DIFFUSION GRADIENT BOOSTING FOR NETWORKED LEARNING

Bicheng Ying and Ali H. Sayed

University of California, Los Angeles

ABSTRACT

Using duality arguments from optimization theory, this work develops an effective distributed gradient boosting strategy for inference and classification by networked clusters of learners. By sharing local dual variables with their immediate neighbors through a diffusion learning protocol, the clusters are able to match the performance of centralized boosting solutions even when the individual clusters only have access to partial information about the feature space.

Index Terms— Gradient boosting, distributed learning, diffusion strategy, AdaBoost.

1. INTRODUCTION AND MOTIVATION

In statistics and machine learning, ensemble learning is a formidable technique that is able to aggregate the recommendations of weak classifiers into a more powerful classification structure with enhanced predictive abilities [1–4]. One prominent example is the AdaBoost algorithm [3, 5, 6], which has achieved great prominence due to its efficient implementation structure, strong performance, and its ability to limit over-fitting [1]. It also satisfies a useful optimality property in that it can be derived from the minimization of exponential risk functions [7]. This connection between exponential risks and AdaBoost has motivated useful generalizations of boosting solutions using other choices for the risk function. Two of the main generalizations introduced in [8] and [9] are the Gradient Boosting Machine and the AnyBoost solution. These works helped solidify the connection between boosting techniques and gradient-descent methods from an optimization theory perspective.

In this article, we exploit this connection more broadly and consider *distributed* implementations. Specifically, we show how to develop *cooperative* boosting strategies by exploiting strong duality arguments from optimization theory [10, 11] and powerful diffusion strategies from distributed learning [12, 13]. We examine the important case in which the weak classifiers are not co-located but are dispersed, either geographically over space or by design through the intentional partitioning of the classifier set. We assume that the classifier set is partitioned into smaller groups, where the elements in each group may only have access to lower dimensional subspaces of the feature space. The groups are also networked by an outer topology – see Fig. 1 further ahead. The objective is to endow the dispersed groups of classifiers, through localized cooperation, with a distributed learning mechanism that ensures that the quality of their predictions is as close to what would result from a centralized solution with access to the entire feature space.

One earlier approach to distributed boosting is studied in [14–16]. It is based on learning from subsets of the training data and then combining the weak classifiers through an aggregation procedure. This formulation is different from the approach pursued in this work, which is fully decentralized and does not involve fusing information from across all classifiers. A second example is the Ivote procedure [17] and its distributed version DIvote [18,19]. These procedures, however, do not rely on boosting and their theoretical limits of performance have not been analyzed as closely as AdaBoost. While these various methods work well in some circumstances, they can still suffer from over-fitting or get trapped at local minima. In comparison, this work devises truly distributed boosting solutions with performance guarantees by relying on strong duality arguments [11, 20] and the theory of diffusion adaptation [12, 13], which allow the algorithms to disperse data and computation in a parallel and distributed manner.

This work was supported in part by NSF grants ECCS-1407712 and CCF-1524250, and DARPA project N66001–14–2–4029. Email: {ybc,sayed}@ucla.edu

2. GRADIENT BOOSTING ALGORITHM

In order to prepare for the derivation of the distributed strategy, we briefly review the well-known gradient boosting technique [8,9,21] in the context of supervised machine learning problems. Thus, consider a collection of N data pairs:

$$\mathcal{D} = \Big\{\{h_1, \gamma(1)\}, \{h_2, \gamma(2)\}, \cdots, \{h_N, \gamma(N)\}\Big\} \tag{1}$$

where $h_n \in \mathbb{R}^M$ are feature vectors and $\gamma(n)$ represent the class variable. In this article, we assume that there are two classes $\gamma(n)\in \{\pm 1\}$. A generic classifier, denoted by $c(h)$, is a transformation that maps a feature vector h into a class value, $\gamma(h)$. Assume we have a collection of L weak classifiers:

$$\mathcal{C} = \{c_1(h), c_2(h), \cdots, c_L(h)\}, \quad (L \text{ can be larger than } N) \tag{2}$$

We would like to select combination coefficients $\{\alpha(\ell)\}$ to construct a prediction for the class variable $\gamma(h)$ by combining the above learners into a more powerful classifier:

$$\widehat{\gamma}(h) \;\triangleq\; \sum_{\ell=1}^{L} \alpha(\ell) c_\ell(h) \tag{3}$$

