Vertically Partitioned Data

Junhao Hua

2014/10/14

Horizontally and Vertically Partitioned Data

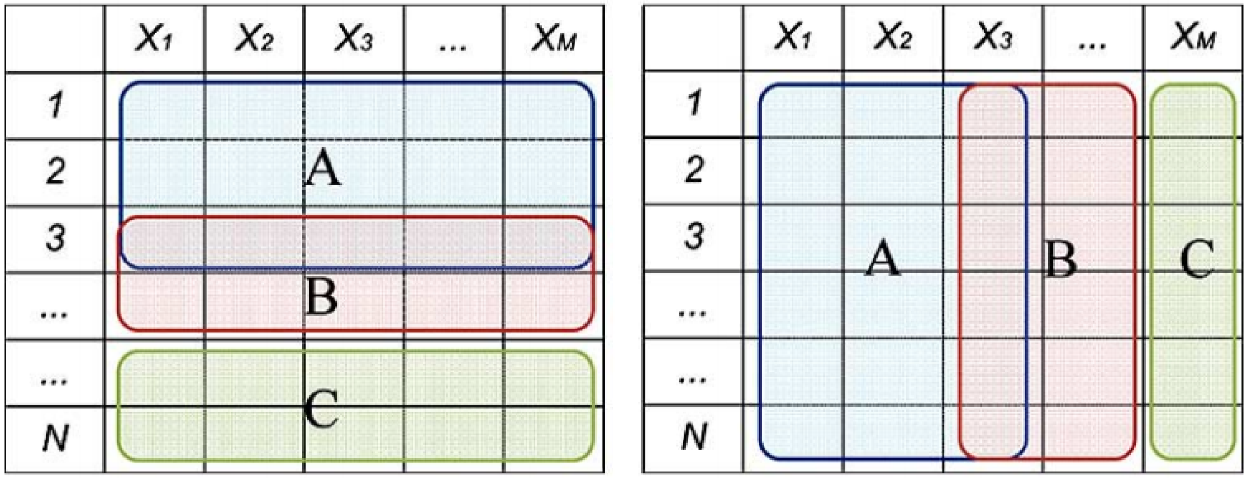


Fig. 1. Two basic scenarios for distributed data: instance-distributed (left) and attribute-distributed (right). In the instance-distributed scenario, each agent (A, B, and C) observes a subset of the instances, with complete information on all attributes; alternatively, in the attribute-distributed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

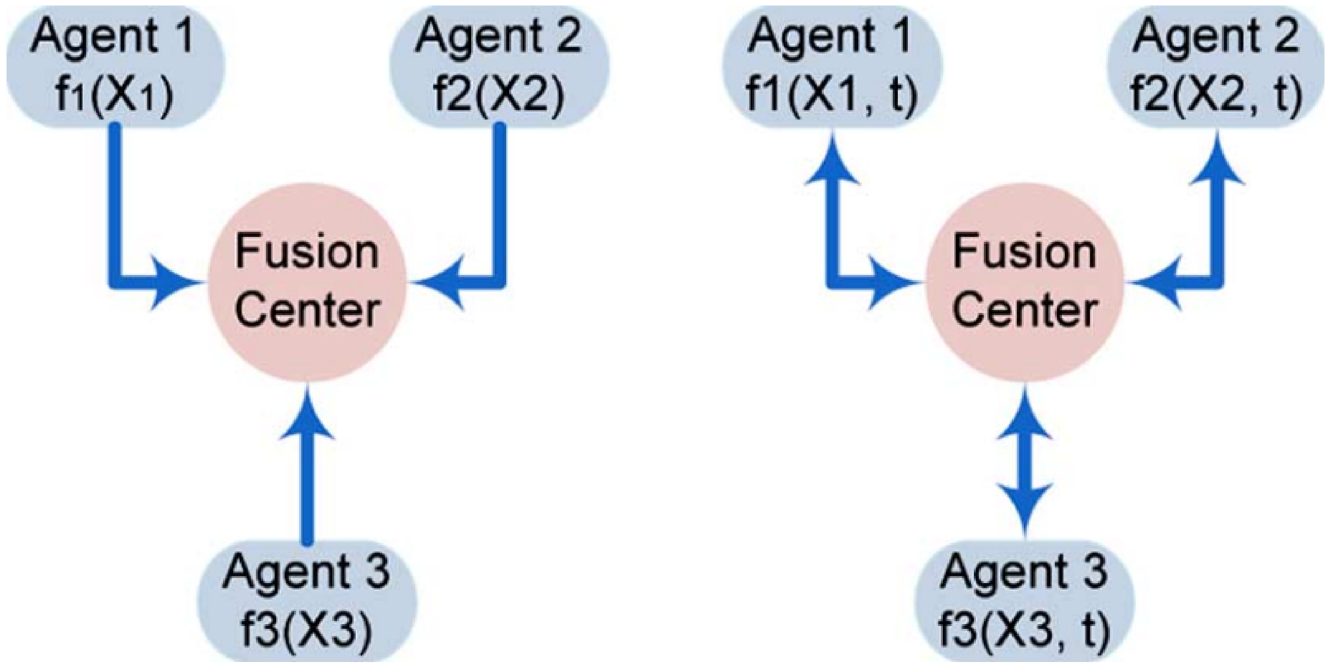


Fig. 2. Comparison between non-collaborative training (left) and collaborative training (right). For non-collaborative training, individual estimators (f_1, f_2 and f_3) are trained locally and fixed, and no agents receives information from other agents. Alternatively, for collaborative training, every 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.

Algorithms

1. Voting/Averaging

- Theory: Naïve Bayes assumption (conditional independent)

$$p(\mathrm{H} \mid x) = \frac{p(x \mid \mathrm{H}) \times p(\mathrm{H})}{p(x)}, \quad p(x \mid \mathrm{H}) = \prod_{j=1}^{m} p(x_j \mid \mathrm{H})$$

  - Even though the probabilities may be estimated wrongly, their ranking is preserved [1].
- Attribute ensembles algorithm [2]
  - (**local predictor**): each local site builds a predictor (decision trees, neural networks) that predicts the target attribute from the values of its local attribute.
  - (**global prediction**): Each local site applies its local predictor to each new records and transmits its prediction to the central site, where the predictions are averaged (for regression), or used as votes for the target class label, with the plurality winning.

