Attribute-Distributed Learning: Models, Limits, and Algorithms

Haipeng Zheng, Sanjeev R. Kulkarni, *Fellow, IEEE*, and H. Vincent Poor, *Fellow, IEEE*

***Abstract*—This paper introduces a framework for distributed learning (regression) on attribute-distributed data. First, the convergence properties of attribute-distributed regression with an additive model and a fusion center are discussed, and the convergence rate and uniqueness of the limit are shown for some special cases. Then, taking residual refitting (or L_2 boosting) as a prototype algorithm, three different schemes, Simple Iterative Projection, a greedy algorithm, and a parallel algorithm (with its derivatives), are proposed and compared. Among these algorithms, the first two are sequential and have low communication overhead, but are susceptible to overtraining. The parallel algorithm has the best performance, but has significant communication requirements. Instead of directly refitting the ensemble residual sequentially, the parallel algorithm redistributes the residual to each agent in proportion to the coefficients of the optimal linear combination of the current individual estimators. Designing residual redistribution schemes also improves the ability to eliminate irrelevant attributes. The performance of the algorithms is compared via extensive simulations. Communication issues are also considered: the amount of data to be exchanged among the three algorithms is compared, and the three methods are generalized to scenarios without a fusion center.**

***Index Terms*—Distributed information systems, distributed processing, statistical learning.**

I. Introduction

DISTRIBUTED learning is a field that generalizes classical machine learning algorithms to a distributed framework. Unlike the classical learning framework, in which one has full access to the entire dataset and has unlimited central computational capability, in the framework of distributed learning, the data are distributed among a number of agents. These agents are capable of exchanging certain types of information, which, due to limited computational power and communication restrictions (limited bandwidth, limited power or confidentiality), is usually restricted in terms of content and amount. Research in distributed learning seeks effective learning algorithms and theoretical limits within such constraints on computation, communication, and confidentiality.

Manuscript received April 09, 2010; accepted September 22, 2010. Date of publication October 18, 2010; date of current version December 17, 2010. This research was supported in part by the U.S. Office of Naval Research under Grant N00014-07-1-0555 and N00014-09-1-0342, the U.S. Army Research Office under Grant W911NF-07-1-0185, and the National Science Foundation under Grant CNS-09-05398 and under Science & Technology Center Grant CCF-0939370. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Mathini Sellathurai.

The authors are with the Department of Electrical Engineering, Princeton University, Princeton, NJ 08544 USA (e-mail: haipengz@princeton.edu; kulkarni@princeton.edu; poor@princeton.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSP.2010.2088393

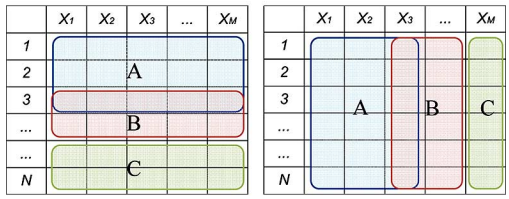


Fig. 1. Two basic scenarios for distributed data: instance-distributed (left) and attribute-distributed (right). In the instance-distributed scenario, each agent (A, B, and C) observes a subset of the instances, with complete information on all attributes; alternatively, in the attribute-distributed scenario, each agent observes all the instances, with a subset of attributes.

In a typical setting of a distributed learning system, there are a number of agents, which are capable of collecting, processing (local training), and communicating a certain amount of data to one another, or to a fusion center. A fusion center may not be required if the links of the agents form a connected component of a graph.

One key issue in distributed learning is the way in which data are distributed among the agents. If each agent observes the entire attribute space with a subset of the instances of the entire data set, then we call the data instance-distributed (or homogeneous data/horizontally partitioned data). On the other hand, if each agent observes all the data instances within a subset of the attribute space, then we call the data attribute-distributed (or heterogeneous data/vertically partitioned data). Of course, there are other hybrid ways to distribute data, but instance- and attribute-distributed data are the two most fundamental cases. Fig. 1 illustrates these two ways to distribute data.

Problems involving instance-distributed data have been widely studied. Two important types of models are established in [17] and [16], respectively: instance-distributed learning with and without a fusion center. The relationship between the information transmitted among individual agents and the fusion center, and the ensemble learning capability, are discussed in these papers.

One of the essential questions about distributed learning is what type of information should be exchanged among the agents. For instance-distributed cases, exchanging estimator information among the agents is a direct, effective and natural choice because the estimator of one agent has exactly the same form (takes input on the same domain) as those of other agents, and hence can be directly applied and evaluated on the data of other agents. Moreover, since estimators to a great extent characterize the statistical properties of the training data, sharing individual estimators can be an efficient way of exchanging information. This is why classical learning algorithms are more easily adapted to the homogeneous data cases with the form

of the classifier/estimator being exactly the same as that of the centralized learning algorithm. Homogeneity in the individual classifiers/estimators is a great advantage for designing distributed learning algorithms that compare and combine them. However, these advantages disappear in the attribute-distributed scenario, in which different agents observe different attributes, and thus have many different forms of classifiers/estimators. This makes it harder to evaluate, compare and combine the estimators.

In this paper, we concentrate on solving distributed learning problems with attribute-distributed data. In particular, we address several fundamental questions:

1) Given the constraints on the observations of each agent, what is the optimal ensemble estimator?
2) Is there an efficient protocol for collaborative training so that the agents can reach the optimal choice of this ensemble estimator?
3) What is the tradeoff between performance and the amount of data exchanged?

In this paper, we will answer these questions in part. In Section II, we briefly review the existing methods in this field, highlighting the residual refitting scheme in detail. In Section III, we pose a fundamental optimization problem: under the residual refitting scheme, find the best thing that we can do, given that we have an infinite amount of data and no noise. Then we propose a solution to this problem. In Section IV, we seek algorithms that perform well with finite, noisy observation data, and we present simulation results to support our conclusions in Section V. In Section VI, we discuss communication issues associated with the distributed learning algorithms, and then conclude our paper in Section VII.

II. Review of Existing Methods

Despite the difficulties of attribute-distributed learning, there are many research results in this area. Basak [1] sets up a general paradigm for classification problem on vertically partitioned data (i.e., attribute-distributed data). There are many algorithms that are distributed versions of different centralized classification algorithms on attribute-distributed data, for instance, k-nearest neighbor [5], support vector machine [13], [23], Bayesian networks [24], [25] and decision trees [6], [22]. In addition, researchers have also generalized unsupervised learning algorithms to attribute-distributed data. For instance, [21] considers k-means clustering problems with attribute-distributed data and [10] generalizes it to arbitrarily partitioned data.

In most of these approaches, the privacy-preservation aspect of distributed algorithms is also highlighted. In other words, in all the algorithms, the protocols under which the agents share information with one another do not involve direct communication of private data.

On the other hand, there are also many works that consider regression problems and emphasize the estimation error of the ensemble estimator, e.g., [14] or [26]. Some basic ideas include voting/averaging, meta-learning, collective data mining, and residual refitting. In terms of the training process, the first two are non-collaborative, i.e., each agent individually trains its own classifier/estimator and fixes it before sending it to others or to a fusion center. The last two are collaborative in the sense that each agent updates its local estimator based on the information/feedback from other agents or a fusion center. Obviously, collaborative training can achieve more accurate estimators, at the price of a greater communication burden of the system. Here, we briefly review the basic idea of these methods. The difference between non-collaborative training and collaborative training is illustrated in Fig. 2.

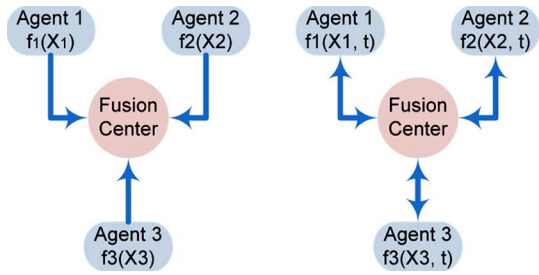


Fig. 2. Comparison between non-collaborative training (left) and collaborative training (right). For non-collaborative training, individual estimators (f_1, f_2 and f_3) are trained locally and fixed, and no agents receives information from other agents. Alternatively, for collaborative training, every agent can get feedback from the fusion center/other agents, and the individual estimators are updated based on external information, evolving as the training algorithm proceeds.

A. Non-Collaborative Training

The voting/averaging algorithm simply combines the predictions of the individual agents, with or without weighting coefficients. The training process is purely non-cooperative and hence requires no communication at all. Usually a fusion center is required for this scenario, and each agent simply sends its prediction on the test set to the fusion center to contribute to the ensemble decision. An extensive simulation of this algorithm is described in [14], in which decision trees are used as individual classifier/estimators, and unweighted/weighted voting schemes are used as combining rules.

In the meta-learning case (see [3], [4], and [15]), the fusion center seeks a more sophisticated way to integrate predictions of individual estimators by taking their predictions as a new training set, i.e., the fusion center treats the output of individual estimators as the input covariates. Although this hierarchical training scheme is somewhat more sophisticated than simple voting/averaging, it is still non-cooperative and hence fails to learn hidden rules in which covariates of different agents are related in a complicated way.

B. Collaborative Training

In contrast to the above methods, collective data mining algorithms (see [9], [11], and [12]) are collaborative. They seek to determine the information required to be shared among the agents (usually certain attributes, or columns of a data matrix $\mathbf{X}$) so that the optimal estimator can be decomposed into an additive form without compromising the performance of the ensemble estimator (compared to an estimator that could be trained by a centralized algorithm). However, this requirement is rather strong and hence this technique relies on specific types of transformations (e.g., wavelets), which require significant prior knowledge of the problem, and thus is hard to generalize to solve other problems.

Another class of cooperative training algorithms, the residual refitting algorithm and algorithms derived from it [19], [26]–[28], has the advantage of not being dependent on individual learning algorithms. The only information the agents communicate with each other is their training residuals. Sharing training residuals is a promising choice for attribute-distributed learning because the training residual of one agent represents the "unexplainable" part of the outcome based on that agent's attributes. Moreover, since residual refitting algorithms bear a natural resemblance to the L_2-boosting algorithm of classical machine learning, many of the conclusions and methodologies of boosting can be borrowed and revised. Nonetheless, there are still many unique problems associated with distributed residual refitting.