The coefficients $\{\alpha(\ell)\}$ are determined by minimizing a surrogate risk of the following form

$$J_{\text{emp}}(\widehat{\gamma}) \;\triangleq\; \frac{1}{N}\sum_{n=1}^{N} Q\Big(\gamma(n), \widehat{\gamma}(n)\Big) \tag{4}$$

where the symbol $Q(\cdot)$ denotes a generic loss function, assumed convex and first-order differentiable. Some popular choices for $Q(\cdot)$ include the exponential loss, quadratic loss, hinge loss, and logistic loss [1,2,22]. Gradient boosting provides a solution technique by applying a greedy strategy to the minimization of (4) [1, 8, 21], where one coefficient $\alpha(\ell)$ and one classifier $c_\ell(h)$ are selected at a time. Specifically, assume that by the end of iteration $t-1$, we have already identified classifiers $\{c_1^o(h), \ldots, c_{t-1}^o(h)\}$ and weights $\{\alpha^o(1), \ldots, \alpha^o(t-1)\}$. Gradient boosting selects the next classifier, $c_t(h)$, and its associated weight, $\alpha(t)$, and enlarges the aggregate prediction from iteration $t-1$ as follows:

$$\widehat{\gamma}^{(t)}(h) = \sum_{s=1}^{t-1} \alpha^o(s) c_s^o(h) + \alpha(t) c_t(h) = \widehat{\gamma}^{(t-1)}(h) + \alpha(t) c_t(h)$$

In order to determine the optimal choices $\{c_t^o(h), \alpha^o(t)\}$, the algorithm evaluates the negative gradient of the empirical risk (4) and selects the classifier and its weight optimally: the index ℓ^o is selected through (7) and the coefficient $\alpha^o(t)$ through (8).

Gradient boosting algorithm [8]

Initialization: choose $\widehat{\gamma}^{(0)}(n),\ n = 1, 2, \ldots, N$
repeat for $t = 1, 2, \ldots, T$:

$$g_t(n) = -\left.\frac{\partial Q\big(\gamma(n), \widehat{\gamma}(n)\big)}{\partial \widehat{\gamma}(n)}\right|_{\widehat{\gamma}(n) = \widehat{\gamma}^{(t-1)}(n)} \tag{5}$$

$$\{\ell^o, \beta^o\} = \arg\min_{\{\ell, \beta\}} \sum_{n=1}^{N} \left(g_t(n) - \beta c_\ell(h_n)\right)^2 \tag{6}$$

$$c_t^o(h) = c_{\ell^o}(h) \tag{7}$$

$$\alpha^o(t) = \arg\min_{\alpha} \sum_{n=1}^{N} Q\left(\gamma(n),\, \widehat{\gamma}^{(t-1)}(h_n) + \alpha c_t^o(h_n)\right) \tag{8}$$

$$\widehat{\gamma}^{(t)}(h) = \widehat{\gamma}^{(t-1)}(h) + \alpha^o(t) c_t^o(h) \tag{9}$$

end

3. DIFFUSION GRADIENT BOOSTING

We now derive an effective decentralized strategy for boosting assuming distributed classifier groups and partial information at the groups. We first describe a formulation that involves the *centralized* fusion of predictions from a collection of dispersed learning groups. We subsequently apply duality arguments to show that this can be transformed into a distributed implementation that relies solely on local interactions among the groups.

3.1. Networked Groups and Partitioning Model

We consider a scenario in which the L classifiers are divided into K groupings. We index the groups by the letter k, with $k = 1, 2, \ldots, K$. We denote the classifiers that are available in group k by:

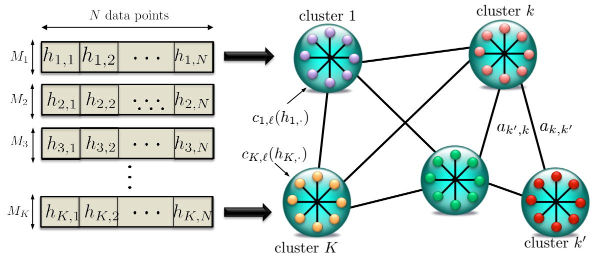
$$\mathcal{C}_k = \{c_{k,1}(h), c_{k,2}(h), \ldots, c_{k,L_k}(h)\} \tag{10}$$