[1] P. Domingos, M.J. Pazzani, On the optimality of the simple Bayesian classifier under zero-one loss, Mach. Learning 29 (2&3) (1997)
[2] Skillicorn, David B., and Sabine M. McConnell. "Distributed **prediction** from vertically partitioned data." *Journal of Parallel and Distributed computing* 68.1 (2008)

2. Meta-Learning

- Meta-data: data about data
- Integrate predictions of individual estimators by taking their predictions as a new training set (hierarchical training scheme)

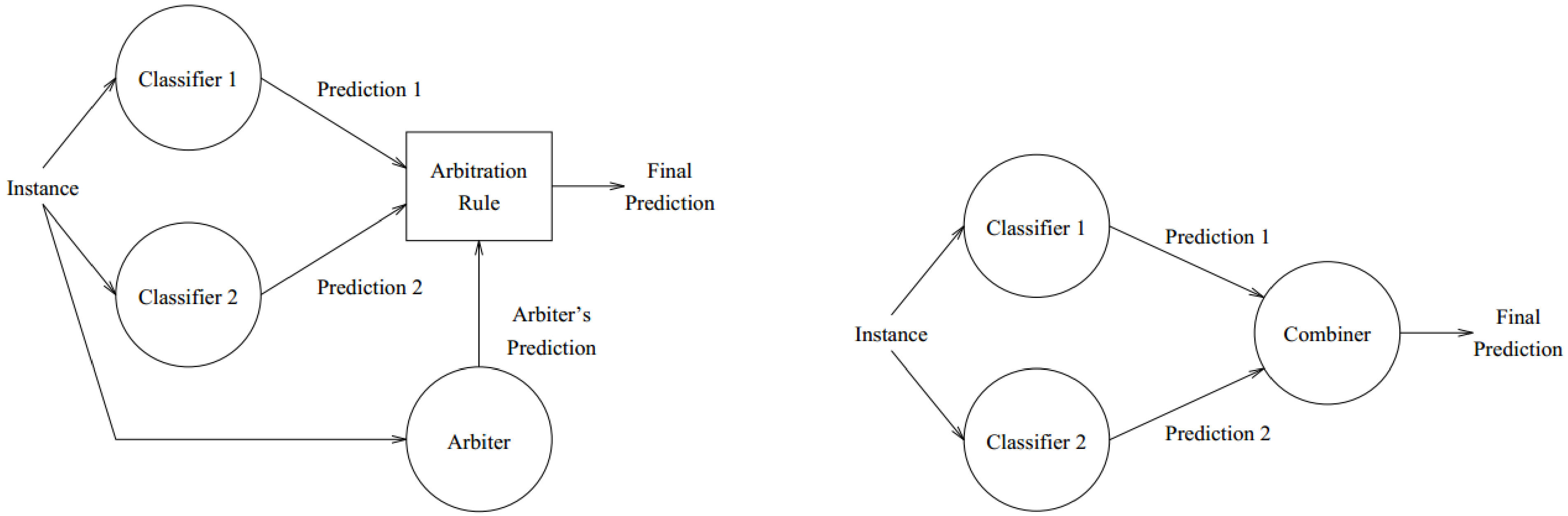
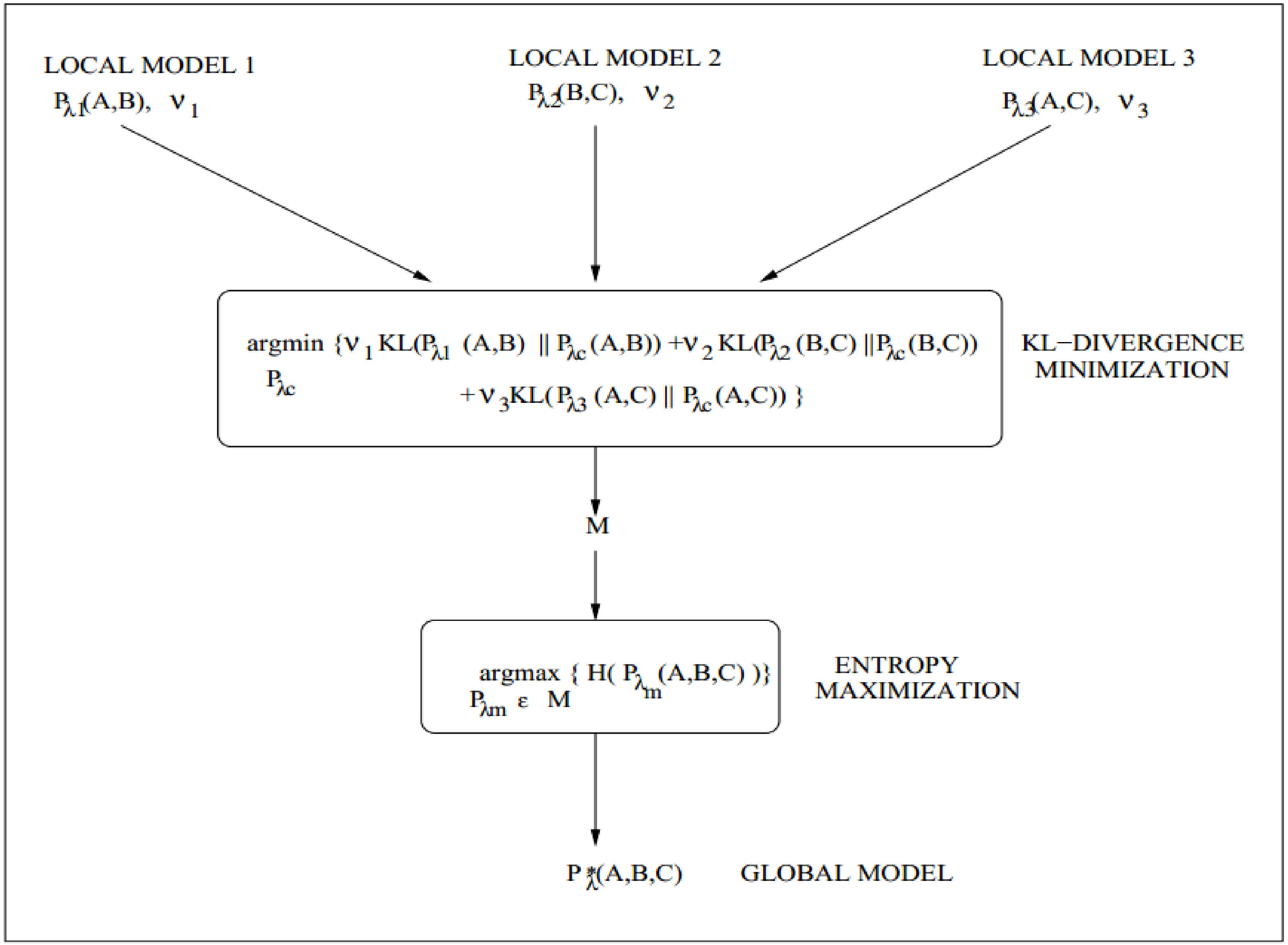


Figure 1: An arbiter and a combiner with two classifiers.

