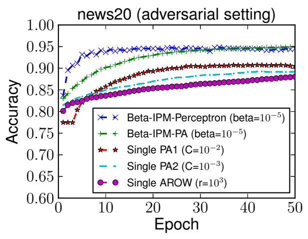

Fig. 6. Classification accuracy of Beta-IPM and single machine PA-I/PA-II/AROW. The algorithms were run with news20 in the adversarial setting.

with no beta (KL-IPM-perceptron), the algorithm failed to suppress the influence of the adversarial workers. Conversely, with β values bigger than optimal ($\beta = 10^{-3}$), the regularization was so strong that even the influence of some of the correct workers was suppressed. As a result, the learning rate with this beta value was very slow.

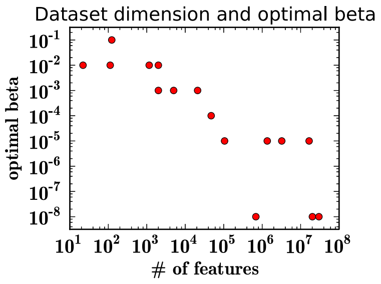

Fig. 7. Number of features and optimal value of beta in the adversarial setting. Each point corresponds to a dataset.

Fig. 5 compares several different algorithms with the best beta values. Given a proper value of β, Beta-IPM with a perceptron or PA successfully learned the parameter vectors. However, PA-I and PA-II[4], the noise-tolerant version of PA, did not perform well. These results indicate that robustness in a distributed environment is essentially different from that of single machine online learning: while we assume some fraction of the data is clean and the rest is contaminated, robust learning in a single machine aims to learn in environments where the clean and contaminated data are mixed. A possible hypothesis is that, the regularization of the learning rate in PA-I and PA-II obscured the difference between clean and contaminated shards, which made the accuracies of IPM with PA-I and PA-II poor.

Fig. 6 compares Beta-IPM-perceptron with a single machine PA-I or PA-II and AROW [6]. The hyper-parameter C in PA-I and PA-II and r in AROW were optimized in $\{10^{-4}, 10^{-3}, ..., 10^{4}\}$. The data of all 100 shards were put into a single shard in the single machine experiments. The two Beta-IPM algorithms performed better than the single machine algorithms. These results are empirical evidence that Beta-IPM can reduce the weights of adversarial shards.

Fig. 7 shows the optimal value of beta as a function of the number of features. Overall, in high-dimensional datasets, the value of beta tends to be small. The reason for

[4] The parameter C in PA-I and PA-II was set to be 0.001.

this is that the weight in Beta-IPM (Equation (6)) is a multivariate Gaussian, which is a product of exponentials over all dimensions and thus is small at high dimensions.

7 Conclusion

We studied robust distributed training of linear classifiers. By minimizing the divergence, we devised a criterion for determining the weights in IPM. Experiments revealed that the performance of IPM is significantly recovered on many contaminated datasets by determining the weights based on the beta divergence. An interesting direction of future work is to remove the statistiscal assumption of Gaussian distribution, by allowing more wider class of distributions, or non-parametric models.