Note that we are attaching a subscript k to the classifiers to indicate that these are the ones used by group k. We allow classifiers to be repeated across groups. We also assume that each group k may have access to only a *subset* of the feature space for its classification decisions. This situation is common. For example, weak classifiers are often chosen as shallow decision trees, or simply stumps [1, 23]. When a number of these weak classifiers is present at a particular group k, then this group will be relying on a subset of the feature entries. To reflect this fact, for any of the training vectors $h_n \in \mathbb{R}^M$, we shall adopt the notation $h_{k,n} \in \mathbb{R}^{M_k}$ to refer to the subset of the feature vector h_n that is used by group k. Again, we allow for feature entries to be repeated across groups. Accordingly, when we write, for example, $c_{k,1}(h_{k,n})$, this notation is meant to refer to the classifier $c_1(\cdot)$ that is present in group k and which operates on the sub-features of h_n that are included in $h_{k,n}$.

We further assume that there is a graph structure that ties the groups together, shown in Fig. 1, so that if groups k and k' are connected by some edge, then this means that these groups can exchange information over this edge. A non-negative scalar $a_{k',k}$ is assigned to the edge connecting k' to k. These scalars are convex combination coefficients and satisfy:

$$a_{k,k'} = a_{k',k} \geq 0, \quad \sum_{k' \in \mathcal{N}_k} a_{k',k} = 1, \tag{11}$$

where $\mathcal{N}_k$ denotes the set of neighbors of group k; these are the groups that are directly connected to k by edges. If we collect the scalars $\{a_{k',k}\}$ into a $K \times K$ matrix $A = [a_{k',k}]$, then the above property implies that A is a symmetric matrix, and each column and each row of A adds up to one. We say that A is a doubly-stochastic matrix. There are many possible choices for such doubly-stochastic matrices. One popular choice is the Metropolis rule [12].



N data points
M_1
$h_{1,1}$ $h_{1,2}$ $\cdots$ $h_{1,N}$
M_2
$h_{2,1}$ $h_{2,2}$ $h_{2,N}$
M_3
$h_{3,1}$ $h_{3,2}$ $\cdots$ $h_{3,N}$
M_K
$h_{K,1}$ $h_{K,2}$ $\cdots$ $h_{K,N}$
cluster 1
cluster k
$c_{1,\ell}(h_{1,\cdot})$
$c_{K,\ell}(h_{K,\cdot})$
$a_{k',k}$
$a_{k,k'}$
cluster K
cluster k'

Fig. 1. Partitioning of the feature space and network topology.

3.2. Centralized Processing at the Groups

In principle, each group k can run its own gradient boosting procedure and, after T iterations, come up with its own prediction, one that is based on the classifiers in $\mathcal{C}_k$. More broadly, a centralized solution would seek to determine a global prediction function, $\widehat{\gamma}(h)$, that combines all group classifiers in the following manner:

$$\widehat{\gamma}^{(T)}(h) \triangleq \sum_{k=1}^{K} \sum_{t=1}^{T} \alpha_k(t) c_{k,t}(h_{k,\cdot}) \tag{12}$$

in terms of some coefficients $\{\alpha_k(t)\}$ that need to be determined. Observe that we are now attaching a subscript k to coefficients arising from group k. The main difference in the derivation that follows in relation to the gradient boosting derivation from the previous section is that we now need to select a total of K classifiers, one from each group, for each iteration t, along with their corresponding weights. That is, every stage t now involves determining K pairs $\{c_{k,t}^o(h_{k,\cdot}), \alpha_k^o(t)\}$ for $k = 1, 2, \ldots, K$.

Assume that by the end of iteration $t-1$, each group k has already identified the optimal classifiers $\{c_{k,1}^o(h),\, c_{k,2}^o(h), \ldots,\, c_{k,t-1}^o(h)\}$ and their combination weights $\{\alpha_k^o(1),\, \alpha_k^o(2), \ldots, \alpha_k^o(t-1)\}$. Next, we would like to select the next K classifiers, denoted generically by $\{c_{k,t}(h)\}$, and their associated weights $\{\alpha_k(t)\}$, for $k = 1, 2, \ldots, K$, in order to enlarge the aggregate prediction from stage $t-1$ by adding to it a term of the following form:

$$\widehat{\gamma}^{(t)}(h) \;=\; \widehat{\gamma}^{(t-1)}(h) + \sum_{k=1}^{K} \alpha_k(t) c_{k,t}(h_{k,\cdot}) \tag{13}$$