[3] P.K. Chan, S.J. Stolfo, Learning arbiter and combiner trees from partitioned data for scaling machine learning, Proceedings of the 1st International Conference on KDDM, AAAI Press, 1995.


LOCAL MODEL 1
$P_{\lambda 1}(A,B),\ \nu_1$
LOCAL MODEL 2
$P_{\lambda 2}(B,C),\ \nu_2$
LOCAL MODEL 3
$P_{\lambda 3}(A,C),\ \nu_3$
$\underset{P_{\lambda c}}{\operatorname{argmin}} \{\nu_1 KL(P_{\lambda 1}(A,B) \| P_{\lambda c}(A,B)) + \nu_2 KL(P_{\lambda 2}(B,C) \| P_{\lambda c}(B,C)) + \nu_3 KL(P_{\lambda 3}(A,C) \| P_{\lambda c}(A,C))\}$
KL−DIVERGENCE MINIMIZATION
M
$\underset{P_{\lambda m} \in M}{\operatorname{argmax}} \{H(P_{\lambda m}(A,B,C))\}$
ENTROPY MAXIMIZATION
$P_{\lambda}^{*}(A,B,C)$
GLOBAL MODEL

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.

3. Residual refitting

Attribute-distributed learning: models, limits, and algorithms

Zheng, Haipeng, Sanjeev R. Kulkarni, and H. Vincent Poor*, IEEE Transactions on Signal Processing* , 2011

Preliminaries

- Boosting: Boosting is based on the question posed by Michael Kearns in 1988: Can a set of **weak learners** create a single **strong learner**? A weak learner is defined to be a classifier which is only slightly correlated with the true classification (it can label examples better than random guessing). In contrast, a strong learner is a classifier that is arbitrarily well-correlated with the true classification.

The boosting algorithms can be seen as functional gradient descent techniques. The task is to estimate the function $F : \mathbb{R}^d \to \mathbb{R}$, minimizing an expected cost

$$\mathbb{E}[C(Y, F(X))], \qquad C(\cdot,\cdot) : \mathbb{R} \times \mathbb{R} \to \mathbb{R}^+ \tag{1}$$

based on data (Y_i, X_i) $(i = 1, \ldots, n)$.

d-dimensional predictor variable. The cost function $C(\cdot,\cdot)$ is assumed to be smooth and convex in the second argument, to ensure that the gradient method works well. The most prominent examples are

$$C(y, f) = \exp(yf) \text{ with } y \in \{-1, 1\}\text{: AdaBoost cost function,}$$

$$C(y, f) = \log_2(1 + \exp(-2yf)) \text{ with } y \in \{-1, 1\}\text{: LogitBoost cost function,} \tag{2}$$

$$C(y, f) = (y - f)^2/2 \text{ with } y \in \mathbb{R} \text{ or } \in \{-1, 1\}\text{: } L_2\text{Boost cost function.}$$

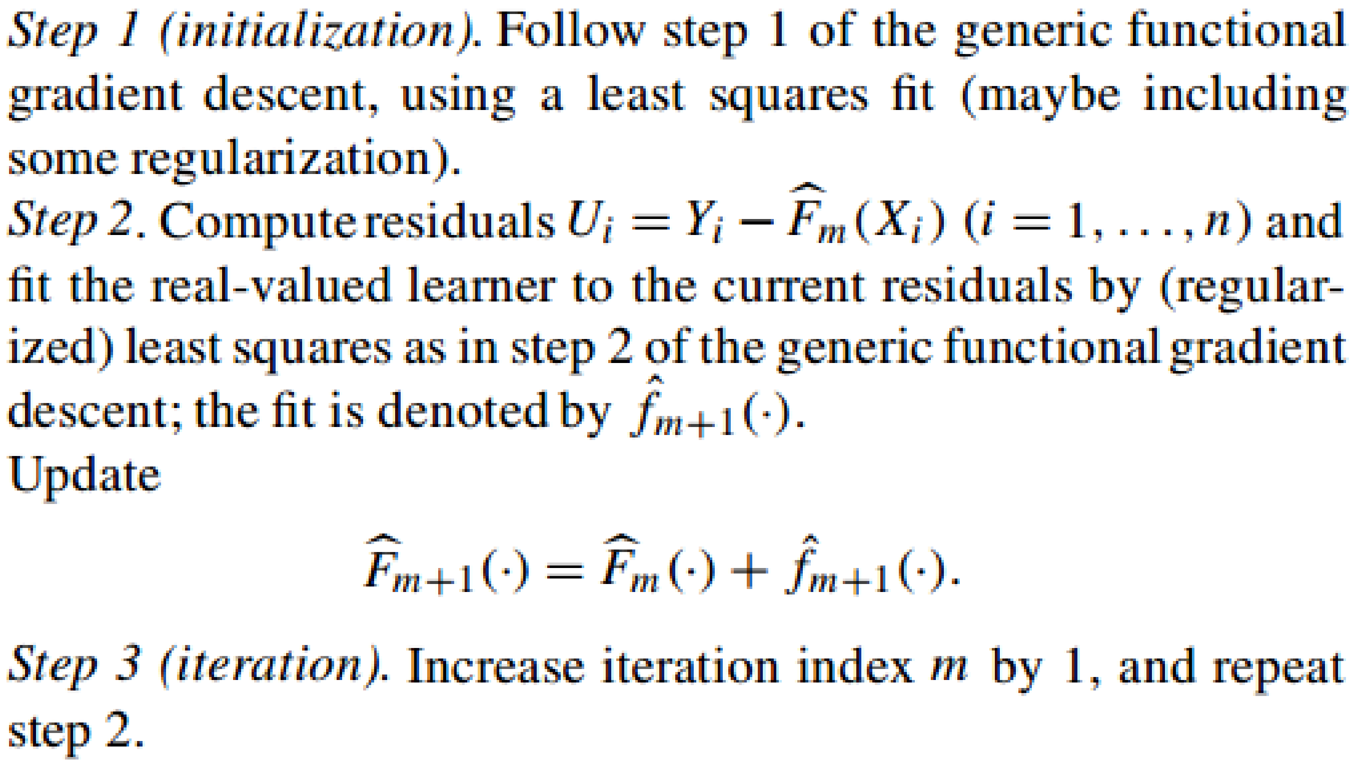
The population minimizers of (1) are then (Friedman et al. 2000)