References

1. Aberdeen, D., Pacovsky, O., Slater, A.: The learning behind gmail priority inbox. In: LCCC: NIPS 2010 Workshop on Learning on Cores, Clusters and Clouds (2010)
2. Basu, A., Harris, I.R., Hjort, N.L., Jones, M.C.: Robust and efficient estimation by minimising a density power divergence. Biometrika 85(3), 549–559 (1998)
3. Chouvardas, S., Slavakis, K., Theodoridis, S.: Adaptive robust distributed learning in diffusion sensor networks. IEEE Transactions on Signal Processing 59(10), 4692–4707 (2011)
4. Chu, C.T., Kim, S.K., Lin, Y.A., Yu, Y., Bradski, G.R., Ng, A.Y., Olukotun, K.: Map-reduce for machine learning on multicore. In: NIPS, pp. 281–288 (2006)
5. Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., Singer, Y.: Online passive-aggressive algorithms. Journal of Machine Learning Research 7, 551–585 (2006)
6. Crammer, K., Kulesza, A., Dredze, M.: Adaptive regularization of weight vectors. Machine Learning 91(2), 155–187 (2013)
7. Curtsinger, C., Livshits, B., Zorn, B.G., Seifert, C.: Zozzle: Fast and precise in-browser javascript malware detection. In: USENIX Security Symposium (2011)
8. Daumé III, H., Phillips, J.M., Saha, A., Venkatasubramanian, S.: Efficient protocols for distributed classification and optimization. In: Bshouty, N.H., Stoltz, G., Vayatis, N., Zeugmann, T. (eds.) ALT 2012. LNCS, vol. 7568, pp. 154–168. Springer, Heidelberg (2012)
9. Dekel, O., Shamir, O., Xiao, L.: Learning to classify with missing and corrupted features. Mach. Learn. 81(2), 149–178 (2010)
10. Djuric, N., Grbovic, M., Vucetic, S.: Distributed confidence-weighted classification on mapreduce. In: IEEE Bigdata (2013)
11. Eguchi, S., Kano, Y.: Robustifying maximum likelihood estimation. Technical report, Institute of Statistical Mathematics (June 2001)
12. Gimpel, K., Das, D., Smith, N.A.: Distributed asynchronous online learning for natural language processing. In: CoNLL 2010, pp. 213–222 (2010)
13. Gong, P., Ye, J., Zhang, C.: Robust multi-task feature learning. In: KDD, pp. 895–903 (2012)
14. Hall, K.B., Inc, G., Gilpin, S., Mann, G.: Mapreduce/bigtable for distributed optimization. In: LCCC: NIPS 2010 Workshop on Learning on Cores, Clusters and Clouds (2010)
15. Hoi, S.C.H., Wang, J., Zhao, P.: Exact soft confidence-weighted learning. In: ICML (2012)
16. Mann, G., McDonald, R.T., Mohri, M., Silberman, N., Walker, D.: Efficient large-scale distributed training of conditional maximum entropy models. In: NIPS, pp. 1231–1239 (2009)
17. McDonald, R., Hall, K., Mann, G.: Distributed training strategies for the structured perceptron. In: NAACL, HLT 2010, pp. 456–464 (2010)

18. Meyer, T.A., Whateley, B.: Spambayes: Effective open-source, bayesian based, email classification system. In: CEAS (2004)
19. Rahm, E., Do, H.H.: Data cleaning: Problems and current approaches. IEEE Data Engineering Bulletin 23, 2000 (2000)
20. Rosenblatt, F.: The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review 65(6), 386–408 (1958)
21. Runnalls, A.R.: A kullback-leibler approach to gaussian mixture reduction. IEEE Trans. Aerosp. Electron. Syst., 989–999 (2007)
22. Tsitsiklis, J., Bertsekas, D., Athans, M.: Distributed asynchronous deterministic and stochastic gradient optimization algorithms. IEEE Transactions on Automatic Control 31(9), 803–812 (1986)
23. Xu, H., Leng, C.: Robust multi-task regression with grossly corrupted observations. Journal of Machine Learning Research - Proceedings Track 22, 1341–1349 (2012)
24. Zinkevich, M., Smola, A.J., Langford, J.: Slow learners are fast. In: NIPS, pp. 2331–2339 (2009)

A Appendix

A.1 Derivation of KL-IPM and Beta-IPM

Derivation of KL-IPM. We want to show that a mixed weight based on KL-IPM minimizes the KL-divergence between P and Q based on the parameter vectors $\{\mathbf{w}^{(1,n)}, ..., \mathbf{w}^{(M,n)}\}$. The following lemma states that KL divergence minimization based on Gaussian distributions preserves the mean and covariance.