Observe that the aggregate update now involves the sum of K weak classifiers: one from each group k. In order to determine the optimal choices $\{c_{k,t}^o(h), \alpha_k^o(t)\}$ for $k = 1, 2, \ldots, K$, we evaluate the negative gradient of the empirical risk:

$$g_{k,t}(n) \triangleq -\left.\frac{\partial Q(\gamma(n), \widehat{\gamma}(n))}{\partial \widehat{\gamma}(n)}\right|_{\widehat{\gamma}(n) = \widehat{\gamma}^{(t-1)}(n)} \tag{14}$$

and set $c_{k,t}^o(h) = c_{k,\ell_k^o}(h)$, where the optimal index ℓ_k^o, for group k, is obtained by solving:

$$\{\ell_k^o, \beta_k^o\} = \arg\min_{\{1 \leq \ell \leq L_k, \beta_k\}} \sum_{n=1}^{N} \Big(g_{k,t}(n) - \beta_k\, c_{k,\ell}(h_{k,n})\Big)^2 \tag{15}$$

Once the $\{c^o_{k,t}(h)\}$ are selected, we then choose the weights $\{\alpha_k(t)\}$ for the K groups in order to result in the steepest decline in the value of the empirical risk, namely,

$$\{\alpha^o_k(t)\} = \arg\min_{\{\alpha_k\}} \sum_{n=1}^{N} Q\Big(\gamma(n),\, \widehat{\gamma}^{(t-1)}(h_n) + \sum_{k=1}^{K} \alpha_k c^o_{k,t}(h_{k,n})\Big) \tag{16}$$

With the $\{c^o_{k,t}(h), \alpha^o_k(t)\}$ so determined, we can rewrite (13) in terms of these optimal choices:

$$\widehat{\gamma}^{(t)}(h) = \widehat{\gamma}^{(t-1)}(h) + \sum_{k=1}^{K} \alpha^o_k(t) c^o_{k,t}(h_{k,\cdot}) \tag{17}$$

The resulting algorithm is *non-distributed*; nevertheless, it solves the problem of selecting K optimal classifiers and their weights at each stage in order to reduce the empirical risk value sequentially. Since this implementation requires access to global information from across all groups, we shall refer to it as a centralized solution.

3.3. Equivalence via Duality Argument

Our purpose now is to device a fully-decentralized scheme whereby groups rely solely on their local information and on exchanges with their immediate group neighbors in order to construct the aggregate classifier without the need to access global information.

For generality, we consider a regularized version of (16), say,

$$\{\alpha^o_k(t)\} = \arg\min_{\{\alpha_k\}} \; \rho \sum_{k=1}^{K} q(\alpha_k) \; + \sum_{n=1}^{N} Q\left(\gamma(n), \widehat{\gamma}^{(t-1)}(h_n) + \sum_{k=1}^{K} \alpha_k c^o_{k,t}(h_{k,n})\right) \tag{18}$$

where $\rho > 0$ is a regularization parameter and $q(\cdot)$ is a convex regularization function. The key observation is that the objective function in (18) has the form of a "cost-of-sum" since the argument of $Q(\cdot)$ involves a sum in terms of the unknowns, $\{\alpha_k\}$. The duality argument will show that this "cost-of-sum" form can be transformed into an equivalent "sum-of-cost" form, which is particularly amenable to distributed implementations [24].

We start by introducing, for every $n = 1, 2, \ldots, N$, a dummy scalar variable $z(n)$ and replace (18) by:

$$\begin{aligned} \min_{\{z,\alpha\}} \quad & \rho \sum_{k=1}^{K} q(\alpha_k) + \sum_{n=1}^{N} Q\left(\gamma(n), \widehat{\gamma}^{(t-1)}(h_n) + z(n)\right) \\ \text{subject to} \quad & z(n) = \sum_{k=1}^{K} \alpha_k c^o_{k,t}(h_{k,n}), \quad \text{for } n = 1, 2, \ldots, N \end{aligned} \tag{19}$$