An interesting question is the following: If we adopt the residual refitting scheme and assume that each individual agent is able to find its conditional expectation estimator (optimal in terms of mean square error), then what is the limit of the residual refitting algorithm? Reference [26] partly answers this question by demonstrating that repeatedly finding the conditional-mean estimator of the current ensemble residual (iterative projection) is a non-expansive map, and under certain additional assumptions, is a contractive map, and hence converges to a unique limit, which is the optimal estimator of the form of a linear combination of individual estimators. The analysis in [26] justifies the efficacy of the residual refitting algorithm in the infinite instance, noise-free data scenario.

However, since training data is generally limited an noisy, direct residual refitting on ensemble training error in a round robin manner fails to perform well, especially in the presence of irrelevant attributes (attributes that are completely unrelated to the outcome). Usually, this phenomenon is considered to be a form of overtraining in the parlance of machine learning. [19], [27] and [28] have proposed different ways of solving this problem. In [27], uncertainty about the covariance of individual training residuals is introduced so that the ensemble estimator can be searched in a minimax manner to avoid overtraining. In [19] and [28], different ways of agent selection/pruning schemes are proposed to eliminate irrelevant agents so that the final ensemble estimator has a sparser form, leading to better generalization capability. Moreover, selecting/pruning agents can also reduce the amount of data exchange to some extent, making the algorithms more efficient in terms of communication requirements.

In this paper, we consider an alternative residual-based algorithm that addresses this problem in a different manner. Instead of projecting the ensemble residual sequentially on the agents (selected or unselected), which is greedy and myopic, we use the training residual (or equivalently, the prediction on training data) in a holistic way, reshaping the entire prediction matrix using gradient descent. This essentially reduces to a problem of parallel residual refitting in which current ensemble training residuals are distributed in proportion to the weighting coefficients of the best linear combination of individual estimators. In addition, if we do not strictly follow gradient descent, and instead adjust the weighting coefficients to emphasize more promising agents and de-emphasize the less promising ones, a better ensemble estimator can be obtained, with most irrelevant agents being eliminated, and only a few of the most important agents surviving.

III. Limits of Additive Models

A. Model and Problem Definition

Suppose that there are M attributes, N instances and D agents. Define the set of indexes of attributes as

$$\mathcal{A} = \{1, \ldots, M\}, \tag{1}$$

and the set of indexes of instances as

$$\mathcal{I} = \{1, \ldots, N\}; \tag{2}$$

then the observation data matrix $\mathbf{X}$ composed of elements $x_{ij},\ i \in \mathcal{I},\ j \in \mathcal{A}$ can be divided into many small parts $\mathcal{P}_k,\ k = 1, \ldots, D$, of the general form

$$\mathcal{P}_k = \{x_{ij} : (i,j) \in \mathcal{Q}_k \subset \mathcal{I} \times \mathcal{A}\}. \tag{3}$$

If we add some extra structural constraints to the sets $\mathcal{Q}_k$, then we can describe instance-distributed and attribute-distributed data. To be more specific, if

$$\mathcal{Q}_k = \mathcal{I}_k \times \mathcal{A},\ \mathcal{I}_k \subset \mathcal{I}, \tag{4}$$

then each data set $\mathcal{P}_k$ contains only complete rows of $\mathbf{X}$, i.e., each agent observes all the attributes for a subset of the instances. In this case, we call the collection $\{\mathcal{P}_k,\ k = 1, \ldots, D\}$ instance-distributed data.

On the other hand, if

$$\mathcal{Q}_k = \mathcal{I} \times \mathcal{A}_k,\ \mathcal{A}_k \subset \mathcal{A}, \tag{5}$$

then each data set $\mathcal{P}_k$ contains only complete columns of $\mathbf{X}$, i.e., each agent observes all the instances of a subset of the attributes. In such situation, we call the collection $\{\mathcal{P}_k,\ k = 1, \ldots, D\}$ attribute-distributed data.

Our further discussion is based on an estimation/regression problem with attribute-distributed data. Suppose attribute $X_j \in \mathcal{S}_j,\ j \in \mathcal{A}$, and outcome $Y \in \mathbb{R}$. Then the entire dataset can be written as

$$\{(x_{i1}, x_{i2}, \ldots, x_{iM}, y_i)\}, \quad i \in \mathcal{I},$$

or, for simplicity

$$\{\mathbf{x}_i, y_i\}, \quad i \in \mathcal{I},$$

where x_{ij} is the ith instance of attribute X_j, and $y_i \in \mathbb{R}$ is the ith instance of Y.

We assume that there exists a hidden deterministic function (or rule/hypothesis)

$$\phi : \prod_{j \in \mathcal{A}} \mathcal{S}_j \to \mathbb{R}$$

such that

$$y_i = \phi(x_{i1}, x_{i2}, \ldots, x_{iM}) + w_i \tag{6}$$

where $\{w_i\},\ i \in \mathcal{I}$ is an independently drawn sample from a zero-mean random variable W that is independent of $X_1, \ldots, X_M$ and Y. For simplicity, we denote the hidden rule as

$$y_i = \phi(\mathbf{x}_i) + w_i. \tag{7}$$

Suppose there are D agents and one fusion center. Each of the agents has only limited access to certain attributes. As defined in the introduction, $\mathcal{A}_j$ $(j = 1, \ldots, D)$ denotes the set of attributes accessible by agent j, and $\mathcal{A} = \cup_{j=1}^{D} \mathcal{A}_j$, assuming that $|\mathcal{A}| = M$. For centralized data, the set of possible estimators is given by

$$\mathcal{H} = \left\{ f : \prod_{k \in \mathcal{A}} \mathcal{S}_k \rightarrow \mathbb{R} \right\},$$

and for each agent j, the set is reduced to

$$\mathcal{H}_j = \left\{ f : \prod_{k \in \mathcal{A}_j} \mathcal{S}_k \rightarrow \mathbb{R} \right\}.$$

We will restrict $\mathcal{H}_j$ and $\mathcal{H}$ to include only functions f satisfying

$$\mathbb{E}[f^2(\mathbf{x})] < \infty. \tag{8}$$

The fusion center observes the outcome Y, with all its instances $\{y_i\}$, $i \in \mathcal{I}$. These assumptions specify the "attribute-distributed" properties of our problem.

Suppose $\mathbf{x}$ is the random variable that generates the samples $\{\mathbf{x}_i\}$, $i \in \mathcal{I}$ with a fixed (but unknown) distribution; then ideally, we need to solve the following problem:

$$\min_{\rho \in \mathcal{J}, f_j \in \mathcal{H}_j} \mathbb{E}\left[(\phi(\mathbf{x}) - \rho(f_1(\mathbf{x}), \ldots, f_D(\mathbf{x})))^2 \right] \tag{9}$$

where

$$\rho \in \mathcal{J} = \{ f : \mathbb{R}^D \rightarrow \mathbb{R} \}. \tag{10}$$

If we assume that the ensemble estimator is of additive form

$$\sum_{j=1}^{D} f_j(\mathbf{x}), \tag{11}$$

then problem (9) can be reduced to a simpler form

$$\min_{f_j \in \mathcal{H}_j} \mathbb{E}\left[\left(\phi(\mathbf{x}) - \sum_{j=1}^{D} f_j(\mathbf{x}) \right)^2 \right]. \tag{12}$$

In practice, if we have only finite, noisy data, it is impossible to exactly solve (12). Instead, we use the training error as a proxy of the objective specified in (12). Then, we have the problem

$$\min_{f_j \in \mathcal{H}_j} \sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{D} f_j(\mathbf{x}_i) \right)^2 \tag{13}$$

of minimizing the mean square training error.

In Section III, we concentrate on showing some results concerning problem (12), closely relating to the question "What is the best we can do?" In Section IV, we go further to find a practical algorithm for solving problem (13).

B. Iterative Conditional Expectation Projection

Now, let us consider a very naive approach to iteratively solve problem (12). Suppose that we have two agents. We initialize $f_1^{(0)}(\mathbf{x})$ and $f_2^{(0)}(\mathbf{x})$ both as 0. Suppose $\mathbf{x} = [x_1, \ldots, x_M]^T$, and Agent 1 and Agent 2 then update their estimators in the following manner:

$$f_1^{(t+1)}(\mathbf{x}) = \mathbb{E}[\phi(\mathbf{x}) - f_2^{(t)}(\mathbf{x}) | x_k, k \in \mathcal{A}_1] \tag{14}$$

and

$$f_2^{(t+1)}(\mathbf{x}) = \mathbb{E}[\phi(\mathbf{x}) - f_1^{(t)}(\mathbf{x}) | x_k, k \in \mathcal{A}_2]. \tag{15}$$

In other words, each agent j repeatedly finds the conditional expectation (conditioned on the set of covariates visible to agent j, i.e., $\{x_k : k \in \mathcal{A}_j\}$) of the difference between the hidden rule and the current projection of the other agent. This is compatible with our intuition because the conditional expectations given by (14) and (15) are the minimum-mean-square-error (MMSE) estimators of the fitting residuals of the other agents. For simplicity, we henceforth denote $\mathbb{E}[f(\mathbf{x}) | x_k, k \in \mathcal{A}]$ as $\mathbb{E}[f|\mathcal{A}]$.

If this conditional expectation projection process is iterated, then hopefully we can asymptotically achieve a limit $f_1^{\infty} + f_2^{\infty}$ that best approximates ϕ. To investigate the limit behavior of the iterative conditional expectation projection, we define the distance between two functions f_1 and f_1' on the space $\mathcal{H}_1$ as

$$d(f_1, f_1') = \mathbb{E}[(f_1(\mathbf{x}) - f_1'(\mathbf{x}))^2]; \tag{16}$$

then we can show that the map defined by (14) and (15) from $f_1^{(t)}(\mathbf{x})$ to $f_1^{(t+1)}(\mathbf{x})$ is a non-expansive map. More specifically, we have the following theorem.