Lemma 5. [Theorem 3.2 in [21]] *Let P be an arbitrary probability distribution on $\mathbb{R}^d$ with a well-defined mean $\boldsymbol{\mu}^*$ and covariance matrix $\boldsymbol{\Sigma}^*$, where $\boldsymbol{\Sigma}^*$ is strictly positive-definite. Let Q be a Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. The unique minimum value of $D_{KL}(P||Q)$ is achieved when $\boldsymbol{\mu} = \boldsymbol{\mu}^*$ and $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}^*$.*

The inequality (4) follows by using Lemma 5 and the fact that the empirical mean of P on the parameter vectors is $(1/M)\sum_i \mathbf{w}^{(i,n)}$.

Derivation of Beta-IPM. Let the parameters of the workers be $\{\mathbf{w}^{(1,n)}, ..., \mathbf{w}^{(M,n)}\}$, which is generated from a distribution P, and $Q(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ be a Gaussian distribution. We would like to minimize the beta divergence, namely, $\mathbf{w}^{(\text{avg},n)} = \arg\min_{\boldsymbol{\mu}} D_\beta(P||Q(\boldsymbol{\mu}, \boldsymbol{\Sigma}))$. Then,

$$\begin{aligned}
&D_\beta(P||Q(\boldsymbol{\mu}, \boldsymbol{\Sigma})) && (11)\\
=\ & -\frac{1}{\beta}\int P(\mathbf{w})Q^\beta(\mathbf{w}|\boldsymbol{\mu}, \boldsymbol{\Sigma})d\mathbf{w} + \frac{1}{\beta+1}\int Q^{\beta+1}(\mathbf{w}|\boldsymbol{\mu}, \boldsymbol{\Sigma})d\mathbf{w} + \text{Const.} && (12)\\
=\ & -\frac{1}{\beta}E_{P(\mathbf{w})}[Q^\beta(\mathbf{w}|\boldsymbol{\mu}, \boldsymbol{\Sigma})] + \frac{1}{\beta+1}\int Q^{\beta+1}(\mathbf{w}|\boldsymbol{\mu}, \boldsymbol{\Sigma})d\mathbf{w} + \text{Const.}, && (13)
\end{aligned}$$

where Const. is a term independent of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, and $E_{P(\mathbf{w})}$ is the expectation under the assumption that $\mathbf{w}$ follows the probability distribution $P(\mathbf{w})$. Replacing the expectation of the first term with an empirical expectation over the parameter vectors $\{\mathbf{w}^{(1,n)}, ..., \mathbf{w}^{(M,n)}\}$ yields

$$(13) = -\frac{1}{\beta}\sum_{i=1}^{M}\frac{1}{M}\left[Q^{\beta}(\mathbf{w}^{(i,n)}|\boldsymbol{\mu},\boldsymbol{\Sigma})\right] + \frac{1}{\beta+1}\int Q^{\beta+1}(\mathbf{w}|\boldsymbol{\mu},\boldsymbol{\Sigma})d\mathbf{w}. \tag{14}$$

The multivariate Gaussian Q is explicitly written as $Q(\mathbf{w}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = Z(\boldsymbol{\Sigma}) \exp S(\mathbf{w}|\boldsymbol{\mu},\boldsymbol{\Sigma})$, where $Z(\boldsymbol{\Sigma}) = 1/\sqrt{(2\pi)^d|\boldsymbol{\Sigma}|}$ and $S(\mathbf{w}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = -\frac{1}{2}(\mathbf{w}-\boldsymbol{\mu})^\top\boldsymbol{\Sigma}^{-1}(\mathbf{w}-\boldsymbol{\mu})$. The second term in (14) is, from the property of multivariate Gaussian distribution,

$$\frac{1}{\beta+1}\int Q^{\beta+1}(\mathbf{w}|\boldsymbol{\mu},\boldsymbol{\Sigma})d\mathbf{w} = Z(\boldsymbol{\Sigma})^{\beta}(\beta+1)^{-1-d/2}, \tag{15}$$