It is easy to see that problem (19) is a standard convex optimization problem and that, under the linear equality constraints, strong duality holds [11]. The corresponding dual function is given by:

$$\begin{aligned} \mathcal{D}(\lambda) \triangleq & \sum_{n=1}^{N} \inf_{z(n)} \Big\{ Q\left(\gamma(n), \widehat{\gamma}^{(t-1)}(h_n) + z(n)\right) + \lambda(n) z(n) \Big\} \\ & + \rho \sum_{k=1}^{K} \inf_{\alpha_k} \left\{ q(\alpha_k) - \sum_{n=1}^{N} \frac{\lambda(n) c^o_{k,t}(h_{k,n})}{\rho} \alpha_k \right\} \end{aligned} \tag{20}$$

where the primal variables $\{\alpha^o_k(t, \lambda), z^o(n, \lambda)\}$ and the dual variable λ are related via:

$$\{z^o(n, \lambda), \alpha^o_k(t, \lambda)\} = \arg\min_{\{z,\alpha\}} \mathcal{L}(z, \alpha, \lambda) \tag{21}$$

It will be shown later that the primal variables can be recovered in a distributed manner. We can now call upon the concept of conjugate functions. For any function $r(x)$ of a scalar variable x, the conjugate function is denoted by $r^\star(\nu)$ [11], where ν is a scalar argument, and defined as $r^\star(\nu) = \sup_x (\nu x - r(x))$. For many common regularization forms, closed form expressions exist for $q^\star(\nu)$ — see [24, 25]. The first minimization in (20) can also be expressed in closed form in important cases, such as when the loss function $Q(\cdot)$ is chosen as the exponential loss, the square loss, or the logit loss. For now, we denote the minimum value of the first term generically by:

$$Q^o(n, \lambda) \triangleq -\inf_{z(n)} \left\{ Q\left(\gamma(n), \widehat{\gamma}^{(t-1)}(h_n) + z(n)\right) \; + \; \lambda(n) z(n) \right\} \tag{22}$$

so that the expression (20) can be written as:

$$\mathcal{D}(\lambda) = -\sum_{k=1}^{K} J_k(\lambda), \tag{23}$$

$$J_k(\lambda) \; \triangleq \; \rho q^\star \left(\sum_{n=1}^{N} \frac{\lambda(n) c^o_{k,t}(h_{k,n})}{\rho} \right) + \frac{1}{K} \sum_{n=1}^{N} Q^o(n, \lambda) \tag{24}$$

Therefore, the problem of determining the optimal dual variable, λ^o, can be equivalently stated as

$$\min_{\lambda} \sum_{k=1}^{K} J_k(\lambda) \implies \lambda^o \tag{25}$$

The "sum-of-costs" formulation (25) is convenient because it admits efficient distributed solutions of the consensus or diffusion type [12, 13], meaning that each group k is now able to estimate λ^o on its own by interacting solely with its neighbors. We shall denote these local estimates by λ^o_k, with a subscript k to indicate the group index.

3.4. Diffusion Learning

In the diffusion implementation, at every stage t and starting from some initial value, each group k repeats the following computations a couple of times until its estimate for the λ, denoted by $\lambda_{k,i}$ at time i, converges close enough to a limiting value, denoted by λ^o_k; this limiting value is a local estimate for the dual variable λ^o.

```
Diffusion strategy [13] (run by every group $k$)
repeat for $i = 1, 2, \ldots, I$:
    $\phi_{k,i} = \lambda_{k,i-1} - \mu \nabla_\lambda J_k(\lambda_{k,i-1})$
    $\lambda_{k,i} = \sum_{k' \in \mathcal{N}_k} a_{k'k} \phi_{k',i}$
end
set $\lambda^o_k = \lambda_{k,I}$
```