$$F(x) = \frac{1}{2}\log\left(\frac{\mathbb{P}[Y = 1|X = x]}{\mathbb{P}[Y = -1|X = x]}\right) \quad \text{for AdaBoost and LogitBoost cost,} \tag{3}$$

$$\boxed{F(x) = \mathbb{E}[Y|X = x] \quad \text{for } L_2\text{Boost cost.}}$$

Preliminaries(cont.)

- L2 Boosting: L2 boosting is nothing else than repeated least squares fitting of residuals (Friedman 2001).



Step 1 (initialization). Follow step 1 of the generic functional gradient descent, using a least squares fit (maybe including some regularization).

Step 2. Compute residuals $U_i = Y_i - \widehat{F}_m(X_i)$ $(i = 1, \ldots, n)$ and fit the real-valued learner to the current residuals by (regularized) least squares as in step 2 of the generic functional gradient descent; the fit is denoted by $\hat{f}_{m+1}(\cdot)$.

Update

$$\widehat{F}_{m+1}(\cdot) = \widehat{F}_m(\cdot) + \hat{f}_{m+1}(\cdot).$$

Step 3 (iteration). Increase iteration index m by 1, and repeat step 2.

Preliminaries(cont.)

- The additive models

$$Y = \alpha + \sum_{j=1}^{p} f_j(X_j) + \varepsilon,$$

  - f: smooth (nonparametric) functions

Algorithm 9.1 *The Backfitting Algorithm for Additive Models.*

```
1. Initialize: $\hat{\alpha} = \frac{1}{N}\sum_1^N y_i$, $\hat{f}_j \equiv 0, \forall i, j$.

2. Cycle: $j = 1, 2, \ldots, p, \ldots, 1, 2, \ldots, p, \ldots,$

       $\hat{f}_j \leftarrow \mathcal{S}_j\left[\{y_i - \hat{\alpha} - \sum_{k \neq j} \hat{f}_k(x_{ik})\}_1^N\right],$
       $\hat{f}_j \leftarrow \hat{f}_j - \frac{1}{N}\sum_{i=1}^{N} \hat{f}_j(x_{ij}).$

   until the functions $\hat{f}_j$ change less than a prespecified threshold.
```

Problem statement

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

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

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

and for each agent j, the set is reduced to

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

ally, we need to solve the following problem:

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

where

$$\rho \in \mathcal{J} = \{ f : \mathbb{R}^D \to \mathbb{R} \}. \qquad (10)$$

Problem statement (cont.)

If we assume that the ensemble estimator is of additive form

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

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

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

of minimizing the mean square training error.

Algorithm 1:Iterative Conditional Expectation Projection

- Minimizers

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

and

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

- Or explicitly

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

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

- Solve the finite, noisy data case (Residual refitting)

