Protocols for Learning Classifiers on Distributed Data

Hal Daumé III
Dept. of Computer Sc.
University of Maryland
hal@umiacs.umd.edu

Jeff M. Phillips
School of Computing
University Of Utah
jeffp@cs.utah.edu

Avishek Saha
School of Computing
University Of Utah
avishek@cs.utah.edu

Suresh Venkatasubramanian
School of Computing
University Of Utah
suresh@cs.utah.edu

Abstract

We consider the problem of learning classifiers for labeled data that has been distributed across several nodes. Our goal is to find a single classifier, with small approximation error, across all datasets while minimizing the communication between nodes. This setting models real-world communication bottlenecks in the processing of massive distributed datasets. We present several very general sampling-based solutions as well as two-way protocols which have a provable exponential speed-up over any one-way protocol. We focus on core problems for *noiseless* data distributed across two or more nodes. The techniques we introduce are reminiscent of active learning, but rather than actively probing labels, nodes actively communicate with each other, each node simultaneously learning important data from another node.

1 Introduction

Distributed learning [Bekkerman et al., 2011] is the study of machine learning on data distributed across multiple locations. Examples of this setting include data gathered from sensor networks, or from data centers located across the world, or even from different cores on a multicore architecture. In all cases, the challenge lies in solving learning problems with minimal *communication* overhead between nodes; learning algorithms cannot afford to ship all data to a central server, and must use limited communication efficiently to perform the desired tasks.

In this paper, we introduce a framework for studying distributed classification that treats inter-node communication as a limited resource, and present a number of algorithms for this problem that uses inter-node interaction to reduce communication. Our main technique is the use of carefully chosen data and classifier descriptors that convey the most useful information about one node to another; in that respect, our work makes use of (in spirit) the *active learning* paradigm [Settles, 2009].

For distributed classification, the dominant strategy [Predd et al., 2006, McDonald et al., 2010, Mann et al., 2009, Lazarevic and Obradovic, 2001] is to design local classifiers that work well on individual nodes. These classifiers are then communicated to a central server, and then aggregation strategies like voting, averaging, or even boosting are used to compute a global classifier. These approaches, while designed to improve communication, do not study communication as a resource to be used sparingly, and ignore the fact that interactions between nodes might reduce communication even further by allowing them to learn from each others' data.

Problem definition. There are many aspects to formalizing the problem of learning classifiers with limited communication, including discussion of the data sources (i.i.d. or adversarial), data quality (noiseless or noisy), communication models (one-way, two-way or k-way) and classifier models (linear, non-linear, mixtures). In this paper, we focus on a simple core model that illustrates both the challenges and the benefits of focusing on the communication bottleneck.

In our model, we first consider one-way and two-way communication between *two* parties Alice and Bob that receive *noiseless* data sets D_A and D_B that result from partitioning a larger data set $D = D_A \cup D_B$. Thereafter, we consider one-way and two-way communication between k parties $P_1, P_2, \ldots, P_k$ that receive noiseless data sets $D_1, D_2, \ldots, D_k$ partitioned from $D = \bigcup_{i=1}^{k} D_i$. In either case, the partitioning may be done randomly, but might also be adversarial: indeed, a number of recent discussions [Cesa-Bianchi et al., 2009, Dekel et al., 2010, Laskov and Lippmann, 2010, adv, 2010, Hsu and Langford, 2011] highlight the need to consider adversarial data in learning scenarios.

In our model, the nodes together learn (via communication) a classifier h_k (h_{AB} for two nodes A and B) from a family of classifiers such as linear classifiers. Let h^* denote the

Appearing in Proceedings of the 15th International Conference on Artificial Intelligence and Statistics (AISTATS) 2012, La Palma, Canary Islands. Volume XX of JMLR: W&CP XX. Copyright 2012 by the authors.