which is independent on $\boldsymbol{\mu}$. By using these facts, the first derivative of $D_\beta(P||Q(\boldsymbol{\mu},\boldsymbol{\Sigma}))$ is equivalent to the one of the first term in the RHS of (14), which is transformed as,

$$\begin{aligned}\frac{d}{d\boldsymbol{\mu}}\left\{-\frac{1}{\beta}\sum_{i=1}^{M}\frac{1}{M}\left[Q^{\beta}(\mathbf{w}^{(i,n)}|\boldsymbol{\mu},\boldsymbol{\Sigma})\right]\right\} &= \frac{Z(\boldsymbol{\Sigma})^{\beta}}{\beta M}\left\{-\frac{d}{d\boldsymbol{\mu}}\sum_{i=1}^{M}\exp S(\mathbf{w}^{(i,n)},\boldsymbol{\mu},\frac{1}{\beta}\boldsymbol{\Sigma})\right\}.\\ &= \frac{Z(\boldsymbol{\Sigma})^{\beta}}{\beta M}\left\{\sum_{i=1}^{M}\exp S(\mathbf{w}^{(i,n)},\boldsymbol{\mu},\frac{1}{\beta}\boldsymbol{\Sigma})(\beta\boldsymbol{\Sigma}^{-1})(\mathbf{w}^{(i,n)}-\boldsymbol{\mu})\right\}\\ &= \frac{Z(\boldsymbol{\Sigma})^{\beta}}{\beta M}\left\{\beta\boldsymbol{\Sigma}^{-1}\sum_{i=1}^{M}\exp S(\mathbf{w}^{(i,n)},\boldsymbol{\mu},\frac{1}{\beta}\boldsymbol{\Sigma})(\mathbf{w}^{(i,n)}-\boldsymbol{\mu})\right\}.\end{aligned} \tag{16}$$

Therefore, we obtain

$$\begin{aligned}\frac{d}{d\boldsymbol{\mu}}\left\{-\frac{1}{\beta}\sum_{i=1}^{M}\frac{1}{M}\left[Q^{\beta}(\mathbf{w}^{(i,n)}|\boldsymbol{\mu},\boldsymbol{\Sigma})\right]\right\} = 0 \;&\Leftrightarrow \sum_{i=1}^{M}\exp S(\mathbf{w}^{(i,n)},\boldsymbol{\mu},\frac{1}{\beta}\boldsymbol{\Sigma})(\mathbf{w}^{(i,n)}-\boldsymbol{\mu}) = 0\\ &\Leftrightarrow \boldsymbol{\mu} = \frac{\sum_{i=1}^{M}\exp S(\mathbf{w}^{(i,n)},\boldsymbol{\mu},\frac{1}{\beta}\boldsymbol{\Sigma})\mathbf{w}^{(i,n)}}{\sum_{j=1}^{M}\exp S(\mathbf{w}^{(j,n)},\boldsymbol{\mu},\frac{1}{\beta}\boldsymbol{\Sigma})}.\end{aligned} \tag{17}$$

RHS of (17) states that $\boldsymbol{\mu}$ that minimizes $D_\beta(P||Q)$ is a weighted mean of each $\mathbf{w}^{(i,n)}$ with weight $\exp S(\mathbf{w}^{(i,n)},\boldsymbol{\mu},\frac{1}{\beta}\boldsymbol{\Sigma})$. Unfortunately, the weight $\exp S(\mathbf{w}^{(i,n)},\boldsymbol{\mu},\frac{1}{\beta}\boldsymbol{\Sigma})$ on the RHS includes $\boldsymbol{\mu}$, and thus, an exact solution is unattainable. To get a reasonable solution, we can approximate $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ on the RHS of (17) by the mean and covariance of the samples $\{\mathbf{w}^{(1,n)}, ..., \mathbf{w}^{(M,n)}\}$, which finally yields (6).