Algorithm 1: Prototype Residual-Refitting Algorithm

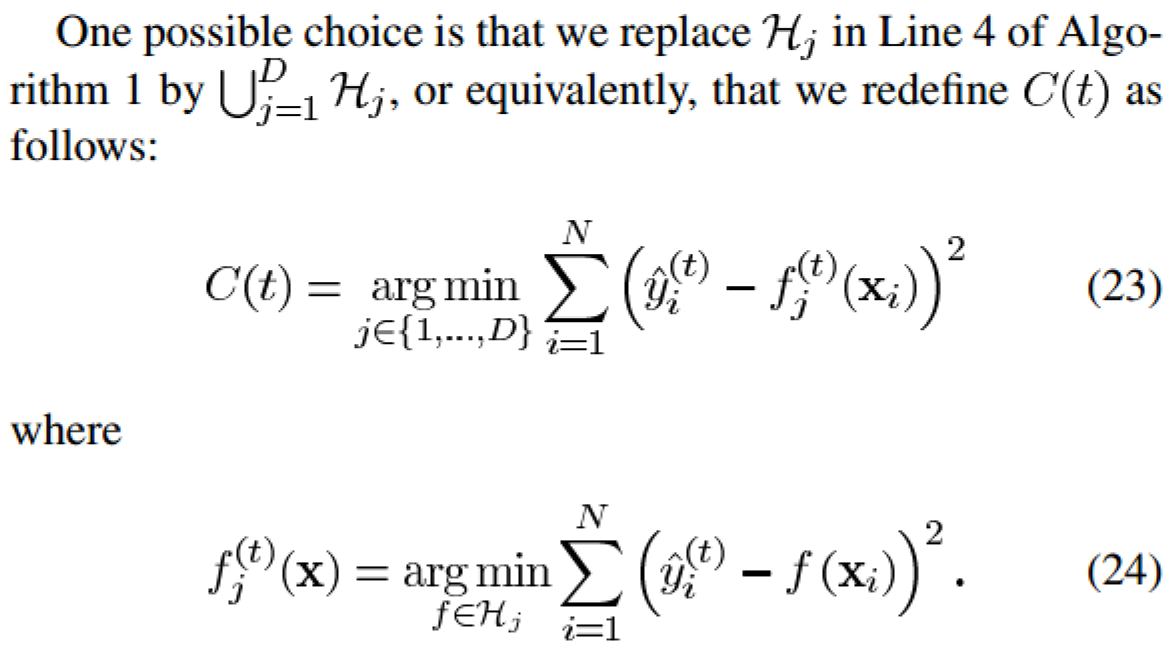
```
$F_0(\mathbf{x}) \leftarrow \frac{1}{N}\sum_{i=1}^{N} y_i$
for $t = 1, \ldots, T$ do
    $\hat{y}_i^{(t)} \leftarrow y_i - F_{t-1}(\mathbf{x}_i),\ \forall i \in \mathcal{I}$
    $f^{(t)}(\mathbf{x}) \leftarrow \arg\min_{f \in \mathcal{H}_{C(t)}} \sum_{i=1}^{N} (\hat{y}_i^{(t)} - f(\mathbf{x}_i))^2$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

Algorithm 2: L_2 Boosting Algorithm

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

Algorithm 2: greedy algorithm

- the fusion center sends the residual to all the agents, each of which finds a local optimal estimator based on its own limited set of functions , and then the fusion center chooses the agent that generates the estimator with the smallest training error to project the current residual.



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:

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

where

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

Algorithm 3: Parallel Gradient Descent

- New optimization problem

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

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.

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

- Solution:
  - Alternating update weights (at fusion center) and functions (at each agent)

Two-stage optimization

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

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

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

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

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

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

where

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

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

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

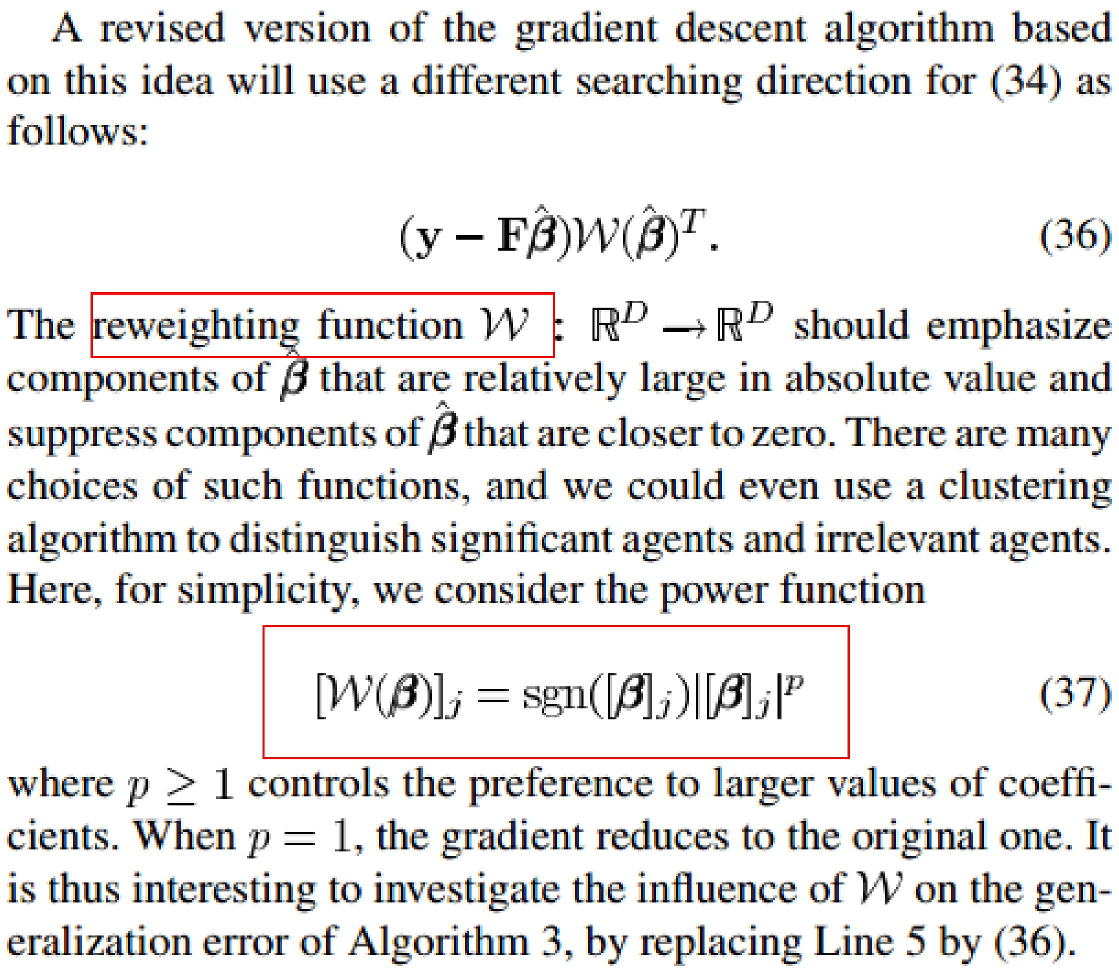
Note that the gradient for $\mathbf{F}$ is simply the current training residual $\mathbf{R} = \mathbf{y} - \mathbf{F}\hat{\boldsymbol{\beta}}$, $\boxed{\text{reweighted}}$ by $\hat{\beta}_j$ for the jth column.

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
                                                        fusion center
5      $\Delta\mathbf{F} \leftarrow (\mathbf{y} - \mathbf{F}\hat{\boldsymbol{\beta}})\hat{\boldsymbol{\beta}}$
6      $\delta \leftarrow \text{BackSearch}(\alpha,\ \beta,\ \mathbf{F},\ \Delta\mathbf{F})$
7      $\hat{\mathbf{F}} \leftarrow \mathbf{F} + \delta_t \Delta\mathbf{F}$
8      $f_j \leftarrow \arg\min_{f \in \mathcal{H}_j} \sum_{i=1}^{N} ([\mathbf{F}]_{ij} - f(\mathbf{x}_i))^2$      each agent
9      $[\mathbf{F}]_{ij} \leftarrow f_j(\mathbf{x}_i)$
10     $\hat{\boldsymbol{\beta}} = (\mathbf{F}^T\mathbf{F})^{-1}\mathbf{F}\mathbf{y}$
11 end
```

the parallel approach, by using individual residual information holistically, performs the best in terms of generalization.