Hypothesis Class	Dimensions	Communication Protocol	Error	Communication Complexity		Reference
				Two-party	k-party	
generic	d	one-way	ε	$O(\nu/\varepsilon \log \nu/\varepsilon)$	$O(k(\nu/\varepsilon)\log \nu/\varepsilon)$	Theorem 3.1 & 5.1
thresholds	1	one-way	0	2	$2k$	Lemma 3.1 & 5.2
aa-rectangles	d	one-way	0	$4d$	$4dk$	Theorem 3.2 & 5.2
hyperplanes	d	one-way	ε	$\Omega(1/\varepsilon)$	$\Omega(k/\varepsilon)$	Theorem 3.3 & 3.4
hyperplanes	2	two-way	ε	$O(\log 1/\varepsilon)$	$O(k^2 \log 1/\varepsilon)$	Theorem 4.1 & 5.3

Table 1: Summary of results obtained for different hypotheses classes under an *adversarial* model with one-way and two-way communications. All results are for the *noiseless* setting. ν denotes the VC-dimension for the family of classifiers.

optimal classifier that can be learned on D. Let $E_D(h)$ denote the number of points misclassified by some classifier h on D. We say that h_k has *ε-approximation error* (ε-error for short) on D if $E_D(h_k) - E_D(h^*) \leq \varepsilon|D|$. *The goal is for h_k to have at most ε-error ($0 < \varepsilon < 1$) while minimizing inter-node communication.*

In this paper, we phrase the learning task in terms of training error, rather than generalization. This is motivated by numerous results that indicate that low training error combined with limits on the hypothesis class used lead to good generalization bounds [Kearns and Vazirani, 1994].

Technical contributions. Our overall contribution, in this paper, is to model communication minimization (in distributed classification) as an active probing problem. We start in Section 2 by showing that, within our proposed framework, the one-way communication problem can be solved trivially under i.i.d. assumptions (ref. Section 2). Hence in this work most of our effort is focused on *adversarial distributions*. In all subsequent cases, we first help build intuition by discussing a two-party protocol and thereafter extend the two-party results to the k-party case. In Section 3 we show that with *one-way* communication, it is possible to learn optimal classifiers *exactly* (i.e., with 0-error) for thresholds (in $\mathbb{R}^1$), intervals (in $\mathbb{R}^1$) and axis-aligned rectangles (in $\mathbb{R}^d$) with only a *constant* amount of communication. For the case of linear separators, we present an $\Omega(1/\varepsilon)$ lower bound. Thereafter in Section 4, we present our *two-way, two-party* communication protocol ITERATIVESUPPORTS which learns an ε-error classifier (under adversarial distributions) using *only* $O(\log 1/\varepsilon)$ communication – *an exponential improvement over the one-way case!* Next in Section 5, we use the results of Section 4 to obtain an $O(k^2 \log 1/\varepsilon)$ bound for k-parties using two-way communication. In Section 6, we present results that demonstrate the correctness and convergence of the linear separator algorithms and also empirically compare its performance with a few other baselines.

Table 1 summarizes the results obtained with references to appropriate sections of this paper. All our results pertain to the noiseless setting which assumes the existence of a classifier that perfectly separates the data. However, in Section 7, we propose some ways to extend our proposed results to noisy data.

2 Randomly Partitioned Distributions

We first consider the case when the data is partitioned randomly among nodes. Specifically, each node i can view its data D_i as being drawn iid from $D \subset \mathbb{R}^d$. We can now apply learning theory results for any family of classifiers $\mathcal{H}$ with bounded VC-dimension ν. Any classifier $h_S \in \mathcal{H}$ which perfectly separates a random sample S of $s = O((\nu/\varepsilon)\log(\nu/\varepsilon))$ samples from D has at most ε-classification error on D, with constant probability [Anthony and Bartlett, 2009]. Thus each D_i can be viewed as such a sample S and if D_i is large enough, with no communication a node can return a classifier with small error.