A.2 Proof of Theorem 3 and 4

The crux in the mistake/loss bound proofs in online classifiers is to find some value that can be bounded from both the upper and lower side: in the case of PA we bound the value $\Delta_n = ||\mathbf{w}^{(\text{avg},n-1)} - \mathbf{u}|| - ||\mathbf{w}^{(\text{avg},n)} - \mathbf{u}||$. By using these lower and upper bounds we obtain Lemma 6, which leads to the proofs of Theorem 3 and 4.

Algorithm 2. Single iteration Passive Aggressive

```
1: $T = \{(\mathbf{x}_t, y_t)\}, \mathbf{w}$
2: for $t = 1, ..., |T|$ do
3:     $\hat{y}_t \leftarrow \text{sign}(\mathbf{w} \cdot \mathbf{x}_t)$
4:     $l_t \leftarrow \max(0, 1 - y_t(\mathbf{w} \cdot \mathbf{x}_t))$
5:     $\tau_t \leftarrow l_t / ||\mathbf{x}_t||^2$
6:     $\mathbf{w} \leftarrow \mathbf{w} + \tau_t y_t \mathbf{x}_t$
7: end for
```

Lemma 6. *Let the index $t = 1, ..., k_{i,n}$ denotes the data points on shard i that the worker suffered non-zero losses, and $(\mathbf{x}_{i,t}, y_{i,t})$ be the data point at that round. Moreover, let $l_{i,t}$ be the corresponding loss of the worker, and $\tau_{i,t} = l_{i,t}/||\mathbf{x}_{i,t}||^2$, and $l^*_{i,t}$ be the loss of any constant classifier $\mathbf{u}$ with the data point. Then,*

$$\sum_{n=1}^{N} \left\{ \sum_{i=1}^{M} \alpha_{i,n} \sum_{t=1}^{k_{i,n}} \left\{ \tau_{i,t}(2l_{i,t} - \tau_{i,t}||\mathbf{x}_{i,t}||^2 - 2l^*_{i,t}) \right\} \right\} \leq ||\mathbf{u}||. \tag{18}$$

Proof (Lemma 6). Consider shard i in epoch n. Let $\Delta_{i,n} = ||\mathbf{w}^{(\text{avg},n-1)} - \mathbf{u}||^2 - ||\mathbf{w}^{(i,n)} - \mathbf{u}||^2$. Notice that the parameter vector is updated only when the loss is non-zero, and For $1 \leq t \leq k_{i,n}$, let $\mathbf{w}^{([i,n]+t)}$ be the parameter vector on shard i in epoch n in the round just before the t-th loss occurred. Also, for $t = k_{i,n} + 1$, let $\mathbf{w}^{([i,n]+t)} = \mathbf{w}^{(i,n)}$. Notice that $\mathbf{w}^{([i,n]+1)} = \mathbf{w}^{(\text{avg},n-1)}$. The update per single loss is,

$$\begin{aligned}
&||\mathbf{w}^{([i,n]+t)} - \mathbf{u}||^2 - ||\mathbf{w}^{([i,n]+(t+1))} - \mathbf{u}||^2 \\
&= ||\mathbf{w}^{([i,n]+t)} - \mathbf{u}||^2 - ||\mathbf{w}^{([i,n]+t)} - \mathbf{u} + y_{i,t}\tau_{i,t}\mathbf{x}_{i,t}||^2 \\
&= ||\mathbf{w}^{([i,n]+t))} - \mathbf{u}||^2 - \left\{ ||\mathbf{w}^{([i,n]+t)} - \mathbf{u}||^2 + 2y_{i,t}\tau_{i,t}(\mathbf{w}^{([i,n]+t)} - \mathbf{u}) \cdot \mathbf{x}_{i,t} + \tau^2_{i,t}||\mathbf{x}_{i,t}||^2 \right\} \\
&= -2y_{i,t}\tau_{i,t}(\mathbf{w}^{([i,n]+t)} - \mathbf{u}) \cdot \mathbf{x}_{i,t} - \tau^2_{i,t}||\mathbf{x}_{i,t}||^2.
\end{aligned} \tag{19}$$