Algorithm 4: Beyond Gradient Descent



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 $\mathcal{W} : \mathbb{R}^D \to \mathbb{R}^D$ 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

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

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).

emphasize more promising agents further improves the capability of eliminating irrelevant attributes

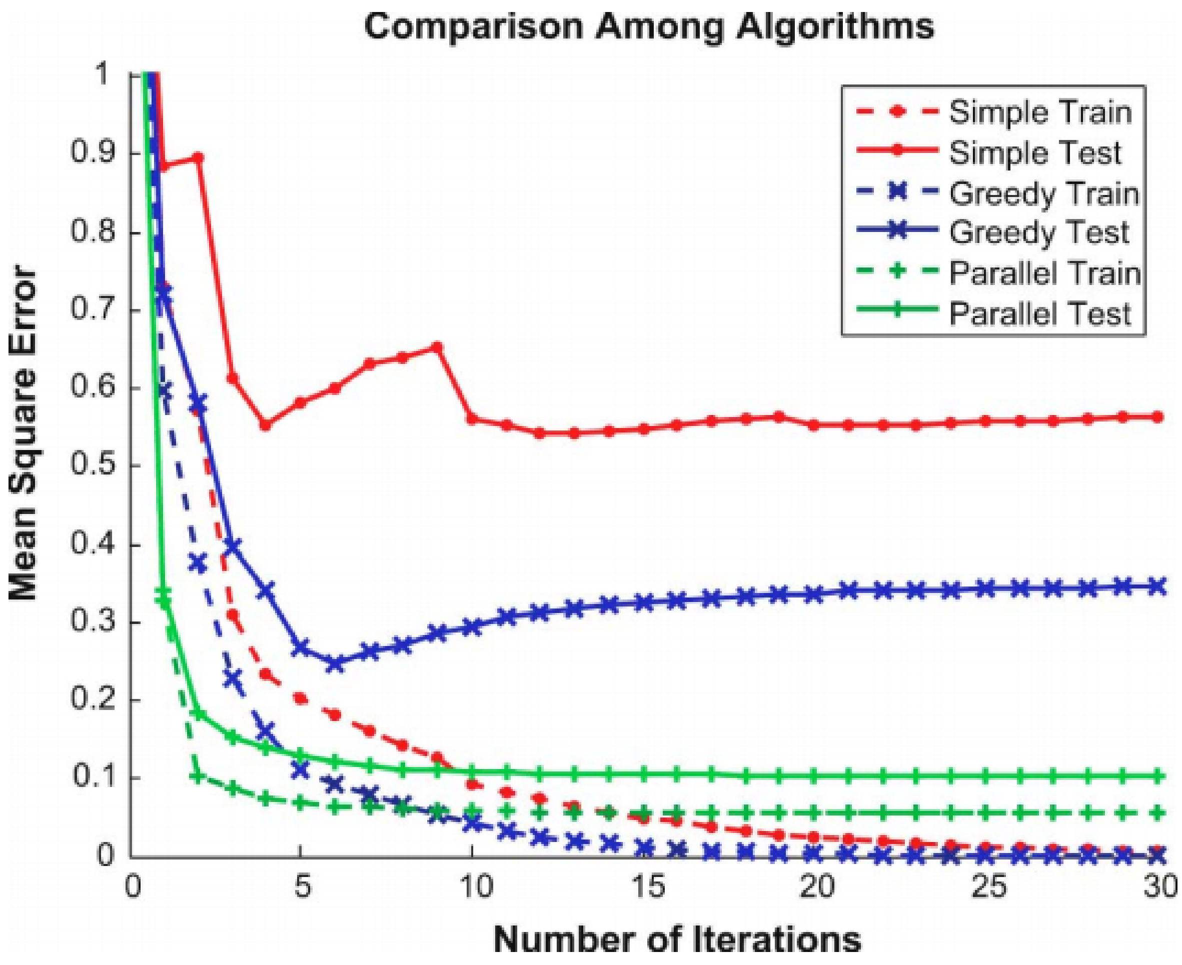
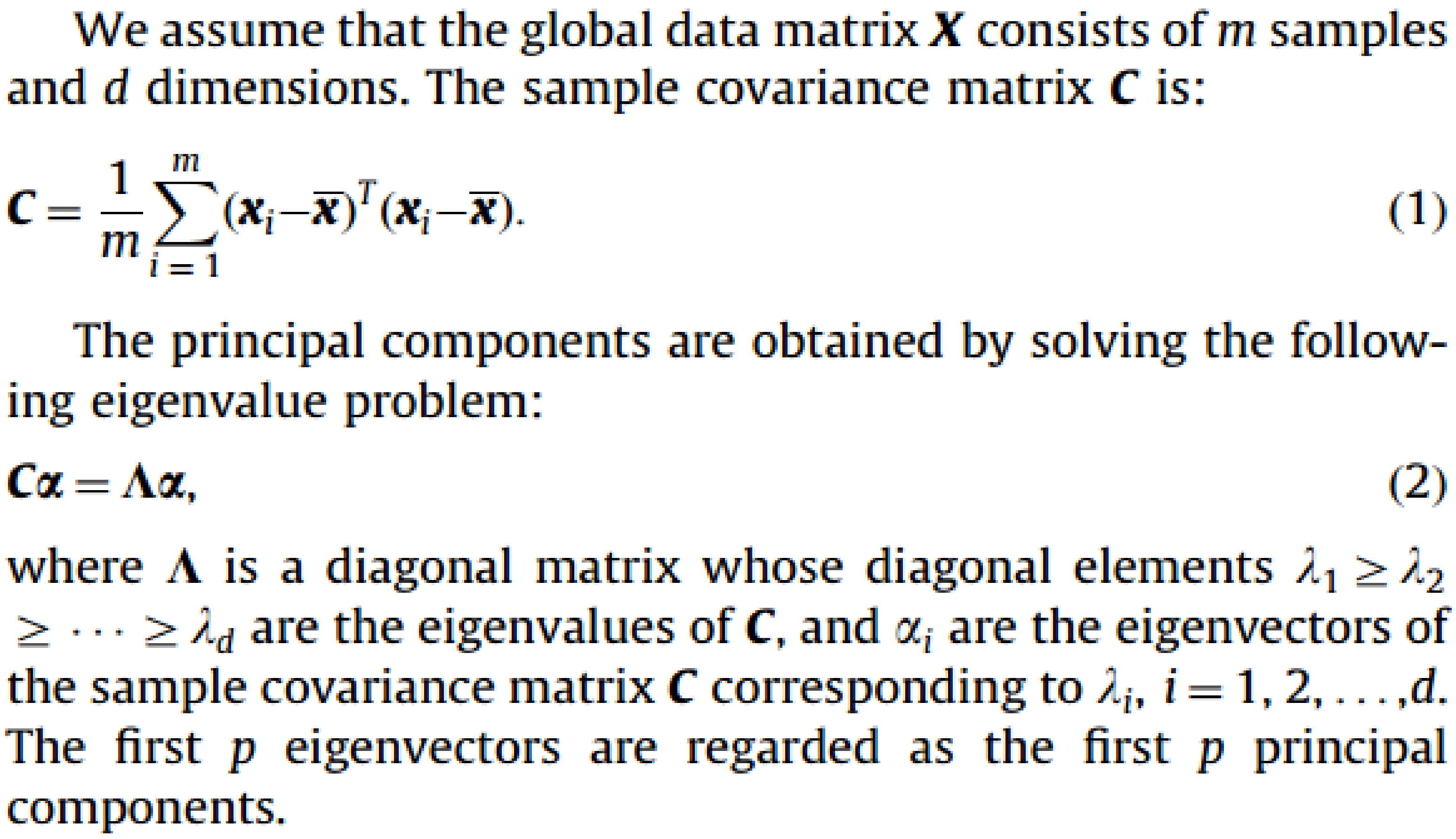