Theorem 2.1. *Let $\{D_1, \ldots, D_k\}$ randomly partition $D \subset \mathbb{R}^d$. In the noiseless setting a node i can produce a classifier from $(\mathbb{R}^d, \mathcal{H})$ (with VC-dimension ν) with at most ε-error for $\varepsilon = O((\nu/|D_i|)\log|D_i|)$, with constant probability.*

A similar result (with slightly worse dependence on the D_i) can be obtained for the noisy setting. These results indicate that the k-party (and hence also two-party) setting is trivial to solve if we assume random partitioning of D. Thus, for the remainder of the paper we focus on protocols for adversarially partitioned data.

3 *One-way* Two-Party Protocols

Consider first a generic setting, with $D \subset \mathbb{R}^d$ and family of hypothesis $\mathcal{H} \subset 2^D$ so $(\mathbb{R}^d, \mathcal{H})$ has VC-dimension ν.

Theorem 3.1. *Assume there exists a 0-error classifier $h^* \in \mathcal{H}$ on D where $(D, \mathcal{H})$ has VC-dimension ν. Then A sending $s_\varepsilon = O((\nu/\varepsilon)\log(\nu/\varepsilon))$ random samples ($S_A \subset D_A$) to B allows B to, with constant probability, produce an ε-error classifier $h \in \mathcal{H}$.*

Proof. The classifier returned by B will have 0 error on $D_B \cup S_A$; thus it only has error on D_A. Since S_A is an ε-net of D_A with constant probability, then it has at most ε-error on D_A and hence at most ε-error on $D_A \cup D_B = D$. $\square$

A similar result with $s_\varepsilon = O(\nu/\varepsilon^2)$ applies to the noisy setting. An important technical contribution of this paper is to show that in many cases we can improve upon these general results.

3.1 Specific Hypothesis Classes

Thresholds. First we describe how to find a threshold $t \in \mathcal{T} \subset \mathbb{R}$ such that all points $p \in D$ with $p < t$ are negative and with $p > t$ are positive. A sends to B a set S_A consisting of two points in D_A: its largest negative point p^- and its smallest positive point p^+. Then B returns a 0-error classifier on $D_B \cup S_A$.

Lemma 3.1. *In $O(1)$ one-way communication we can find a 0-error classifier in $(D, \mathcal{T})$.*

Proof. The optimal classifier $t \in \mathcal{T}$ must lie in the range $[p^-, p^+]$ otherwise, it would misclassify some point in D_A, breaking our noiseless assumption. Then any 0-error classifier on D_B within this range is has 0 error on D. □

Intervals. We can now apply Lemma 3.1 to get stronger bounds. In particular, this generalizes to the family $\mathcal{I}$ of intervals in $\mathbb{R}^1$. First A finds h_A, its optimal classifier for D_A. This interval has two end points each of which lies in between a pair of a positive and a negative point (if there are no negative or no positive points, A returns the empty set). These two pairs of points form a set S_A that A sends to B. B now returns the classifier that optimally separates $D_B \cup S_A$, and if S_A is empty then the interval classifier is as small as possible.

Lemma 3.2. *In $O(1)$ one-way communication we can find a 0-error classifier $h \in \mathcal{I}$.*

Proof. When S_A is nonempty, this encodes two versions of Lemma 3.1. Assume without loss of generality that the positive points are contained in an interval with negative points lying outside the interval. Then we can pick any positive point p from either set D_A or D_B and consider the points greater than p in the first instance of Lemma 3.1 and points less than p in the second instance. Invoking Lemma 3.1 proves this case. When S_A is empty, and a perfect classifier exists, then the minimal separating interval on D_B will not violate any points in S_A, and will have no error. □

Axis-aligned rectangles. We now consider finding a 0-error classifier from the family $\mathcal{R}^d$ of all axis-aligned rectangles in $\mathbb{R}^d$. An axis-aligned rectangle $R \in \mathcal{R}^d$ can be defined by d-values in $\mathbb{R}^d$, a minimum and maximum value along each coordinate axis. Given a data set P, the *minimum axis-aligned rectangle* for P is the smallest axis-aligned rectangle that contains all of P; that is, it has the smallest maximum coordinate possible along each coordinate axis and the largest minimum coordinate possible along each coordinate axis. These $2d$ terms can be optimized independently as long as P is non-empty.