Since we assumed $l_{i,t} > 0$ with this data point, $y_{i,t}(\mathbf{w}^{([i,n]+t)} \cdot \mathbf{x}_{i,t}) = 1 - l_{i,t}$ and $l^*_{i,t} \geq 1 - y_{i,t}(\mathbf{u} \cdot \mathbf{x}_{i,t})$ always holds. Thus, (19) can be bounded as,

$$(19) \geq 2\tau_{i,t}((1 - l^*_{i,t}) - (1 - l_{i,t})) - \tau^2_{i,t}||\mathbf{x}_{i,t}||^2 = \tau_{i,t}(2l_{i,t} - \tau_{i,t}||\mathbf{x}_{i,t}||^2 - 2l^*_{i,t}). \tag{20}$$

By using (20), $\Delta_{i,n}$ is bounded as,

$$\begin{aligned}
\Delta_{i,n} &= ||\mathbf{w}^{(\text{avg},n-1)} - \mathbf{u}||^2 - ||\mathbf{w}^{(i,n)} - \mathbf{u}||^2 \\
&= \sum_{t=1}^{k_{i,n}} \left(||\mathbf{w}^{([i,n]+t)} - \mathbf{u}||^2 - ||\mathbf{w}^{([i,n]+(t+1))} - \mathbf{u}||^2 \right) \\
&\geq \sum_{t=1}^{k_{i,n}} \left\{ \tau_{i,t}(2l_{i,t} - \tau_{i,t}||\mathbf{x}_{i,t}||^2 - 2l^*_{i,t}) \right\}.
\end{aligned} \tag{21}$$

We now lower-bound $\varDelta_n$ as follows:

$$\begin{aligned}\varDelta_n &= ||\mathbf{w}^{(\mathrm{avg},n-1)} - \mathbf{u}||^2 - ||\mathbf{w}^{(\mathrm{avg},n)} - \mathbf{u}||^2 = ||\mathbf{w}^{(\mathrm{avg},n-1)} - \mathbf{u}||^2 - ||\sum_i \alpha_{i,n}(\mathbf{w}^{(i,n)} - \mathbf{u})||^2 \\ &\geq \sum_{i=1}^{M} \alpha_{i,n} \left(||\mathbf{w}^{(\mathrm{avg},n-1)} - \mathbf{u}||^2 - ||\mathbf{w}^{(i,n)} - \mathbf{u}||^2\right) = \sum_{i=1}^{M} \alpha_{i,n} \varDelta_{i,n} \\ &\geq \sum_{i=1}^{M} \alpha_{i,n} \sum_{t=1}^{k_{i,n}} \left\{\tau_{i,t}(2l_{i,t} - \tau_{i,t}||\mathbf{x}_{i,t}||^2 - 2l^*_{i,t})\right\}, \qquad (22)\end{aligned}$$

where we have used $\sum_i \alpha_{i,n} = 1$ in going between the first and second line, and used (21) at the last transformation.

On the other hand, the sum of $\varDelta_n$ is upper-bounded as follows:

$$\begin{aligned}\sum_{n=1}^{N} \varDelta_n = \sum_{n=1}^{N} &\left(||\mathbf{w}^{(\mathrm{avg},n-1)} - \mathbf{u}||^2 - ||\mathbf{w}^{(\mathrm{avg},n)} - \mathbf{u}||^2\right) \\ &= ||\mathbf{w}^{(\mathrm{avg},0)} - \mathbf{u}||^2 - ||\mathbf{w}^{(\mathrm{avg},N)} - \mathbf{u}||^2 \leq ||\mathbf{u}||, \qquad (23)\end{aligned}$$

where the last inequality follows from the fact that the initial parameter vector is the zero vector and $||\mathbf{w}^{(\mathrm{avg},N)} - \mathbf{u}||^2 \geq 0$. Using (22) and (23) yields (18).