Theorem 1: For two functions $f_1 \in \mathcal{H}_1$ and $f_1' \in \mathcal{H}_1$, the map $T : \mathcal{H}_1 \rightarrow \mathcal{H}_1$ specified by (14) and (15), i.e.,

$$T(f_1) = \mathbb{E}\left[\phi - \mathbb{E}[\phi - f_1 | \mathcal{A}_2] | \mathcal{A}_1 \right] \tag{17}$$

satisfies the property

$$d(T(f_1), T(f_1')) \leq d(f_1, f_1'). \tag{18}$$

That is, T is a non-expansive map on the metric space $\mathcal{H}_1$ with distance defined by (16).

Non-expansive is weaker than contractive, and "fixed-point" theorems generally require a contractive map. The following theorem shows that if the covariates of the two agents are jointly Gaussian and the hidden rule is a finite-order polynomial, then T is a contractive map.

Theorem 2: If the hidden rule $\phi : \mathbb{R}^2 \rightarrow \mathbb{R}$ is a finite-order polynomial function with zero mean, and its input covariates $\mathbf{X} = (X_1, X_2)$ are jointly Gaussian with correlation coefficients ρ, then T defined by (17) is a contractive map, with contraction factor ρ^4, i.e.,

$$d(T(f_1), T(f_1')) \leq \rho^4 d(f_1, f_1'). \tag{19}$$

The proofs of Theorems 1 and 2 can be found in Appendixes A and B, respectively.

By the fixed-point theorem for metric spaces, under the assumptions of Theorem 2, the iterative conditional expectation algorithm specified by (14) and (15) converges to a unique limit, which is the solution of the simultaneous equations

$$f_1(x_1) = \mathbb{E}[\phi(x_1, x_2) - f_2(x_2) | x_1] \tag{20}$$

$$f_2(x_2) = \mathbb{E}[\phi(x_1, x_2) - f_1(x_1) | x_2]. \tag{21}$$

Moreover, the convergence speed is $O(\rho^{4n})$, where n is the number of iterations. If the covariates of the two agents are independent, then the algorithm converges in one iteration.

In [26], a detailed example is provided to support Theorem 2 in the section containing our simulation results. In the example, the limit of the iterative conditional expectation projection is exactly the same as the one that is found by solving (20) and (21). Moreover, the distance, as defined by (16), between the estimated function and the limit (i.e., the fixed point) converges with a rate of $O(\rho^{4n})$.

We now turn to the realistic case in which the data are finite and noisy. Moreover, we do not assume that each agent is perfect at finding the conditional expectation of the current ensemble residual projected onto its local data. On the contrary, we assume that the agents may employ different learning algorithms.

With these new relaxed assumptions, we turn to designing an algorithm to solve problem (13) iteratively to create an ensemble estimator with the least possible generalization error.

IV. Methods and Algorithms

A. Minimizing Training Error Sequentially

If we directly employ the iterative conditional expectation algorithm in the last section specified by (14) and (15) and apply it to solve the finite, noisy data case (13), i.e., to refit the current ensemble residual in a round robin manner, then we can obtain Algorithm 1, in which $C(t)$ denotes the index of the agent at iteration t, which, for a round robin refitting, is given by

$$C(t) = (t \bmod D) + 1. \tag{22}$$

Applying Algorithm 1 with $C(t)$ specified by (22), henceforth called *Simple Iterative Projection*, does not achieve very desirable results, especially in the presence of many irrelevant attributes. Since the algorithm treats every agent equally, neglecting the fact that some of them are better at predicting the outcome than others, and thereby projecting the ensemble residual onto some spaces orthogonal to the outcome, it can overfit the data.

Algorithm 1: Prototype Residual-Refitting Algorithm

1 $F_0(\mathbf{x}) \leftarrow \frac{1}{N}\sum_{i=1}^{N} y_i$
2 **for** $t = 1, \ldots, T$ **do**
3 $\quad \hat{y}_i^{(t)} \leftarrow y_i - F_{t-1}(\mathbf{x}_i),\ \forall i \in \mathcal{I}$
4 $\quad f^{(t)}(\mathbf{x}) \leftarrow \arg\min_{f \in \mathcal{H}_{C(t)}} \sum_{i=1}^{N} (\hat{y}_i^{(t)} - f(\mathbf{x}_i))^2$
5 $\quad F_t(\mathbf{x}) \leftarrow F_{t-1}(\mathbf{x}) + f_t(\mathbf{x})$
6 **end**
7 $f_j(\mathbf{x}) = \sum_{t \in C^{-1}(j)} f^{(t)}(\mathbf{x}),\ j \in \{1, \ldots, D\}$

Before we explore other algorithms that improve on Simple Iterative Projection, it is worth noting the resemblance of our algorithm to the L_2 boosting algorithm [2], as shown in Algorithm 2.

Algorithm 2: L_2 Boosting Algorithm

1 $F_0(\mathbf{x}) \leftarrow \frac{1}{N}\sum_{i=1}^{N} y_i$
2 **for** $t = 1, \ldots, T$ **do**
3 $\quad \hat{y}_i^{(t)} \leftarrow y_i - F_{t-1}(\mathbf{x}_i), \forall i \in \mathcal{I}$
4 $\quad f^{(t)}(\mathbf{x}) \leftarrow \arg\min_{f \in \mathcal{H}} \sum_{i=1}^{N} (\hat{y}_i^{(t)} - f(\mathbf{x}_i))^2$
5 $\quad F_t(\mathbf{x}) \leftarrow F_{t-1}(\mathbf{x}) + f^{(t)}(\mathbf{x})$
6 **end**

Notice that the major difference between Algorithm 2 and Algorithm 1 is in Line 4, where the sets of functions from which the latest estimator of the ensemble residual can be selected are different for different iterations in Algorithm 1. For the distributed case, we have to select the estimator from a smaller space $\mathcal{H}_j \subset \mathcal{H}$ where $j = C(t)$. If we can expand the space somehow, and search for the latest estimator in a larger set of functions, then we can make the algorithm more efficient in terms of reducing the training mean square error.

One possible choice is that we replace $\mathcal{H}_j$ in Line 4 of Algorithm 1 by $\bigcup_{j=1}^{D} \mathcal{H}_j$, or equivalently, that we redefine $C(t)$ as follows:

$$C(t) = \arg\min_{j \in \{1,\ldots,D\}} \sum_{i=1}^{N} \left(\hat{y}_i^{(t)} - f_j^{(t)}(\mathbf{x}_i)\right)^2 \tag{23}$$

where

$$f_j^{(t)}(\mathbf{x}) = \arg\min_{f \in \mathcal{H}_j} \sum_{i=1}^{N} \left(\hat{y}_i^{(t)} - f(\mathbf{x}_i)\right)^2. \tag{24}$$

In other words, for each iteration t, the fusion center sends the residual to all the agents $j = 1, \ldots, D$, each of which finds a local optimal estimator $f_j^{(t)}$ based on its own limited set of functions $\mathcal{H}_j$, and then the fusion center chooses the agent that generates the estimator with the smallest training error to project the current residual. We call this algorithm the *greedy algorithm*, which is a distributed version of L_2 boosting, with extra constraints on the set of functions from which the latest estimator can be selected.

Nonetheless, in the greedy algorithm, the information of the ensemble residual is still used in a rather myopic manner. As pointed out in [20], the algorithm can be overaggressive in reducing the training error at each step, and hence may discard some relevant agents, while using the predictions of estimators that are orthogonal to the outcome. If we use the training residual information of every agent in a more holistic manner, we may be able to achieve better results.

B. Redefine the Model

Before we introduce our parallel approach, we need to reformulate another model that is equivalent to (13), given that $\mathcal{H}_j,\ j = 1, \ldots, D$ are linear.

In the new model, we still stipulate that the ensemble estimator has an additive form, but this time we introduce a weighting coefficient β_j for each agent. Thus, we have a new

optimization problem:

$$\min_{f_j\in\mathcal{H}_j,\beta_j}\sum_{i=1}^{N}\left(y_i-\sum_{j=1}^{D}\beta_j f_j(\mathbf{x}_i)\right)^2. \tag{25}$$

Since $f_j\in\mathcal{H}_j$ implies $\beta f_j\in\mathcal{H}_j$ (due to the linearity of $\mathcal{H}_j$), the set of functions over which we search in (25) does not expand relative to (13) so that the two problems are equivalent. Now we convert (25) into a two-stage optimization:

$$\min_{f_j\in\mathcal{H}_j}\min_{\beta_j}\sum_{i=1}^{N}\left(y_i-\sum_{j=1}^{D}\beta_j f_j(\mathbf{x}_i)\right)^2. \tag{26}$$

First, let us observe the inner stage of (26), which can be interpreted as: given $f_j(\mathbf{x}), j=1,\ldots,D$ fixed, find the best linear combination so that the ensemble estimator has minimal mean-square training error:

$$\min_{\beta_j}\sum_{i=1}^{N}\left(y_i-\sum_{j=1}^{D}\beta_j f_j(\mathbf{x}_i)\right)^2. \tag{27}$$

Define the vector $\mathbf{y}$ as $[y]_i=y_i$, the vector $\mathbf{f}_j$ as $[\mathbf{f}_j]_i=f_j(\mathbf{x}_i)$, the vector $\boldsymbol{\beta}$ as $[\boldsymbol{\beta}]_j=\beta_j$ and the matrix $\mathbf{F}\in\mathbb{R}^{N\times D}$ as $\mathbf{F}=[\mathbf{f}_1,\ldots,\mathbf{f}_D]$. Then if the individual estimator of each agent is given (i.e., $\mathbf{F}$ is given and fixed) as in the meta-learning case, we can rewrite the optimization problem in (27) as

$$\min_{\boldsymbol{\beta}}\|\mathbf{y}-\mathbf{F}\boldsymbol{\beta}\|^2 \tag{28}$$

which is a standard linear least squares regression problem, and can be readily solved analytically: the solution is given by

$$\hat{\boldsymbol{\beta}}=(\mathbf{F}^T\mathbf{F})^{-1}\mathbf{F}^T\mathbf{y} \tag{29}$$

with the minimal value given by

$$\mathcal{L}(\hat{\boldsymbol{\beta}},\mathbf{F})=\|\mathbf{y}-\mathbf{F}\hat{\boldsymbol{\beta}}\|^2=\mathbf{y}^T(\mathbf{I}-\mathbf{F}(\mathbf{F}^T\mathbf{F})^{-1}\mathbf{F}^T)\mathbf{y}. \tag{30}$$

Interestingly, in a cooperative training algorithm, the agents have opportunities to change their training residuals so as to reshape $\mathbf{F}$ and hence reduce the optimal value of (30), thereby minimizing the ensemble training error. It is this stage of optimization that distinguishes the cooperative problem from linear least squares regression.