For a dataset D_A we can define two minimum axis-align rectangles R_A^+ and R_A^- defined on the positive and negative points, respectively. If the positive or negative point set is empty, then each coordinate minimum and maximum is set to a special character $\emptyset$. Two such rectangles can be defined for D_B and $D = D_{A\cup B}$ in the same way.

Theorem 3.2. *A one-way protocol where A sends R_A^+ and R_A^- to B is sufficient to find a 0-error classifier $h_{AB} \in \mathcal{R}^d$ in the noiseless setting. It requires $O(d)$ communication complexity.*

Proof. The key observation is that the minimum axis-aligned rectangle that contains R_A^+ and R_B^+ is precisely $R_{A\cup B}^+$ (and symmetrically for negative points). Since the minimum and maximum for each coordinate axis is set independently, then we can optimize each using that value from R_A^+ and R_B^+. Thus B can compute this using points from D_B and R_A^+.

First, consider the case where positive points are inside the classifier and negative points are outside. Since there exist a 0-error classifier h^*, then $R_{A\cup B}^+$ must be contained in that classifier, since no smaller classifier can contain all positive points. It follows by our assumption that h^* and thus also $R_{A\cup B}^+$ contains no negative points, and can be returned as our 0-error classifier h_{AB}. B can determine if positive or negative points are inside by which of $R_{A\cup B}^+$ and $R_{A\cup B}^-$ is smaller. If R_A^+ or R_A^- is $\emptyset$, then $R_{A\cup B}^+ = R_B^+$ or $R_{A\cup B}^- = R_B^-$, respectively. □

Hyperplanes in $\mathbb{R}^2$. The positive results from simpler geometric concepts do not extend to hyperplanes. We present the following two results but defer any formal analysis to a full version.

Theorem 3.3. *Using only one-way communication from A to B, it requires $\Omega(1/\varepsilon)$ communication to find an ε-error linear classifier in $\mathbb{R}^2$.*

Note that due to Theorem 2.1, this is tight up to a $\log(1/\varepsilon)$ factor for one-way communication.

We can extend this lower bound to the k-node one-way model of computation where we assume each node P_i can only send data to P_{i+1}. In this case, we give node A's input to P_1, and node B's input to node P_k, and nodes P_i for $i \in [2, k-1]$ have no data. Then each node P_i is forced to send the $\Omega(1/\varepsilon)$ communication that A wants to send to B along the chain.

Theorem 3.4. *Using only one-way communication among k-players in a chaining model, it requires $\Omega(k/\varepsilon)$ communication to find an ε-error linear classifier in $\mathbb{R}^2$.*

4 *Two-way* Two-Party Protocols for Linear Separators

In this section, we present a two-party algorithm that uses two-way communication to learn an ε-optimal combined classifier h_{AB}.

4.1 Algorithm Overview

Our algorithm proceeds in rounds. In each round both nodes send a constant number of points to the other. The

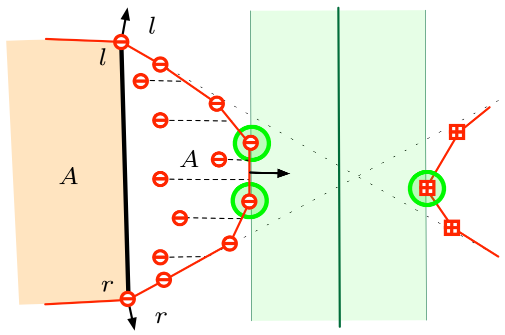


Figure 1: 3 support points chosen from U_A, and the family of 0-error classifiers for A parallel to h_A.

goal is to limit the number of rounds to $O(\log(1/\varepsilon))$ resulting in a total communication complexity of $O(\log(1/\varepsilon))$. At the end of $O(\log(1/\varepsilon))$ rounds of communication, the algorithm yields a combined classifier h_{AB} that has ε error on D.