Fig. 4. Comparison between the convergence of the parallel algorithm, the greedy algorithm and Simple Iterative Projection for the Friedman-1 dataset with 5 irrelevant attributes. The parallel algorithm is least susceptible to overtraining than the greedy algorithm and Simple Iterative Projection, and 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.

Distributed PCA on vertically partitioned data

- Guo, Yue-Fei, et al. "A covariance-free iterative algorithm for distributed principal component analysis on vertically partitioned data." *Pattern Recognition* 45.3 (2012): 1211-1219.

Problem statement



We assume that the global data matrix $\boldsymbol{X}$ consists of m samples and d dimensions. The sample covariance matrix $\boldsymbol{C}$ is:

$$\boldsymbol{C} = \frac{1}{m}\sum_{i=1}^{m}(\boldsymbol{x}_i - \overline{\boldsymbol{x}})^T(\boldsymbol{x}_i - \overline{\boldsymbol{x}}). \tag{1}$$

The principal components are obtained by solving the following eigenvalue problem:

$$\boldsymbol{C\alpha} = \boldsymbol{\Lambda\alpha}, \tag{2}$$

where $\boldsymbol{\Lambda}$ is a diagonal matrix whose diagonal elements $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$ are the eigenvalues of $\boldsymbol{C}$, and α_i are the eigenvectors of the sample covariance matrix $\boldsymbol{C}$ corresponding to λ_i, $i = 1, 2, \ldots, d$. The first p eigenvectors are regarded as the first p principal components.

Problem statement (cont.)

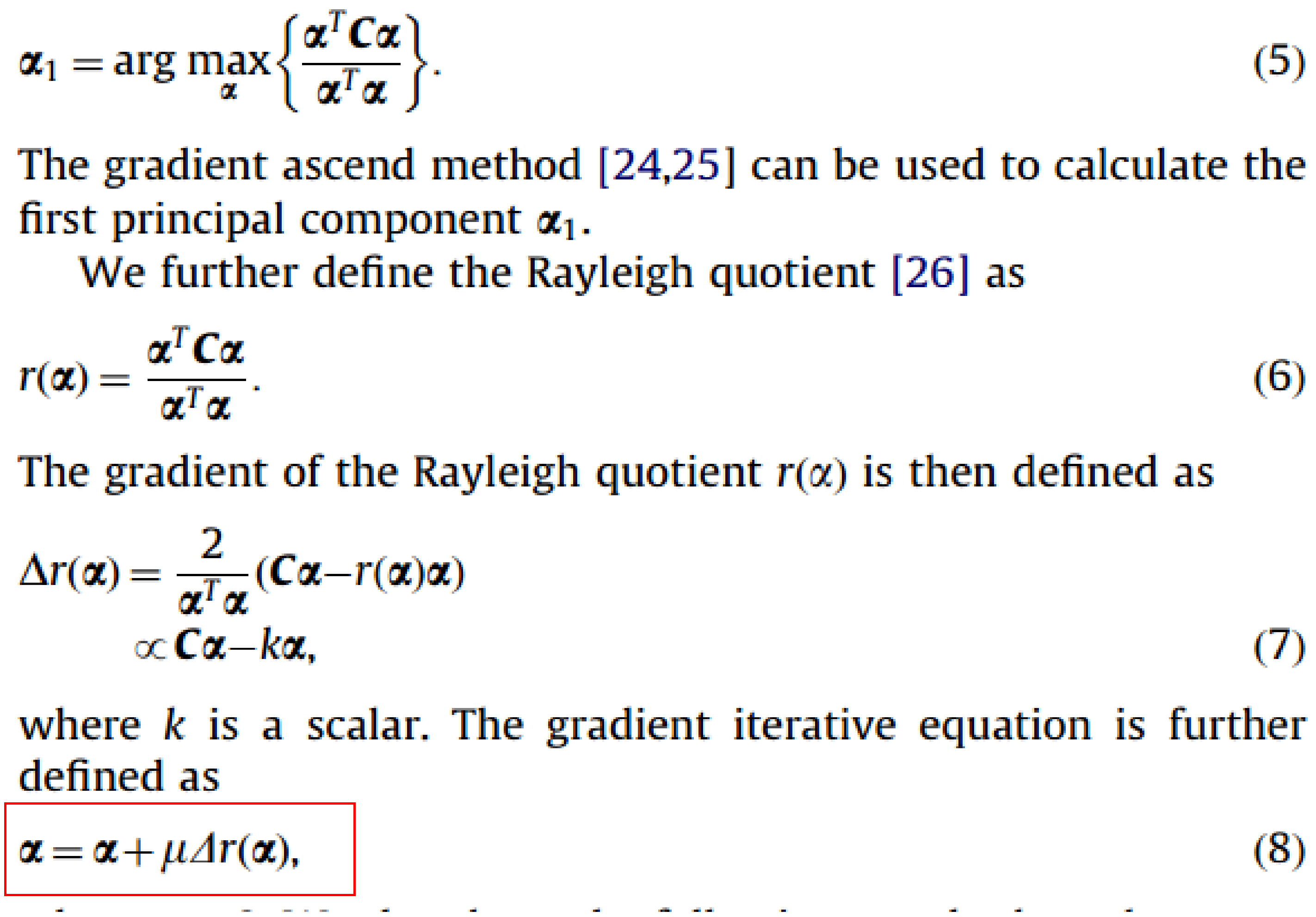
In the vertical data partition setting, each site contains all the data samples but each sample is represented by only a subset of the attributes. Let the data matrix $\boldsymbol{X}$ be partitioned into s subsets as $\boldsymbol{X} = (\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_s)$, $\boldsymbol{X}_i$ is $m \times d_i$ sub-matrices of $\boldsymbol{X}$, $i = 1, \ldots, s$, with $d = \sum_{i=1}^{s} d_i$. The global covariance matrix is then defined as