In the second stage, the optimization problem thus becomes

$$\min_{\mathbf{F}\in\mathcal{F}}\mathbf{y}^T(\mathbf{I}-\mathbf{F}(\mathbf{F}^T\mathbf{F})^{-1}\mathbf{F}^T)\mathbf{y} \tag{31}$$

which is equivalent to

$$\max_{\mathbf{F}\in\mathcal{F}}\mathbf{y}^T\mathbf{F}(\mathbf{F}^T\mathbf{F})^{-1}\mathbf{F}^T\mathbf{y} \tag{32}$$

where

$$\mathcal{F}=\{\mathbf{F}\in\mathbb{R}^{N\times D}:\exists f_j\in\mathcal{H}_j,\ \text{s.t. }[\mathbf{F}]_{ij}=f_j(\mathbf{x}_i)\ \forall i,j\}, \tag{33}$$

i.e., $\mathcal{F}$ is the set in which all possible prediction matrices $\mathbf{F}$ reside.

Solving (32) requires an iterative algorithm since the training constraint $\mathcal{F}$ is not explicit. A gradient descent (in this case, it is actually hill climbing) algorithm to optimize (32) requires an explicit expression for the gradient of $\eta=\mathbf{y}^T\mathbf{F}(\mathbf{F}^T\mathbf{F})^{-1}\mathbf{F}^T\mathbf{y}$ with respect to $\mathbf{F}$, which is

$$\frac{\partial\eta}{\partial\mathbf{F}}=2(\mathbf{y}-\mathbf{F}\hat{\boldsymbol{\beta}})\hat{\boldsymbol{\beta}}^T. \tag{34}$$

The derivation of (34) can be found in Appendix C.

Note that the gradient for $\mathbf{F}$ is simply the current training residual $\mathbf{R}=\mathbf{y}-\mathbf{F}\hat{\boldsymbol{\beta}}$, reweighted by $\hat{\beta}_j$ for the jth column. If we solve this problem sequentially, i.e., we send the current residual to one of the agents to refit, then this closely relates the second stage of our optimization to the L_2 boosting algorithm, introduced and developed in [2] and [7], in which the training residual is repeatedly refitted. The basic idea of the second stage optimization is now clear. Namely, given that the optimal weighting coefficient can always be determined analytically, the system repeatedly updates its basis $\mathbf{f}_j$, and hence reshapes $\mathbf{F}$ so that the ensemble training mean square error can be minimized.

C. Parallel Gradient Descent

As stated above, we use a gradient descent algorithm to solve the optimization problem (32). The difficult part is to consolidate the training constraint $\mathcal{F}$ for each agent. Since the specification of $\mathcal{H}_j$ is implicit, the most straightforward way to use $\mathcal{H}_j$ is through repeated training.

Moreover, there is a practical issue in our algorithm. In particular, if two agents observe exactly the same data and hence generate the same estimator, the objective becomes singular and the optimization becomes meaningless. To prevent this from happening, one needs either to bootstrap the data for each agent, or to add a regularization term to the original objective of the optimization problem:

$$\min_{\boldsymbol{\beta}}\|\mathbf{y}-\mathbf{F}\boldsymbol{\beta}\|^2+\lambda\|\boldsymbol{\beta}\|^2. \tag{35}$$

Based on the above ideas, the gradient descent algorithm is described as follows: First each agent fits the outcome as well as it can, and the fusion center finds a best linear combination of the agents' predictions on the training set. Based on the weighting coefficients, the fusion center finds the gradient for the prediction matrix $\Delta\mathbf{F}$ and uses a back-search algorithm to find the optimal step length δ. Then the fusion center sends the desirable next-step prediction to each agent, which fits to this ideal next-step prediction as well as it can. The fusion center again updates the weighting coefficients and repeats the above process until the training error stops decreasing. The algorithm described above is specified in Algorithm 3.

Algorithm 3: Parallel Algorithm

1 $f_j^{(0)} \leftarrow \arg\min_{f\in\mathcal{H}_j} \sum_{i=1}^{N}(y_i - f(\mathbf{x}_i))^2,\ \forall j$
2 $[\mathbf{F}]_{ij} \leftarrow f_j(\mathbf{x_i}),\ \forall i, j$
3 $\hat{\boldsymbol{\beta}} = (\mathbf{F}^T\mathbf{F})^{-1}\mathbf{F}\mathbf{y}$
4 **for** $t = 1, \ldots, T$ **do**
5 $\quad \Delta\mathbf{F} \leftarrow (\mathbf{y} - \mathbf{F}\hat{\boldsymbol{\beta}})\hat{\boldsymbol{\beta}}$
6 $\quad \delta \leftarrow \mathrm{BackSearch}(\alpha,\ \beta,\ \mathbf{F},\ \Delta\mathbf{F})$
7 $\quad \hat{\mathbf{F}} \leftarrow \mathbf{F} + \delta_t\Delta\mathbf{F}$
8 $\quad f_j \leftarrow \arg\min_{f\in\mathcal{H}_j} \sum_{i=1}^{N}([\mathbf{F}]_{ij} - f(\mathbf{x}_i))^2$
9 $\quad [\mathbf{F}]_{ij} \leftarrow f_j(\mathbf{x}_i)$
10 $\quad \hat{\boldsymbol{\beta}} = (\mathbf{F}^T\mathbf{F})^{-1}\mathbf{F}\mathbf{y}$
11 **end**

The $\mathrm{BackSearch}(\alpha,\ \beta,\ \mathbf{F},\ \Delta\mathbf{F})$ function in Line 6 of Algorithm 3 is designed to find the optimal step length δ to expedite the convergence of the training error, where $\alpha \in (0,\ 0.5)$ and $\beta \in (0,\ 1)$. Algorithm 4 summarizes this procedure.

Algorithm 4: Back-Search Algorithm

1 **function** $\mathrm{BackSearch}(\alpha,\ \beta,\ \mathbf{F},\ \Delta\mathbf{F})$
2 $\delta \leftarrow 1$
3 $\hat{\mathbf{F}} \leftarrow \mathbf{F} + \delta\Delta\mathbf{F}$
4 **define** $\mathcal{E}(\mathbf{F},\ \mathbf{y}) = \mathbf{y}^T\mathbf{F}(\mathbf{F}^T\mathbf{F})^{-1}\mathbf{F}^T\mathbf{y}$
5 **while** $\mathcal{E}(\hat{\mathbf{F}}, y) < \mathcal{E}(\mathbf{F}, y) + \alpha\delta\mathrm{Tr}(\Delta\mathbf{F}^T\Delta\mathbf{F})$ **do**
6 $\quad \delta \leftarrow \beta\delta$
7 $\quad \hat{\mathbf{F}} \leftarrow \mathbf{F} + \delta\Delta\mathbf{F}$
8 **end**
9 **return** δ

D. Beyond Gradient Descent

There are two sources of overtraining in distributed algorithms. First, the local training algorithm of each agent may lead to overtraining. The fusion center does not have control over this type of overtraining. Second, including overtrained estimators in the ensemble estimator also may lead to overtraining. Those agents whose observations are irrelevant to the outcome will produce this type of overtrained estimators. The fusion center, however, can decide whether to incorporate an estimator into the final ensemble estimator or not. Avoiding the second type of overtraining is essential to the distributed algorithm.

Applying gradient descent directly to (32) may lead to overtraining, since irrelevant attributes will still play a role in the final ensemble estimator. This raises the question: is there a way to iteratively lower the weight of irrelevant estimators in the final ensemble estimator?

One possible approach to this problem is to modify the update direction for iteratively solving the optimization problem (32). Note that the gradient (34) is merely a weighted version of the ensemble training residual. The training residual is redistributed to each agent by its weighting coefficient β_j. Intuitively, the larger the magnitude of β_j, the more important the jth agent is in the ensemble estimator. Those agents whose weighting coefficients are rather small in absolute value are less likely to be relevant agents. Therefore, we can adjust the reweighting process of the training residual so that the more promising agents can obtain a larger portion of the residual, while the less promising agents are assigned a smaller portion of the residual. In this way, we can iteratively phase out irrelevant agents.

A revised version of the gradient descent algorithm based on this idea will use a different searching direction for (34) as follows:

$$(\mathbf{y} - \mathbf{F}\hat{\boldsymbol{\beta}})\mathcal{W}(\hat{\boldsymbol{\beta}})^T. \tag{36}$$

The reweighting function $\mathcal{W} : \mathbb{R}^D \rightarrow \mathbb{R}^D$ should emphasize components of $\hat{\boldsymbol{\beta}}$ that are relatively large in absolute value and suppress components of $\hat{\boldsymbol{\beta}}$ that are closer to zero. There are many choices of such functions, and we could even use a clustering algorithm to distinguish significant agents and irrelevant agents. Here, for simplicity, we consider the power function

$$[\mathcal{W}(\boldsymbol{\beta})]_j = \mathrm{sgn}([\boldsymbol{\beta}]_j)|[\boldsymbol{\beta}]_j|^p \tag{37}$$

where $p \geq 1$ controls the preference to larger values of coefficients. When $p = 1$, the gradient reduces to the original one. It is thus interesting to investigate the influence of $\mathcal{W}$ on the generalization error of Algorithm 3, by replacing Line 5 by (36).