In order to bound the number of rounds, each node must maintain information about which points the *other node* might be classifying correctly or not at any stage of the algorithm. Specifically, suppose node A is sent a classifier h_B from node B (learned on D_B and hence has zero error on D_B) and this classifier misclassifies some points in D_A. We denote these points as the *Set of Disagreement* (*SOD*) where $SOD \subseteq D_A$. The remaining points in D_A can be divided into the *Set of Total Agreement* (*SOTA*), which are the points on which classifiers from A and B will continue to agree on in the future, and the *Set of Luck* (*SOL*), which are points on which the two nodes currently agree, but might disagree later on. The set of disagreement and the set of luck together form the *Set of Uncertainty* $SOU = SOD \cup SOL$, representing all points that may or may not be classified incorrectly by B in the future.

Our goal will be to show that the *SOU* decreases in cardinality by a constant factor in each round. Achieving this will guarantee that at the end of $\log(1/\varepsilon)$ rounds, the size of the *SOU* will be at most an ε-fraction of the total input. Since $|SOU| \geq |SOD|$, we obtain the desired ε-error classifier.

Definitions and notation. Let $\mathcal{P}_A^+$ and $\mathcal{P}_A^-$ denote polytopes that contain positive and negative points in D_A, respectively. Let $\mathcal{C}_A^+$ and $\mathcal{C}_A^-$ denote the convex hulls formed by the positive and negative *SOTA* in D_A after the i^{th} round, respectively. In general, when sets have a $+$ or $-$ superscript it will denote the restriction of that set to only positive or negative points, respectively. Often to simplify messy but usually straightforward technical details we will drop the superscript and refer to either or both sets simultaneously. We denote the *region of uncertainty* $\mathcal{U}_A$ as $\mathcal{P}_A \setminus \mathcal{C}_A$, and note $U_A = \mathcal{U}_A \cap D_A$.

In each round A will send to B a set $S_A \subset D_A$; these points imply a max-margin classifier h_A on S_A that has 0 error on D_A; see Figure 1. Then B will either terminate with an ε-error classifier h_B, or symmetrically return a set of points $S_B \subset D_B$. This process is summarized in Algorithm 1.

Algorithm 1 ITERATIVESUPPORTS

```
Input: $D_A$ and $D_B$
Output: $h_{AB}$ (classifier with $\varepsilon$-error on $D_A \cup D_B$)
$S_A :=$ SUPPORT($D_A$); send $S_A$ to $B$;
while (1) do
    ——— B's move ———
    compute error (err) using $h_A$ (from $S_A$) on $D_B$;
    if(err $\leq \varepsilon|D_B|$) then exit;
    $D_B = D_B \cup S_A$; $S_B :=$ SUPPORT($D_B$); send $S_B$ to $A$;
    ——— A's move ———
    compute error (err) using $h_B$ (from $S_B$) on $D_A$;
    if(err $\leq \varepsilon|D_A|$) then exit;
    $D_A = D_A \cup S_B$; $S_A :=$ SUPPORT($D_A$); send $S_A$ to $B$;
end while
```

Two aspects remain: determining if a player may exit the protocol with a ε-error classifier (early termination), and computing the support points in the function SUPPORT.

4.2 Early Termination

Note that in Algorithm 1, under certain *early-termination* conditions, player B may terminate the protocol and return a valid classifier, even if h_A has more than ε error on D_B. Any classifier that is parallel to h_A and is shifted less than the margin of the max-margin classifier also has 0 error on D_A. Thus if any such classifier has at most ε-error on D_B, player B can terminate the algorithm and return that classifier.