In the above adapt-then-combine (ATC) diffusion strategy [12,13,26], for every i, agent k first moves along the negative direction of its cost gradient to generate the intermediate estimate $\phi_{k,i}$, followed by a consultation step where it combines the intermediate estimates $\{\phi_{k',i}\}$ from its neighbors to obtain $\lambda_{k,i}$. We shall represent the diffusion strategy more compactly by writing

$$\lambda^o_k = \text{diffusion}\,\{J_k(\lambda), \mathcal{N}_k, I\} \tag{26}$$

where $\mathcal{N}_k$ denotes the neighborhood of group k, and I denotes the number of iterations to be used; this parameter is set by the designer. All groups apply the diffusion strategy simultaneously. Consequently, every group k will end up with a local version, λ^o_k, for the global dual variable λ^o. In this way, each group k can now compute a local version for $z^o(n)$ and its optimal coefficient $\alpha^o_k(t)$ by solving:

$$z^o_k(n) = \arg\min_{z} \left\{ Q\left(\gamma(n), \widehat{\gamma}^{(t-1)}_k(h_n) + z\right) + \lambda^o_k(n) z \right\} \tag{27}$$

$$\alpha^o_k(t) = \arg\min_{\alpha_k} \left\{ q(\alpha_k) \; - \; \left(\sum_{n=1}^{N} \frac{\lambda^o_k(n) c^o_{k,t}(h_{k,n})}{\rho} \right) \alpha_k \right\} \tag{28}$$

We now explain how the prediction variables can be estimated for arbitrary features, h. Indeed, note that after completing T stages of the diffusion strategy to learn the dual variable, each agent k will have available its optimal coefficients $\alpha_k^o(t)$ and classifier selections $c_{k,t}^o(\cdot)$. During testing, when a new feature vector h is received, each agent k is able to use this local information to evaluate:

$$b_k^{(T)}(h) \triangleq \sum_{t=1}^T \alpha_k^o(t) c_{k,t}^o(h_{k,\cdot}) \tag{29}$$

Then, from the general form (12) we know that the overall prediction variable is the aggregate sum of these individual decision variables. When A is doubly-stochastic, this sum can be evaluated in a distributed manner by each agent k running the traditional consensus iteration [13, 27] to combine repeatedly local values.

```
Local averaging (run by every group k)
Initialization : start from $s_k^{(0)}(h) = b_k^{(T)}(h)$
repeat for $j = 1, 2, \ldots, J$:
    $s_k^{(j)}(h) = \sum_{k' \in \mathcal{N}_k} a_{k'k}\, s_k^{(j-1)}(h)$
end
set $\widehat{\gamma}_k^{(T)}(h) = K \cdot s_k^{(J)}(h)$
```

We represent the above averaging procedure compactly by writing:

$$\widehat{\gamma}_k^{(T)}(h) = K \cdot \text{average}\left\{b_k^{(T)}(h), \mathcal{N}_k, J\right\} \tag{30}$$

In summary, the resulting distributed algorithm is the following.

```
Diffusion gradient boosting algorithm
Initialization:  choose $\widehat{\gamma}_k^{(0)}(n)$, for $n = 1 \ldots N$, $k = 1 \ldots K$
repeat for $t = 1, 2, \ldots, T$:
    for every agent (in parallel) $k = 1, 2, \ldots, K$:
        $g_{k,t}(n) = -\left.\frac{\partial Q(\gamma, \widehat{\gamma})}{\partial \widehat{\gamma}(n)}\right|_{\widehat{\gamma} = \widehat{\gamma}_k^{(t-1)}(n)}$   (31)
        $\{\ell_k^o, \beta_k^o\} = \arg\min_{\{1 \le \ell \le L_k, \beta_k\}} \sum_{n=1}^N \left(g_{k,t}(n) - \beta_k\, c_{k,\ell}(h_{k,n})\right)^2$   (32)
        $c_{k,t}^o(h) = c_{k,\ell_k^o}(h)$   (33)
        $J_k(\lambda) = \text{expression (23)}$   (34)
        $\lambda_k^o = \text{diffusion}\,\{J_k(\lambda), \mathcal{N}_k, I\}$   (35)
        $\alpha_k^o(t) = \arg\min_{\alpha_k} \left\{ q(\alpha_k) - \left(\sum_{n=1}^N \frac{\lambda_k^o(n) c_{k,t}^o(h_{k,n})}{\rho}\right)\alpha_k \right\}$   (36)
        $b_k^{(t)}(h) = b_k^{(t-1)}(h) + \alpha_k^o(t) c_{k,t}^o(h)$   (37)
    end
end
$\widehat{\gamma}_k^{(t)}(h) = K \cdot \text{average}\left\{b_k^{(t)}(h), \mathcal{N}_k, J\right\}$   (38)
```