Before we apply the revised search direction to our algorithm, if the optimal solution is path independent, then no matter what path we go through to reach it, we will achieve the same result. To make the above approach work, we need to show that the solution to problem (32) is actually path-dependent.

For agent j, define

$$\mathcal{F}_j = \left\{\mathbf{f} \in \mathbb{R}^N : \exists f \in \mathcal{H}_j,\ \text{s.t. } [\mathbf{f}]_i = f(\mathbf{x}_i),\ \forall i \in \mathcal{I}\right\}. \tag{38}$$

Then, suppose that each space $\mathcal{F}_j$ is spanned by M_j orthonormal basis vectors $\{\mathbf{q}_{j1}, \ldots, \mathbf{q}_{jM_j}\}$. On defining

$$\mathbf{Q}_j = [\mathbf{q}_{j1}, \ldots, \mathbf{q}_{jM_j}], \tag{39}$$

the prediction of agent j can be written as a linear combination of the basis vectors, i.e.,

$$\mathbf{f}_j = \mathbf{Q}_j\boldsymbol{\gamma}_j. \tag{40}$$

When the back search algorithm terminates, the residual must be orthogonal to all the basis vectors of every agent. Therefore, we have

$$\mathbf{Q}_j^T(\mathbf{y} - \mathbf{F}\boldsymbol{\beta}) = 0,\ \forall j \tag{41}$$

where $\mathbf{F} = [\mathbf{f}_1, \ldots, \mathbf{f}_D]$ is determined by (40), and $\boldsymbol{\beta} = (\mathbf{F}^T\mathbf{F})^{-1}\mathbf{F}^T\mathbf{y}$.

We can rewrite (41) as

$$\mathbf{Q}_j^T\mathbf{y} = \mathbf{Q}_j^T\mathcal{K}(\boldsymbol{\gamma}_1, \ldots, \boldsymbol{\gamma}_D)\mathbf{y},\ \forall j \tag{42}$$

where

$$\mathcal{K}(\boldsymbol{\gamma}_1, \ldots, \boldsymbol{\gamma}_D) = \mathbf{F}(\mathbf{F}^T\mathbf{F})^{-1}\mathbf{F}^T. \tag{43}$$

Notice that (42) can be simplified into a polynomial equation of high order (each of the equations has order $2D$). As long as $\{\mathbf{Q}_j\}_{j=1}^D$ are not trivial (e.g., mutually identical), we have an equal number of unknown variables and equations. This means that we have many solutions for $\{\boldsymbol{\gamma}_j\}_{j=1}^D$, and problem (32) is indeed path dependent. The simulations in the next section will demonstrate the influence of search direction on the generalization error of the ensemble estimator.

V. Simulation Results

In this section, we examine the efficacy of our parallel algorithm (Algorithm 3 in Section IV) and its variants on several different data sets, and compare it with Simple Iterative Projection (Algorithm 1 with (22) in Section IV) and the greedy algorithm (Algorithm 1 with (23) in Section IV) in different simulation scenarios.

A. Generalization Error

We test the algorithms on both artificial and real data sets. For artificial data, we employ three functions used in [18] (originally from [8]) as the hidden rule to generate our simulation training data sets. The three functions and the corresponding joint distributions of the covariates are as follows:

- Friedman-1:

$$\phi(\mathbf{x}) = 10\sin(\pi x_1 x_2) + 20\left(x_3 - \frac{1}{2}\right)^2 + 10x_4 + 5x_5 + w$$

  where $x_j \sim U[0,1], j = 1 \ldots 5$;
- Friedman-2:

$$\phi(\mathbf{x}) = \left(x_1^2 + \left(x_2 x_3 - \frac{1}{x_2 x_4}\right)^2\right)^{1/2} + w$$

  where $x_1 \sim U[1,100]$, $x_2 \sim U[40\pi, 560\pi]$, x_3, $x_5 \sim U[0,1]$ and $x_4 \sim U[1,11]$;
- Friedman-3:

$$\phi(\mathbf{x}) = \tan^{-1}\left(\frac{x_2 x_3 - \frac{1}{x_2 x_4}}{x_1}\right) + w$$

  where the distributions of the attributes are the same as those of Friedman-2.

All these attributes are independent of one another, and before running the algorithm, the variance of the outcomes is normalized to 1. In our simulations, we choose 2000 training data instances and 2000 test data instances, and the standard deviation of white noise w is set to $\sigma = 0.05$.

Moreover, to examine the algorithms' performance on real data, we further test them on the concrete compressive strength (CCS) dataset from the UCI Machine Learning Repository. This is a multivariate regression dataset with $M = 8$ attributes and 1030 instances, from which we randomly select $N = 700$ as the training set, leaving the rest for testing. Moreover, since in practice it is very likely that there exist some attributes irrelevant to the outcome, to test the algorithms under the presence of nuisance attributes, we added eight nuisance attributes to the CCS dataset and also compared the algorithms.

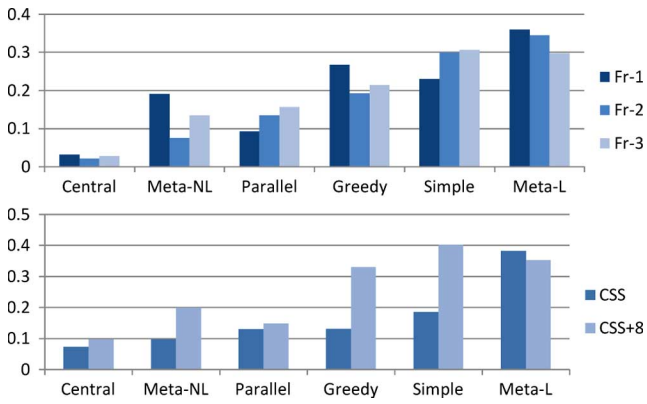


Fig. 3. Comparison of the test errors (mean square error) of the centralized L_2 boosting, linear/non-linear meta-learning, parallel algorithm, the greedy algorithm, and Simple Iterative Projection on the data sets Friedman-1, -2, -3, CCS and CCS with eight irrelevant attributes.

The distributed system is constructed as follows: Assume that M is the total number of attributes, and there are $D = M$ agents, with each agent j observing attribute j, i.e., each agent observes only one unique feature. In addition, each agent uses a regression tree (with a constraint on the smallest leaf) as its individual estimator.

For comparison, we also introduce three other algorithms, centralized learning, meta-learning with a linear model, and meta-learning with a non-linear model. The centralized learning is a L_2 boosting algorithm with a discounting factor $\nu = 0.25$. The meta-learning with a linear model simply takes the output of individual agents as its input, and use an L_2 linear regression to refit a model; its non-linear counterpart also takes the same input, yet it employs the L_2 boosting algorithm to process the agents' predictions.

The simulation results are shown in Fig. 3, which vividly illustrates the comparison among the algorithms on different data sets. It is conspicuous that on all the datasets, the parallel scheme outperforms the other distributed algorithms with linear models (greedy algorithm and Simple Iterative Projection, and meta-learning with a linear model).

Moreover, in the right subgraph of Fig. 3, it can be seen that the performance of the parallel scheme on the CCS data set is not affected very much by the introduction of eight irrelevant variables (the same number of relevant variables). However, most of the other distributed learning algorithms almost double their test error in MMSE. This "agent selection" property of the parallel scheme is highly desirable in the presence of irrelevant attributes. In the next subsection, we investigate how to further enhance this advantage of the parallel scheme.

It is also worth pointing out that the test errors for the greedy algorithm and Simple Iterative Projection are selected at the optimal stopping time (according to the true test error), while the test error for the parallel algorithm is blindly selected after a fixed number of iterations. This can be better demonstrated through Fig. 4.

It is apparent that the greedy algorithm, with the complexity of its ensemble estimator growing aggressively, suffers from

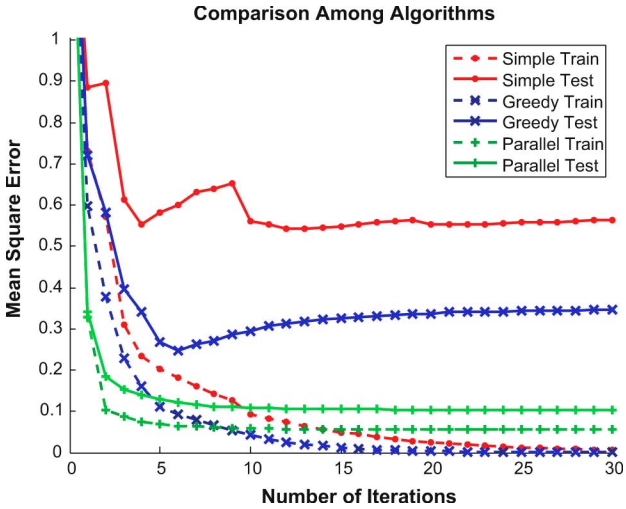


Fig. 4. Comparison between the convergence of the parallel algorithm, the greedy algorithm and Simple Iterative Projection for the Friedman-1 dataset with 5 irrelevant attributes. The parallel algorithm is least susceptible to overtraining than the greedy algorithm and Simple Iterative Projection, and the training error curve for the parallel algorithm parallels its test error curve with a small gap between them. Yet for the greedy algorithm and Simple Iterative Projection, though the training errors converge rapidly, they do not correctly reflect the test errors of the ensemble estimator, with large gaps and opposite trends.

TABLE I
TEST ERRORS (MEAN SQUARE) OF THE PARALLEL ALGORITHM OF REWEIGHTING FUNCTION (37) OF DIFFERENT VALUES OF p. THE OPTIMAL VALUE IS ACHIEVED WHEN $p \in (3, 4)$

p	1	2	3	4	5	6	7
Errors	.242	.204	.192	.190	.202	.222	.257

overtraining, while the parallel algorithm, with the ensemble estimator complexity fixed, remains effective once reaching a good result.