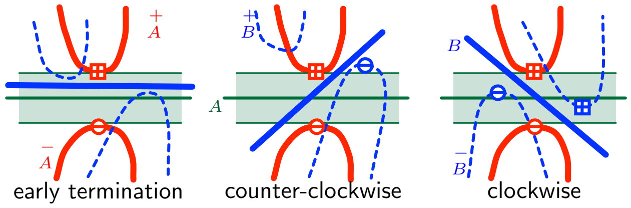


Figure 2: Cases for either early termination, or for the direction of the normal to the linear separator being forced counter-clockwise or clockwise.

This early-termination observation is important because it allows B to send to A information regarding a 0-error classifier, with respect to h_A, and the points S_A that define it. If B cannot terminate, then either some point in D_B must be completely misclassified by all separators within the margin, or some negative point in D_B and some positive point in D_B must both be in the margin and cannot be separated; see Figure 2. Either scenario implies that any ε-error classifier on D_B must rotate in some direction (either clockwise or counter-clockwise) relative to h_A. This is important, because it informs A that all points on $\partial\mathcal{P}_A$ (the boundary of $\mathcal{P}_A$) in the clockwise (resp. counter-clockwise) direction from S_A will never be misclassified by B if h_B rotates in the counterclockwise (resp. clockwise) direction from h_A, increasing the SOTA, and decreasing the SOU.

4.3 Choice of Support Points

What remains is describing how A chooses a set S_A, i.e. how to implement the subroutine SUPPORT in Algorithm 1. If the set S_A always has half of U_A on either side, then this process will terminate in at most $O(\log(1/\varepsilon))$ rounds, via the consequences of no early-termination. But if no points are on one side of S_A and B's response always forces h_A to rotate towards the other side, then this cannot be assured. Thus the set S_A should be chosen judiciously to ensure that $|U_A|$ decreases by at least half each round.

We present two methods to choose S_A. This first does not have the half-on-either-side guarantee, but is a very simple heuristic that often works quite well, even in higher dimensions (see Section 6). The second is only slightly more complicated and is designed precisely to have this half-on-either-side guarantee. Both methods start by computing the region of uncertainty $\mathcal{U}_A$ and the set of its points D_A which lie in that region U_A.

The first is called MAXMARG, and simply chooses the max-margin support points as S_A. These points may include points sent over in previous iterations from B to A.

The second is called MEDIAN, and is summarized in Algorithm 2 (shown from A's perspective). It projects all of U_A onto $\partial\mathcal{P}_A$ (the boundary of $\mathcal{P}_A$); this creates a weight for each edge of $\partial\mathcal{P}_A$, defined by the number of points projected onto it. Then MEDIAN chooses the weighted median edge E. Finally, the orientation of h_A is set parallel to edge E, and the corresponding support vectors are constructed.

Algorithm 2 SUPPORT implemented as MEDIAN

```
1: Input: $D = D_A \cup \{S_B\}$
2: Output: $S_A$ (a set of support points)
3: project points in $U_A$ onto $\partial\mathcal{P}_A$;
4: $E$ := weighted median edge of $\partial\mathcal{P}_A$;
5: $h_A$ := classifier on $D$ parallel to edge $E$;
6: $S_A$ := support points of $h_A$;
```

We mention the number of rounds required by ITERATIVESUPPORTS to converge, with the detailed proof deferred to a full version.

Theorem 4.1. *The 2-player two-way protocol for linear separators always terminates in at most $O(\log(1/\varepsilon))$ rounds, using at most $O(\log(1/\varepsilon))$ communication.*

5 *Multiparty*

In the noiseless setting, extending from a two-party protocol to a k-party (where data is distributed to k disjoint nodes) can be achieved by allowing an additional factor k or k^2 communication, depending on the hypothesis class.

5.1 One-way Protocols

For k-players one-way protocols pre-determine an ordering among players $P_1 < P_2 < \ldots < P_k$, and all communication goes from P_i to P_{i+1} for $i \in [1, k-1]$. In this section, we show that for k-players, ε-error classifiers can be achieved even with this restricted communication pattern. All discussed protocols can also be transformed into hierarchical one-way protocols that may have certain advantages in latency, or where all nodes just send information one-way to a predetermined coordinator node.