□

Proof (Theorem 3). By using the fact that $l^*_{i,t} = 0$, $l_{i,t} = \tau_{i,t}||\mathbf{x}_{i,t}||^2$, Lemma 6 is transformed as follows:

$$\sum_{n=1}^{N} \left\{ \sum_{i=1}^{M} \alpha_{i,n} \sum_{t=1}^{k_{i,n}} \frac{(l_{i,t})^2}{||\mathbf{x}_{i,t}||^2} \right\} \leq ||\mathbf{u}||^2. \qquad (24)$$

With the fact that $||\mathbf{x}_{i,t}|| < R$, we finally obtain

$$\sum_{n=1}^{N} \left\{ \sum_{i=1}^{M} \alpha_{i,n} \sum_{t=1}^{k_{i,n}} (l_{i,t})^2 \right\} \leq ||\mathbf{u}||^2 R^2. \qquad (25)$$

$L_{i,n}$, the cumulative squared loss the worker i suffers during epoch n, corresponds to $\sum_{t=1}^{k_{i,n}} (l_{i,t})^2$. Therefore, the inequality (25) is equivalent to (9).

□

Proof (Theorem 4). Next, we consider the case where $l^*_{i,t}$ is not necessarily zero. Let us assume $||\mathbf{x}_{i,t}|| = 1$. Notice that $\tau_{i,t} = l_{i,t}/||\mathbf{x}_{i,t}||^2 = l_{i,t}$. By these facts and Lemma 6,

$$\sum_{n=1}^{N} \left\{ \sum_{i=1}^{M} \alpha_{i,n} \sum_{t=1}^{k_{i,n}} \left\{(l_{i,t})^2 - 2l_{i,t}l^*_{i,t})\right\} \right\} \leq ||\mathbf{u}||^2. \qquad (26)$$

Let

$$X_N = \sqrt{\sum_{n=1}^{N} \sum_{i=1}^{M} \alpha_{i,n} \sum_{t=1}^{k_{i,n}} (l_{i,t})^2}, \text{ and } Y_N = \sqrt{\sum_{n=1}^{N} \sum_{i=1}^{M} \alpha_{i,n} \sum_{t=1}^{k_{i,n}} (l^*_{i,t})^2}. \qquad (27)$$

Using the Cauchy-Schwarz inequality on the LHS of (26), we obtain $X_N^2 - 2X_N Y_N \leq ||\mathbf{u}||^2$, which is a quadratic inequality of X_N, and thus $X_N \leq Y_N + \sqrt{Y_N^2 + ||\mathbf{u}||^2} \leq ||\mathbf{u}|| + 2Y_N$, where we used the fact that $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$ for $a, b \geq 0$. By explicitly writing X_N and Y_N we obtain

$$\sqrt{\sum_{n=1}^{N}\sum_{i=1}^{M}\alpha_{i,n}\sum_{t=1}^{k_{i,n}}(l_{i,t})^2} \leq ||\mathbf{u}|| + 2\sqrt{\sum_{n=1}^{N}\sum_{i=1}^{M}\alpha_{i,n}\sum_{t=1}^{k_{i,n}}(l_{i,t}^*)^2}. \quad (28)$$

The cumulative squared loss the worker i suffers in epoch n is $L_{i,n} = \sum_{t=1}^{k_{i,n}}(l_{i,t})^2$. Moreover, $L_i^* \geq \sum_{t=1}^{k_{i,n}}(l_{i,t}^*)^2$ holds because the index t runs the subset of data on shard i. Taking these into consideration, we finally obtain (10).

□