The overtraining problem of the greedy algorithm can be partly solved by taking steps more cautiously (by multiplying a discount factor to the prediction when updating the ensemble residual), similarly to stagewise regression. Yet this may greatly reduce the speed of convergence of the algorithm, and add a heavy burden to the communication between the fusion center and the agents.

B. Performance in the Presence of Irrelevant Attributes

It is also of interest to test the capability of the parallel algorithm to eliminate irrelevant variables, especially in the presence of a significant fraction of them. Moreover, we are also interested in the possible improvement of performance by introducing the weighting function $\mathcal{W}(\boldsymbol{\beta})$. We use the Friedman-3 data set (1000 training data points and 1000 test data points) with noise level $\sigma = 0.05$ for the simulation. There are only four relevant attributes in the data set, and we mix 26 irrelevant attributes (white noise) into the data set so that the relevant attributes comprise less than 14% of all the attributes.

Under this setting, keeping the configuration of the distributed system unchanged, we test our algorithm with adjusted gradient by reweighting the residual using $\mathcal{W}(\boldsymbol{\beta})$ in the form of (37). The results in terms of mean square test errors are shown in Table I. For comparison, the error for the greedy algorithm for this scenario is 0.2746.

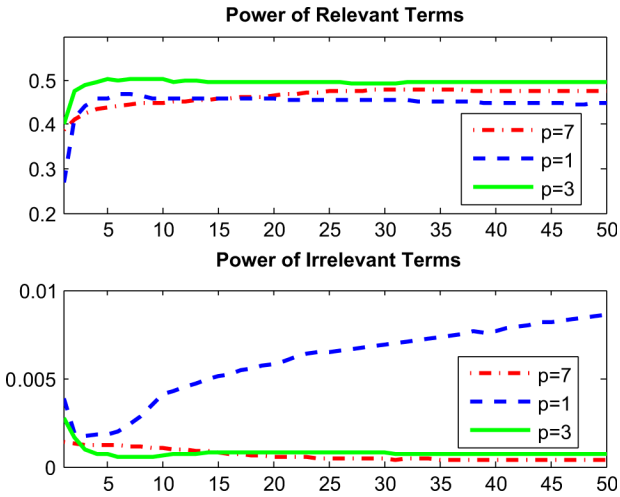


Fig. 5. Comparison between power of relevant terms (44) and power of irrelevant terms (45) in the final ensemble estimator for different values of p in the reweighting function (37).

It is clear that there is an optimal value of p in terms of test errors. To show further detail about the influence of $\mathcal{W}$ and its connection to the test error of the ensemble estimator, we compare the sum of the power of the predictions of the agents with relevant attributes and that of the agents with irrelevant attributes. To be more specific, let $\mathcal{R}$ be the set of relevant attributes, and $\mathcal{O}$ be the set of irrelevant attributes. We then compare

$$\sum_{j \in \mathcal{R}} \|\beta_j \mathbf{f}_j\|^2 \tag{44}$$

to

$$\sum_{j \in \mathcal{O}} \|\beta_j \mathbf{f}_j\|^2. \tag{45}$$

The results are shown in Fig. 5. This provides us with insight into why the use of the weighting function $\mathcal{W}$ improves the performance of the system. When we do not de-emphasize the irrelevant variables, as in the case of $p = 1$, the total power of terms corresponding to irrelevant estimators (45) increases as the number of iterations increases. This means that the fusion center allocates a larger portion of the outcome to the irrelevant variables. This is a major source of the second type of overtraining, and drags down the performance of the vanilla version of the parallel algorithm.

On the other hand, when we oversuppress the irrelevant variables, as in the case of $p = 7$, we achieve a very small value of (45), yet the sum of the powers of the relevant estimators (44) is also smaller. Instead of overtraining, the system "under-uses" some of the relevant variables such that the ensemble estimator does not fully explain the outcome using relevant data.

Achieving a balance between these two extremes, as in the case of $p = 3$, we can both eliminate the irrelevant attributes and fully use the relevant attributes. This requires a proper choice of $\mathcal{W}$, which depends on the data. It is desirable to design adaptive algorithms to automatically select a proper reweighting function based on the data itself.

TABLE II
COMMUNICATION COST FOR ATTRIBUTE-DISTRIBUTED ALGORITHMS

	Average	Simple	Greedy	Parallel
Fusion Center	0	N	N	N
Active Agent	0	N	N	DN
Other Agents	0	0	$O(1)$	-
Convergence	Instant	Slow	Fast	Fast

VI. DISCUSSION

A. Communication Issues

In practice, the residuals or predictions of each agent, i.e., the columns of $\mathbf{F}$, need to be transmitted among the agents under the coordination of a fusion center. Therefore, it is of interest to look into the amount of data that must be exchanged for different algorithms.

For the greedy algorithm, for each iteration, the fusion center sends the current residual to every agent and receives from them the correlation coefficients as feedback. Deciding which agent to select, the fusion center again asks for the updated residual from the selected agent, who sends it back to the fusion center. This completes an iteration. The fusion center needs to broadcast N data points to each agent, and each agent needs to feed back an $O(1)$ indicator, and one of them needs to further feed N data points back.

The parallel algorithm on the other hand, needs more data transmission. For each iteration, the fusion center needs to broadcast the current residual and send each agent (N data instances) its weighting coefficients. Then each agent updates its individual estimator and sends back to the fusion center its new prediction on the training set (N data instances).

Moreover, if $C(t)$ in Algorithm 1 follows some deterministic sequence as in the case of Simple Iterative Projection, then we can greatly reduce the amount of data exchanged for each iteration, to only $O(N)$ for one active agent. This is further discussed in [23], where the value of $C(t)$ for the next iteration can be predicted by some heuristics to reduce the amount of data to be exchanged.

Table II summarizes the communication cost for several typical attribute-distributed algorithms for each iteration during training.

B. The Role of Fusion Center

Throughout the paper, to keep the problem simple, we have kept a fusion center in the system. However, for all our algorithms, the Simple Iterative Projection, the greedy algorithm and the parallel algorithm, there is a possible implementation without a fusion center.

For instance, the Simple Iterative Projection algorithm requires only sending the latest training residual to the next agent in a predefined order. As long as the communication links among the agents form one connected component of a graph, it is always possible to project the latest residual to the next agent.

The greedy algorithm needs to achieve two things: First, select the agent that reduces the ensemble training residual the most. This is equivalent to the problem of finding the minimum value distributed system. Second, the selected agent must broadcast its updated training residual to every agent. These two tasks can be easily achieved by flooding or gossip like algorithms.

For the parallel algorithm, it is rather difficult not to have a fusion center, because each update needs full knowledge of the prediction matrix $\mathbf{F}$. As long as all the agents are connected as one component, this can also be achieved by flooding/gossiping. Moreover, once we reach a point at which some agents have been proved to be irrelevant to the outcome, we can stop some agents from sending their own training residuals and reduce the amount of data needed.

Therefore, keeping the fusion center in our system is purely for the elegance and simplicity of the problem. In practice, we can achieve the same goals in a purely distributed system.

VII. CONCLUSION

In this paper, we have introduced a framework for distributed learning, especially for regression problems, on attribute-distributed data. We have introduced two models: the ideal case (12) and the realistic case (13). Based on the ideal case (12), the convergence properties of attribute-distributed regression with an additive model have been discussed, and the convergence rate and uniqueness of the limit have been proven for the case in which the attributes of the two agents are jointly Gaussian, and the hidden rule is a finite order polynomial. Then, taking residual refitting, the classical machine learning counterpart of which being L_2 boosting, as the prototype algorithm, we have developed and compared three different schemes, Simple Iterative Projection, the greedy algorithm and a parallel algorithm (with its derivatives). In terms of the amount of data needed to be exchanged for training, sequential Simple Iterative Projection is the simplest and the parallel algorithm the most intensive. On the other hand, the parallel approach, by using individual residual information holistically, performs the best in terms of generalization. Moreover, designing residual redistribution schemes that emphasize more promising agents further improves the capability of eliminating irrelevant attributes. In addition, we have noted that the fusion center of the system is not required: all three algorithms have corresponding decentralized implementations without a fusion center.

APPENDIX A
PROOF OF THEOREM 1

For simplicity, we denote the conditional expectation $\mathbb{E}[f(\mathbf{x})|x_k,\ k \in \mathcal{A}]$ as $\mathbb{E}[f|\mathcal{A}]$. Then, the distance between two mapped functions can be simplified as

$$\begin{aligned} d(T(f_1), T(f_1')) &= \mathbb{E}\left[\left(T(f_1) - T(f_1')\right)^2\right] \\ &= \mathbb{E}\left[\left(\mathbb{E}\left[\mathbb{E}\left[f_1 - f_1'|\mathcal{A}_2\right]|\mathcal{A}_1\right]\right)^2\right]. \end{aligned}$$

By definition, $d(f_1,\ f_1') = \mathbb{E}\left[(f_1 - f_1')^2\right]$. Define $g = f_1 - f_1' \in \mathcal{H}_1$; then we only need to show that

$$\mathbb{E}\left[\left(\mathbb{E}[\mathbb{E}[g|\mathcal{A}_2]|\mathcal{A}_1]\right)^2\right] \le \mathbb{E}[g^2]. \tag{46}$$

Define $\mu = \mathbb{E}[g]$, then $\mathbb{E}\left[g^2\right] - \mu^2 = \mathbb{E}\left[(g-\mu)^2\right]$. Notice the fact

$$\mathbb{E}\left[\left(\mathbb{E}[\mathbb{E}[g|\mathcal{A}_2]|\mathcal{A}_1]\right)^2\right] - \mu^2 = \mathbb{E}\left[\left(\mathbb{E}[\mathbb{E}[g|\mathcal{A}_2]|\mathcal{A}_1] - \mu\right)^2\right];$$

equation (46) is then equivalent to

$$\mathbb{E}\left[\left(\mathbb{E}[\mathbb{E}[g|\mathcal{A}_2]|\mathcal{A}_1] - \mu\right)^2\right] \leq \mathbb{E}\left[(g-\mu)^2\right], \qquad (47)$$

of which the left-hand side satisfies

$$\begin{aligned} LHS =& \mathbb{E}\left[\left(\mathbb{E}[\mathbb{E}[g|\mathcal{A}_2]|\mathcal{A}_1] - \mathbb{E}[\mathbb{E}[g]|\mathcal{A}_1]\right)^2\right] \\ =& \mathbb{E}\left[\left(\mathbb{E}\left[\mathbb{E}[g|\mathcal{A}_2] - \mathbb{E}[g]|\mathcal{A}_1\right]\right)^2\right] \\ \leq& \mathbb{E}\left[\mathbb{E}\left[\left(\mathbb{E}[g|\mathcal{A}_2] - \mathbb{E}[g]\right)^2|\mathcal{A}_1\right]\right] \\ =& \mathbb{E}\left[\left(\mathbb{E}[g|\mathcal{A}_2] - \mathbb{E}[g]\right)^2\right]. \end{aligned}$$

The inequality step is due to *Jensen's inequality*, and the last step is due to the *tower property*.