Sampling results for k-players. In sampling-based protocols, along the chain of players, player P_i maintains a random sample R_i of size $O((\nu/\varepsilon)\log(\nu/\varepsilon))$ from $\bigcup_{j=1}^{i} D_i$ and the total size $m_i = \sum_{j=1}^{i} |D_i|$. This can be easily achieved with reservoir sampling [Vitter, 1985]. The final player P_k computes and returns a 0-error classifier on $R_{k-1} \cup D_k$.

Theorem 5.1. *Consider any family of hypothesis $(\mathbb{R}^d, \mathcal{A})$ that has VC-dimension ν. Then there exists a one-way k-player protocol using $O(k(\nu/\varepsilon)\log(\nu/\varepsilon))$ total communication that achieves ε-error, with constant probability.*

Proof. The final set R_{k-1} is an ε-net, so any 0-error classifier on R_{k-1}, is an ε-error classifier on $\bigcup_{j=1}^{k-1} D_i$. So since the total number of points misclassified is at most $\sum_{j=1}^{k-1} \varepsilon|D_j| \leq \varepsilon|D|$, this achieves the proper error bound. The communication cost follows by definition of the protocol. □

0-Error protocols for k-players. Any 0-error one-way protocol extends directly from 2-player to k-players. This requires that each player can send exactly the subset of the family of classifiers that permit 0 error to the next player in the sequence. This chain of players only refines this subset, so by our noiseless assumption that there exists some 0-error classifier, the final player can produce a classifier that has 0-error on all data.

Theorem 5.2. *In the noiseless setting, any one-way two-player 0-error protocol of communication complexity C extended to a one-way k-player 0-error protocol with $O(Ck)$ communication complexity.*

This implies that k-players can execute a one-way 0-error protocol for axis-aligned rectangles with $O(dk)$ communication. Classifiers from the families of thresholds and intervals follow as special case.

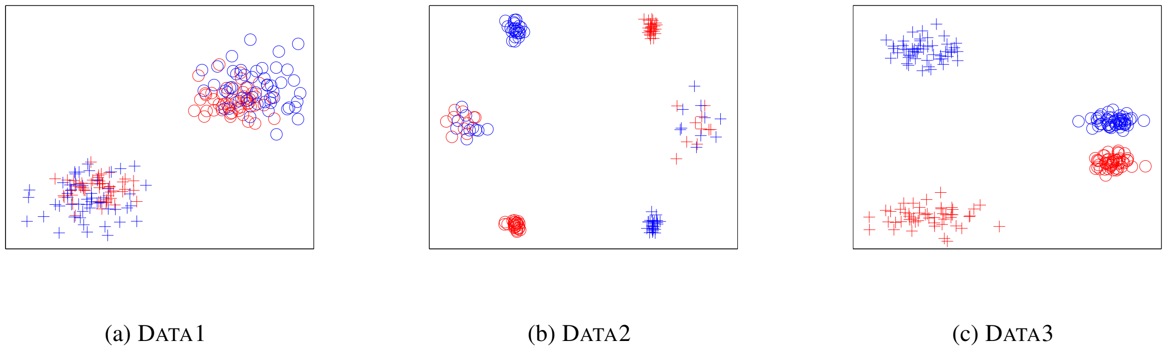

(a) DATA1 (b) DATA2 (c) DATA3

Figure 3: Red represents A and blue represents B. Positive and negative examples (for all datasets) are denoted by '$+$'s and '$\circ$'s, respectively.

5.2 Two-way Protocols

When not restricted to one-way protocols, we assume all players take turns talking to each other in some preconceived or centrally organized fashion. This fits within standard techniques of organizing communication among many nodes that prevents transmission interference.