4. SIMULATION ON SPECIAL LOSS FUNCTIONS

In this section, we consider the exponential loss function, $Q(\gamma, \widehat{\gamma}) = e^{-\gamma\widehat{\gamma}}$, which is associated with AdaBoost learning. In this case, by exploiting the fact that $\gamma(n), c_{k,\ell}(h) \in \{\pm 1\}$, it can be verified that

$$-\left.\frac{\partial Q(\gamma(n), \widehat{\gamma}(n))}{\partial \widehat{\gamma}(n)}\right|_{\widehat{\gamma}(n) = \widehat{\gamma}_k^{(t-1)}(n)} = \gamma(n)\tau_{k,t}(n), \tag{39}$$

$$\tau_{k,t}(n) \triangleq \exp\{-\gamma(n)\widehat{\gamma}_k^{(t-1)}(n)\} \tag{40}$$

$$\ell_k^o = \arg\min_{1 \le \ell \le L_k} \sum_{n=1}^N \tau_{k,t}(n)\, \mathbb{I}\left[c_{k,\ell}(h_{k,n}) \ne \gamma(n)\right] \tag{41}$$

where $\mathbb{I}[x]$ denotes the indicator function; it is equal to one when its argument is true and zero otherwise. Result (41) indicates that ℓ_k^o is selected as the optimal classifier that results in the smallest sum of weights $\tau_{k,t}(n)$ over the misclassified data. Next, we evaluate the function $Q^o(n, \lambda)$ defined by (22) and find:

$$Q_{k,t}^o(n, \lambda) = \gamma(n)\lambda(n)\left[\ln\left(\frac{\gamma(n)\lambda(n)}{\tau_{k,t}(n)}\right) - 1\right] \tag{42}$$

Assume we employ elastic-net regularization:

$$q(x) = \delta|x| + \frac{1}{2}|x|^2 \iff q^\star(\nu) = \frac{1}{2}\left|\mathcal{T}_\delta(\nu)\right|^2 \tag{43}$$

where $\mathcal{T}_\delta(\nu)$ represents the soft-thresholding operator $\mathcal{T}_\delta(\nu) = \text{sgn}(\nu) \cdot \max(|\nu| - \delta, 0)$. It then follows from (23) and (28) that

$$\alpha_k^o(t) = \mathcal{T}_\delta\left(\frac{1}{\rho}\sum_{n=1}^N \lambda_{k,t}^o(n) c_{k,t}^o(h_{k,\cdot})\right) \tag{44}$$

$$\widehat{\gamma}_k^{(t)}(n) = \widehat{\gamma}_k^{(t-1)}(n) - \gamma(n)\ln\left(\frac{\gamma(n)\lambda_{k,t}^o(n)}{\tau_{k,t}(n)}\right) \tag{45}$$

We can use this update to derive an alternative expression for the weight $\tau_{k,t}(n)$ in terms of the local dual variable $\lambda_{k,t}^o(n)$ by $\tau_{k,t+1}(n) = \gamma(n)\lambda_{k,t}^o(n)$ It is useful to assign $\lambda_{k,t-1}^o(n)$ as a starting point for $\lambda_{k,t}^o(n)$ in the diffusion update (26).

We now compare the performance of the diffusion Adaboost implementation with elastic-net regularization against the standard (centralized) Adaboost algorithm [8]. The set of weak classifiers is chosen as

$$c_\ell(h_n) = \text{sign}(h_n(p) > \text{thres}_\text{p}) \tag{46}$$

For the diffusion AdaBoost setting, we assigned 10 groups, which are connected through a random doubly stochastic matrix. Each group is in charge of one-tenth of the feature entries and the corresponding weak classifiers. The test data is obtained from the LIBSVM website[1]. We use the Reuters Corpus Volume I (RCV1) dataset. The agent setting is the same and the parameter setting is $\rho = 0.01$, $\delta = 0.2$, and $\mu = 1 \times 10^{-6}$.

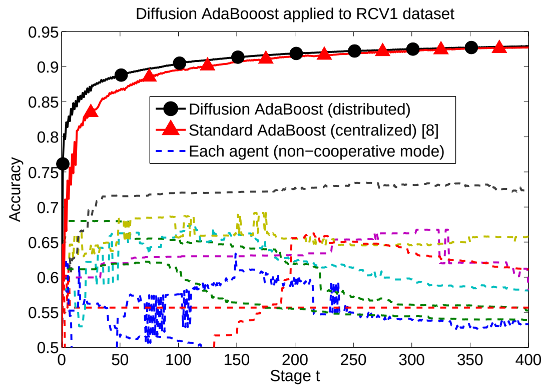


Fig. 2. Evolution of the performance curves.