On noting that

$$\mathbb{E}\left[\left(\mathbb{E}[g|\mathcal{A}_2] - \mathbb{E}[g]\right)^2\right] = \mathrm{Var}\left[\mathbb{E}[g|\mathcal{A}_2]\right] \qquad (48)$$

we hence have that the left-hand side satisfies

$$\mathrm{LHS} \leq \mathrm{Var}\left[\mathbb{E}[g|\mathcal{A}_2]\right].$$

Moreover, the right-hand side of (47) satisfies

$$\mathrm{RHS} = \mathrm{Var}[g].$$

On noting an important relationship

$$\mathrm{Var}[g] \geq \mathrm{Var}\left[\mathbb{E}[g|\mathcal{A}_2]\right] + \mathbb{E}[\mathrm{Var}[g|\mathcal{A}_2]], \qquad (49)$$

we hence have

$$\mathrm{RHS} \geq \mathrm{LHS} + \mathbb{E}[\mathrm{Var}[g|\mathcal{A}_2]] \geq \mathrm{LHS}. \qquad (50)$$

Equation (50) guarantees that map T is a non-expansive map.

Appendix B
Proof of Theorem 2

Lemma: Suppose g is a polynomial of order M, $g(x) = \sum_{n=0}^{M} a_n x^n$, and $g'(x)$ is given by

$$g'(x) = \int_{-\infty}^{\infty}\left(\int_{-\infty}^{\infty} g(x) f_{X|Y}(x|y)\mathrm{d}x\right) f_{Y|X}(y|x)\mathrm{d}y.$$

Then, with the additional assumption that $\int_{-\infty}^{\infty} g(x) f_X(x)\mathrm{d}x = 0$, we have the inequality

$$\frac{\int_{-\infty}^{\infty} g'^2(x) f_X(x)\mathrm{d}x}{\int_{-\infty}^{\infty} g^2(x) f_X(x)\mathrm{d}x} \leq \rho^4$$

where $f_{X|Y}$, $f_{Y|X}$, and f_X are all probability densities derived from the joint Gaussian distribution with zero mean, unit variance and correlation ρ.

Proof: The conditional distribution of X given Y and the distribution of Y given X are, respectively, $X|Y \sim N(\rho y, 1-\rho^2)$, and $Y|X \sim N(\rho x, 1-\rho^2)$. Therefore, we have

$$g'(x) = \sum_{n=0}^{M} a_n \frac{\mathrm{d}^n}{\mathrm{d}t^n} \exp\left\{\rho^2 x t + \frac{(1-\rho^4)t^2}{2}\right\}\Big|_{t=0}.$$

Notice that the exponential term, with proper manipulation, can be expressed in the form of the sum of Hermite polynomials. Thus, on defining

$$X = \frac{\rho^2}{i\sqrt{2(1-\rho^2)}}x, \quad \text{and} \quad s = i\sqrt{\frac{1-\rho^4}{2}}t,$$

we have

$$\exp\left\{\rho^2 x t + \frac{1-\rho^4}{2}t^2\right\} = e^{2Xs - s^2} = \sum_{k=0}^{\infty} H_k(X)\frac{s^k}{k!}.$$

Then the expression of $g'(x)$ can be rewritten as

$$\begin{aligned} g'(x) &= \sum_{n=0}^{M} a_n \frac{\mathrm{d}^n}{\mathrm{d}t^n}\left(\sum_{k=0}^{\infty} H_k(X)\frac{s^k}{k!}\right)\Big|_{s=0} \\ &= \sum_{n=0}^{M} a_n H_n(X)\left(i\sqrt{\frac{1-\rho^4}{2}}\right)^n. \end{aligned}$$

Therefore, we have a closed-form expression for $g'(x)$, which is also an M^{th}-order polynomial:

$$\begin{aligned} g'(x) &= \sum_{n=0}^{M} a_n H_n\left(\frac{\rho^2}{i\sqrt{2(1-\rho^4)}}x\right)\left(i\sqrt{\frac{1-\rho^4}{2}}\right)^n \\ &= \sum_{n=0}^{M} a_n \sum_{k=0}^{[n/2]} \frac{n!}{k!(n-2k)!}\left(\frac{1-\rho^4}{2}\right)^k (\rho^2 x)^{n-2k}. \end{aligned}$$

It is straight forward to derive that

$$\int_{-\infty}^{\infty} g^2(x) f_X(x)\mathrm{d}x = \sum_{n=0}^{M} n!\left(\sum_{k=0}^{[M-n/2]} a_{n+2k}\frac{(n+2k)!}{2^k n! k!}\right)^2$$

and

$$\begin{aligned} &\int_{-\infty}^{\infty} g'^2(x) f_X(x)\mathrm{d}x \\ &\quad = \sum_{n=0}^{M} (\rho^4)^n n!\left(\sum_{k=0}^{[M-n/2]} a_{n+2k}\frac{(n+2k)!}{2^k n! k!}\right)^2. \end{aligned}$$

Moreover, since g is of zero mean,

$$\sum_{k=0}^{[M/2]} a_{2k}\frac{(2k)!}{2^k k!} = 0.$$

Therefore,

$$\frac{\int_{-\infty}^{\infty} g'^2(x) f_X(x)\mathrm{d}x}{\int_{-\infty}^{\infty} g^2(x) f_X(x)\mathrm{d}x} = \frac{\sum_{n=1}^{M} (\rho^4)^n n! \left(\sum_{k=0}^{[M-n/2]} a_{n+2k} \frac{(n+2k)!}{2^k n! k!} \right)^2}{\sum_{n=1}^{M} n! \left(\sum_{k=0}^{[M-n/2]} a_{n+2k} \frac{(n+2k)!}{2^k n! k!} \right)^2} \leq \frac{\sum_{n=1}^{M} \rho^4 n! \left(\sum_{k=0}^{[M-n/2]} a_{n+2k} \frac{(n+2k)!}{2^k n! k!} \right)^2}{\sum_{n=1}^{M} n! \left(\sum_{k=0}^{[M-n/2]} a_{n+2k} \frac{(n+2k)!}{2^k n! k!} \right)^2} = \rho^4.$$

If the hidden rule $\phi(x_1, x_2)$ is restricted to a bivariate polynomial with finite order M and zero mean, and $g_1(x_1)$ and $g_2(x_2)$ are both initialized as 0, then after each iteration, $g_1(x_1)$ and $g_2(x_2)$ will remain in the space of zero-mean polynomials of finite order M. If we define the distance between two polynomials $g_1(x)$ and $g_2(x)$ as $\mathbb{E}[(g_1 - g_2)^2]$, then, according to the above lemma, the map T that converts $g(x)$ to $g'(x)$ is a contractive map.

APPENDIX C
DERIVATION OF THE GRADIENT (34)

Given the expression for weighting coefficients

$$\hat{\boldsymbol{\beta}} = (\mathbf{F}^T\mathbf{F})^{-1}\mathbf{F}^T\mathbf{y} \tag{51}$$

and the objective function

$$\eta = \mathbf{y}^T\mathbf{F}(\mathbf{F}^T\mathbf{F})^{-1}\mathbf{F}^T\mathbf{y}, \tag{52}$$

we have

$$\begin{aligned}
\frac{\partial \eta}{\partial \mathbf{F}} &= \frac{\partial}{\partial \mathbf{F}} \mathbf{y}^T\mathbf{F}(\mathbf{F}^T\mathbf{F})^{-1}\mathbf{F}^T\mathbf{y} \\
&= \frac{\partial}{\partial \mathbf{F}} \mathrm{Tr}(\mathbf{y}^T\mathbf{F}(\mathbf{F}^T\mathbf{F})^{-1}\mathbf{F}^T\mathbf{y}) \\
&= \frac{\partial}{\partial \mathbf{F}} \mathrm{Tr}(\mathbf{F}(\mathbf{F}^T\mathbf{F})^{-1}\mathbf{F}^T\mathbf{y}\mathbf{y}^T) \\
&= -2\mathbf{F}(\mathbf{F}^T\mathbf{F})^{-1}\mathbf{F}^T\mathbf{y}((\mathbf{F}^T\mathbf{F})^{-1}\mathbf{F}^T\mathbf{y})^T \\
&\quad + 2\mathbf{y}\mathbf{y}^T\mathbf{F}(\mathbf{F}^T\mathbf{F})^{-1} \\
&= 2(\mathbf{y} - \mathbf{F}(\mathbf{F}^T\mathbf{F})^{-1}\mathbf{F}^T\mathbf{y})((\mathbf{F}^T\mathbf{F})^{-1}\mathbf{F}^T\mathbf{y})^T \\
&= 2(\mathbf{y} - \mathbf{F}\hat{\boldsymbol{\beta}})\hat{\boldsymbol{\beta}}^T.
\end{aligned}$$

REFERENCES

[1] J. Basak and R. Kothari, *A Classification Paradigm for Distributed Vertically Partitioned Data*. Cambridge, MA: MIT Press, 2004, vol. 16, pp. 1525–1544.