$$\begin{aligned}\mathbf{C} &= \frac{1}{m}(\boldsymbol{X} - \mathbf{1} \times \overline{\boldsymbol{x}})^T (\boldsymbol{X} - \mathbf{1} \times \overline{\boldsymbol{x}}) \\ &= \frac{1}{m}(\boldsymbol{X}_1 - \mathbf{1} \times \overline{\boldsymbol{x}}_1, \ldots, \boldsymbol{X}_s - \mathbf{1} \times \overline{\boldsymbol{x}}_s)^T (\boldsymbol{X}_1 - \mathbf{1} \times \overline{\boldsymbol{x}}_1, \ldots, \boldsymbol{X}_s - \mathbf{1} \times \overline{\boldsymbol{x}}_s) \\ &= \frac{1}{m}\begin{pmatrix} \boldsymbol{Y}_1^T \boldsymbol{Y}_1 & \cdots & \boldsymbol{Y}_1^T \boldsymbol{Y}_s \\ \cdots & & \\ \boldsymbol{Y}_s^T \boldsymbol{Y}_1 & \cdots & \boldsymbol{Y}_s^T \boldsymbol{Y}_s \end{pmatrix},\end{aligned} \tag{3}$$

where $\boldsymbol{Y}_i = \boldsymbol{X}_i - \mathbf{1} \times \overline{\boldsymbol{x}}_i$, $\overline{\boldsymbol{x}}_i$ is the mean of the data subset i with d_i dimensions of the sample attributes, $1 \le i \le s$, and $\mathbf{1}$ is a unit column vector of size m.

Solution

- Step 1: Find the eigenvector corresponding to the maximum eigenvalue using gradient ascend method.



$$\boldsymbol{\alpha}_1 = \arg\max_{\boldsymbol{\alpha}} \left\{ \frac{\boldsymbol{\alpha}^T \mathbf{C} \boldsymbol{\alpha}}{\boldsymbol{\alpha}^T \boldsymbol{\alpha}} \right\}. \tag{5}$$

The gradient ascend method [24,25] can be used to calculate the first principal component $\boldsymbol{\alpha}_1$.

We further define the Rayleigh quotient [26] as

$$r(\boldsymbol{\alpha}) = \frac{\boldsymbol{\alpha}^T \mathbf{C} \boldsymbol{\alpha}}{\boldsymbol{\alpha}^T \boldsymbol{\alpha}}. \tag{6}$$

The gradient of the Rayleigh quotient $r(\alpha)$ is then defined as

$$\begin{aligned} \Delta r(\boldsymbol{\alpha}) &= \frac{2}{\boldsymbol{\alpha}^T \boldsymbol{\alpha}} (\mathbf{C}\boldsymbol{\alpha} - r(\boldsymbol{\alpha})\boldsymbol{\alpha}) \\ &\propto \mathbf{C}\boldsymbol{\alpha} - k\boldsymbol{\alpha}, \end{aligned} \tag{7}$$

where k is a scalar. The gradient iterative equation is further defined as

$$\boxed{\boldsymbol{\alpha} = \boldsymbol{\alpha} + \mu \Delta r(\boldsymbol{\alpha}),} \tag{8}$$

- Distributed calculating the gradient ascent

We denote the initial vector

$$\boldsymbol{\alpha}=\begin{pmatrix}\varphi_1\\ \varphi_2\\ \vdots\\ \varphi_s\end{pmatrix},$$

where φ_i is the d_i-dimensional column vector, $i=1,\ldots,s$. The iterative equation (10) can be represented as

$$\begin{pmatrix}\varphi_1\\ \varphi_2\\ \vdots\\ \varphi_s\end{pmatrix}=\begin{pmatrix}\varphi_1\\ \varphi_2\\ \vdots\\ \varphi_s\end{pmatrix}+\mu\begin{pmatrix}\mathbf{Y}_1^T\mathbf{Y}_1 & \mathbf{Y}_1^T\mathbf{Y}_2 & \cdots & \mathbf{Y}_1^T\mathbf{Y}_s\\ \mathbf{Y}_2^T\mathbf{Y}_1 & \mathbf{Y}_2^T\mathbf{Y}_2 & \cdots & \mathbf{Y}_2^T\mathbf{Y}_s\\ & & \cdots & \\ \mathbf{Y}_s^T\mathbf{Y}_1 & \mathbf{Y}_s^T\mathbf{Y}_2 & \cdots & \mathbf{Y}_s^T\mathbf{Y}_s\end{pmatrix}\begin{pmatrix}\varphi_1\\ \varphi_2\\ \vdots\\ \varphi_s\end{pmatrix}. \tag{28}$$

This is equivalent to

$$\begin{cases}\varphi_1=\varphi_1+\mu\mathbf{Y}_1^T\sum_{i=1}^{s}\mathbf{Y}_i\varphi_i\\ \varphi_2=\varphi_2+\mu\mathbf{Y}_2^T\sum_{i=1}^{s}\mathbf{Y}_i\varphi_i\\ \vdots\\ \varphi_s=\varphi_s+\mu\mathbf{Y}_s^T\sum_{i=1}^{s}\mathbf{Y}_i\varphi_i\end{cases} \tag{29}$$

- Step 2: Find the eigenvector corresponding to the maximum eigenvalue in the orthogonal complement space of \alpha_1.
- Algorithm flow: see in paper.