One observation stands out from these results. The dotted lines in the figure confirm that if the individual groups were to rely solely on their classifiers to solve the inference task, then their performance will be poor. However, once they start cooperating locally and sharing local information, the network of dispersed groups is able to match the performance of the centralized Adaboost solution.

[1] `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets`

5. REFERENCES

[1] T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning*, Springer, 2009.

[2] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.

[3] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[4] T. Hastie and R. Tibshirani, *Generalized Additive Models*, CRC Press, 1990.

[5] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proc. ICML*, Bari, Italy, 1996, vol. 96, pp. 148–156.

[6] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. CVPR*, HI, USA, 2001, vol. 1, pp. I–511.

[7] L. Breiman, "Arcing classifier," *The Annals of Statistics*, vol. 26, no. 3, pp. 801–849, 1998.

[8] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *The Annals of Statistics*, pp. 1189–1232, 2001.

[9] L. Mason, J. Baxter, P. Bartlett, and M. Frean, "Boosting algorithms as gradient descent in function space," in *Proc. NIPS*, 1999.

[10] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation*, Prentice-Hall, 1989.

[11] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004.

[12] A. H. Sayed, "Adaptive networks," *Proceedings of the IEEE*, vol. 102, no. 4, pp. 460–497, 2014.

[13] A. H. Sayed, "Adaptation, learning, and optimization over networks," *Foundations and Trends in Machine Learning*, vol. 7, no. 4-5, pp. 311–801, 2014.

[14] A. Lazarevic and Z. Obradovic, "Boosting algorithms for parallel and distributed learning," *Distributed and Parallel Databases*, vol. 11, no. 2, pp. 203–229, 2002.

[15] A. Lazarevic and Z. Obradovic, "The distributed boosting algorithm," in *Proc. ACM SIGKDD*, San Francisco, CA, USA, 2001, pp. 311–316.

[16] W. Fan, S. J. Stolfo, and J. Zhang, "The application of adaboost for distributed, scalable and on-line learning," in *Proc. ACM SIGKDD*, San Diego, CA, USA, 1999, pp. 362–366.

[17] L. Breiman, "Pasting small votes for classification in large databases and on-line," *Machine Learning*, vol. 36, no. 1-2, pp. 85–103, 1999.

[18] N. V. Chawla, L. O. Hall, K. W. Bowyer, T.E. Moore Jr., and W. P. Kegelmeyer, "Distributed pasting of small votes," in *Multiple Classifier Systems*, F. Roli and J. Kittler, Eds., pp. 52–61. Springer, June 2002.

[19] N. V. V Chawla, L. O. O Hall, K. W. Bowyer, and W. P. Kegelmeyer, "Learning ensembles from bites: A scalable and accurate approach," *The Journal of Machine Learning Research*, vol. 5, pp. 421–451, 2004.

[20] D. P. Bertsekas, *Nonlinear Programming*, Athena scientific Belmont, 1999.

[21] J. H. Friedman, "Stochastic gradient boosting," *Computational Statistics and Data Analysis*, vol. 38, no. 4, pp. 367–378, 2002.

[22] J. Friedman and R. Hastie, T.and Tibshirani, "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)," *The Annals of Statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[23] L. Breiman, J. Friedman, C. Stone, and R. Olshen, *Classification and Regression Trees*, CRC press, 1984.

[24] J. Chen, Z. J. Towfic, and A. H. Sayed, "Dictionary learning over distributed models," *IEEE Trans. Signal Process.*, vol. 63, no. 4, pp. 1001–1016, 2015.

[25] J. M. Borwein and A. S. Lewis, *Convex Analysis and Nonlinear Optimization: Theory and Examples*, Springer, 2010.

[26] J. Chen and A. H. Sayed, "On the learning behavior of adaptive networks—Part I: Transient analysis," *IEEE Trans. Information Theory*, vol. 61, no. 6, pp. 3487–3517, June 2015.

[27] M. H. DeGroot, "Reaching a consensus," *Journal of the American Statistical Association*, vol. 69, no. 345, pp. 118–121, 1974.