[2] P. Buhlmann and B. Yu, "Boosting with the l2-loss: Regression and classification," *J. Amer. Statist. Assoc.*, vol. 98, pp. 324–339, 2003.

[3] P. Chan and S. J. Stolfo, "Experiments in multistrategy learning by meta-learning," in *Proc. 2nd Int. Conf. Information Knowledge Management*, Washington, DC, 1993, pp. 314–323.

[4] P. Chan and S. J. Stolfo, "Toward parallel and distributed learning by meta-learning," in *Working Notes of the AAAI Workshop, Knowledge Discovery in Databases*, Washington, DC, 1993, pp. 227–240.

[5] E. Dellis, B. Seeger, and A. Vlachou, "Nearest neighbor search on vertically partitioned high-dimensional data," in *Proc. 7th Int. Conf. Data Warehousing and Knowledge Discovery*, Copenhagen, Denmark, Aug. 2005, pp. 243–253.

[6] W. Fang and B. Yang, "Privacy preserving decision tree learning over vertically partitioned data," in *Proc. 2008 Int. Conf. Computer Science and Software Engineering*, Washington, DC, 2008, pp. 1049–1052.

[7] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Ann. Statist.*, vol. 29, pp. 1189–1232, 2000.

[8] J. H. Friedman, E. Grosse, and W. Stuetzle, "Multidimensional additive spline approximation," *SIAM J. Scientif. Statist. Comput.*, vol. 4, no. 2, pp. 291–301, 1983.

[9] D. E. Hershberger and H. Kargupta, "Distributed multivariate regression using wavelet-based collective data mining," *J. Parallel Distrib. Comput.*, vol. 61, no. 3, pp. 372–400, 2001.

[10] G. Jagannathan and R. N. Wright, "Privacy-preserving distributed k-means clustering over arbitrarily partitioned data," in *Proc. 11th ACM SIGKDD Int. Conf. Knowledge Discovery Data Mining*, New York, 2005, pp. 593–599.

[11] H. Kargupta, E. Johnson, E. R. Sanseverino, B. H. Park, L. D. Silvestre, and D. Hershberger, "Collective data mining from distributed vertically partitioned feature space," in *Proc. Workshop on Distributed Data Mining*, New York, 1998, pp. 70–91.

[12] H. Kargupta, B. Park, D. Hershberger, and E. Johnson, "Collective data mining: A new perspective toward distributed data mining," in *Advances in Distributed and Parallel Knowledge Discovery*. Cambridge, MA: AAAI/MIT Press, 1999, pp. 133–184.

[13] O. L. Mangasarian, E. W. Wild, and G. M. Fung, "Privacy-preserving classification of vertically partitioned data via random kernels," *ACM Trans. Knowl. Discov. Data*, vol. 2, no. 3, pp. 1–16, 2008.

[14] S. McConnell and D. B. Skillicorn, "Building predictors from vertically distributed data," in *Proc. 2004 Conf. Centre for Advanced Studies on Collaborative Research*, Markham, ON, Canada, 2004, pp. 150–162.

[15] S. Merugu and J. Ghosh, "A distributed learning framework for heterogeneous data sources," in *Proc. 11th ACM SIGKDD Int. Conf. Knowledge Discovery Data Mining*, Chicago, IL, Aug. 2005, pp. 208–217.

[16] J. B. Predd, "Topics in distributed inference," Ph.D. dissertation, Elect. Eng. Dept., Princeton Univ., Princeton, NJ, 2006.

[17] J. B. Predd, S. B. Kulkarni, and H. V. Poor, "Distributed learning in wireless sensor networks," *IEEE Signal Process. Mag.*, vol. 23, no. 4, pp. 56–69, Jul. 2006.

[18] G. Ridgeway, D. Madigan, and T. Richardson, "Boosting methodology for regression problems," in *Proc. 7th Int. Workshop Artificial Intelligence Statistics*, Fort Lauderdale, FL, 1999, pp. 152–161.

[19] D. Shutin, H. Zheng, B. H. Fleury, S. R. Kulkarni, and H. V. Poor, "Space-alternating attribute-distributed sparse learning," in *Proc. 2nd Int. Workshop Cognitive Information Processing*, Elba, Italy, Jun. 2010.

[20] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. Roy. Statist. Soc., Series B*, vol. 58, pp. 267–288, 1994.

[21] J. Vaidya and C. Clifton, "Privacy-preserving k-means clustering over vertically partitioned data," in *Proc. 9th Int. Conf. Knowledge Discovery Data Mining*, New York, 2003, pp. 206–215.

[22] J. Vaidya, C. Clifton, M. Kantarcioglu, and A. S. Patterson, "Privacy-preserving decision trees over vertically partitioned data," *ACM Trans. Knowl. Discov. Data*, vol. 2, no. 3, pp. 1–27, 2008.

[23] J. Vaidya, H. Yu, and X. Jiang, "Privacy-preserving SVM classification," *Knowl. Inf. Syst.*, vol. 14, no. 2, pp. 161–178, 2008.

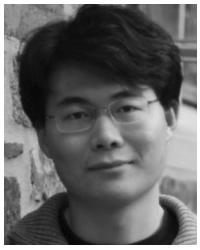
[24] Z. Yang and R. N. Wright, "Improved privacy-preserving Bayesian network parameter learning on vertically partitioned data," in *Proc. 21st Int. Conf. Data Engineering Workshops (ICDEW)*, Washington, DC, 2005, pp. 1196–1196.

[25] Z. Yang and R. N. Wright, "Privacy-preserving computation of Bayesian networks on vertically partitioned data," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 9, pp. 1253–1264, 2006.

[26] H. Zheng, S. R. Kulkarni, and H. V. Poor, "Dimensionally distributed learning: Models and algorithm," in *Proc. 11th Int. Conf. Information Fusion*, Cologne, Germany, Jul. 2008, pp. 1–8.

[27] H. Zheng, S. R. Kulkarni, and H. V. Poor, "Cooperative training for attribute-distributed data: Trade-off between data transmission and performance," in *Proc. 12th Int. Conf. Information Fusion*, Seattle, WA, Jul. 2009, pp. 664–671.

[28] H. Zheng, S. R. Kulkarni, and H. V. Poor, "Agent selection for regression on attribute distributed data," in *Proc. 35th IEEE Int. Conf. Acoustics, Speech, Signal Processing*, Dallas, TX, Mar. 2010.



Haipeng Zheng received the B.Eng. degree from Tsinghua University, Beijing, China, in 2006 and the M.A. degree from Princeton University, Princeton, NJ, in 2008, both in electrical engineering. He is currently working towards the Ph.D. degree in electrical engineering at Princeton University.

His research interests include statistical learning/inference on distributed system, robust shape reconstruction, multiuser precoding, and intelligent transportation.

Sanjeev R. Kulkarni (M'91–SM'96–F'04) received the B.S. degree in mathematics and the B.S. in electrical engineering and the M.S. degree in mathematics from Clarkson University, Potsdam, NY, in 1983, 1984, and 1985, respectively; the M.S. degree in electrical engineering from Stanford University, Stanford, CA, in 1985; and the Ph.D. degree in electrical engineering from the Massachusetts Institute of Technology (MIT), Cambridge, in 1991.

From 1985 to 1991, he was a Member of the Technical Staff at the MIT Lincoln Laboratory, Lexington, MA, working on the modeling and processing of laser radar measurements. In spring 1986, he was a part-time faculty member at the University of Massachusetts, Boston. Since 1991, he has been with Princeton University, Princeton, NJ, where he is currently Professor of Electrical Engineering and an affiliated faculty member in the Department of Operations Research and Financial Engineering and the Department of Philosophy. He spent January 1996 as a Research Fellow at the Australian National University, 1998 with Susquehanna International Group, and summer 2001 with Flarion Technologies. His research interests include statistical pattern recognition, nonparametric estimation, learning and adaptive systems, information theory, wireless networks, and image/video processing.

Prof. Kulkarni received an ARO Young Investigator Award in 1992, an NSF Young Investigator Award in 1994, and several teaching awards at Princeton University. He has served as an Associate Editor for the IEEE TRANSACTIONS ON INFORMATION THEORY.

H. Vincent Poor (S'72–M'77–SM'82–F'87) received the Ph.D. degree in electrical engineering and computer science from Princeton University, Princeton, NJ, in 1977.

From 1977 until 1990, he was on the faculty of the University of Illinois at Urbana-Champaign. Since 1990, he has been on the faculty of Princeton University, where he is the Dean of Engineering and Applied Science, and the Michael Henry Strater University Professor of Electrical Engineering. His research interests are in the areas of stochastic analysis, statistical signal processing, and information theory, and their applications in wireless networks and related fields. Among his publications in these areas are the recent books *Quickest Detection* (Cambridge Univ. Press, 2009), coauthored with O. Hadjiliadis, and *Information Theoretic Security* (Now Publishers, 2009), coauthored with Y. Liang and S. Shamai.

Dr. Poor is a member of the National Academy of Engineering, a Fellow of the American Academy of Arts and Sciences, and an International Fellow of the Royal Academy of Engineering of the United Kingdom. He is also a Fellow of the Institute of Mathematical Statistics, the Optical Society of America, and other organizations. In 1990, he served as President of the IEEE Information Theory Society, and from 2004 to 2007 as the Editor-in-Chief of the IEEE TRANSACTIONS ON INFORMATION THEORY. He was the recipient of the 2005 IEEE Education Medal. Recent recognition of his work includes the 2009 Edwin Howard Armstrong Achievement Award of the IEEE Communications Society, the 2010 IET Ambrose Fleming Medal for Achievement in Communications, and the 2011 IEEE Eric E. Sumner Award.