Linear separators in $\mathbb{R}^2$ with k players. We proceed in a series of epochs. In each epoch, each player takes one turn as coordinator. On its turn as coordinator, player P_i plays one round of the 2-player protocol with each other player. That is, it sends out its proposed support points, and each other player responds with either early termination or an alternative set of support points, including at least one that "violates" the family of linear separators proposed by the coordinator. The protocol terminates if all non-coordinators agree to terminate early and their proposed family of linear separators all intersect. Note that even if all other players may want to terminate early, they might not agree on a single linear separator along the proposed direction; but by replying with a modified set of support points, they will designate a range, and the manner in which these ranges fail to intersect will indicate to the coordinator a "direction" to turn.

Theorem 5.3. *In the noiseless setting, k-parties can find an ε-error classifier over halfspaces in $\mathbb{R}^2$ in $O(k^2 \log(1/\varepsilon))$ communication.*

Proof. Each epoch requires $O(k^2)$ communication; each of k players uses a turn to communicate a constant number of bits with each of k other players. We now just need to argue that the algorithm must terminate in at most $O(\log(1/\varepsilon))$ epochs.

We do so by showing that each player decreases its region of uncertainty by at least half for each turn it spends at coordinator, or it succeeds in finding a global separating half space and terminates. If any non-coordinator does not terminate early, it rules out at least half of the coordinator's points in the region of uncertainty since by Lemma **??**, the coordinator's broadcasted support points represent the median of its uncertain points. If all non-coordinators agree on the proposed direction, and return a range of offsets that intersect, then the coordinator terminates the algorithm and can declare victory, since the sum of all error must be at most $\sum_i \varepsilon|D_i| \leq \varepsilon|D|$ in that range.

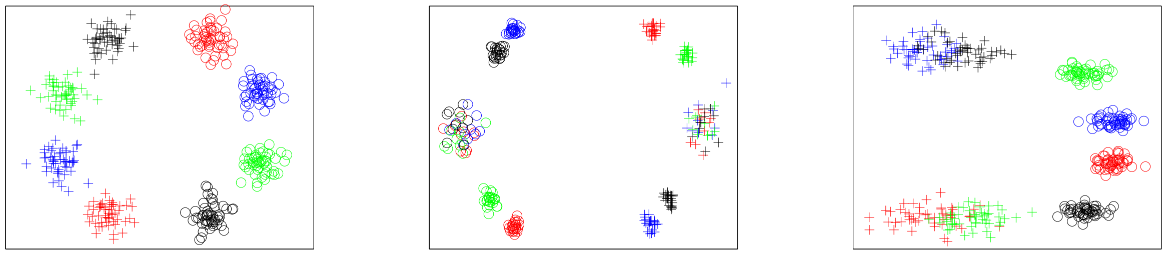
The difficult part is when all non-coordinators individually want to terminate early, but the range of acceptable offsets along the proposed normal direction of the linear separator do not globally intersect. This corresponds to the rightmost picture in Figure 2 where the direction is forced clockwise or counter-clockwise because a negative point from one non-coordinator is "above" the positive point from a separate non-coordinator. The combination of these points thus allow the coordinator to prune half of its region of uncertainty just as if a single non-coordinator did not terminate early. $\square$

6 Experiments

In this section, we present results to empirically demonstrate the correctness and convergence of ITERATIVESUPPORTS.

6.1 Two-Party Results

For the two-party results, we empirically compare the following methods: (a) NAIVE- a naive approach that sends all points in A to B and then learns at B, (b) VOTING- a simple voting strategy that uses the majority voting rule to combine the predictions of h_A and h_B on $D = D_A \cup D_B$; ties are broken by choosing the label whose prediction has higher confidence, (c) RANDOM- A sends a random sample (an ε-net S_A of size $(d/\varepsilon)\log(d/\varepsilon)$) of D_A to B and B learns on $D_B \cup S_A$, (d) MAXMARG- ITERATIVESUPPORTS that selects informative points heuristically (ref. Section 4), and (e) MEDIAN- ITERATIVESUPPORTS that selects informative points with convergence guarantees (ref. Section 4). SVM was used as the underlying classifier for all aforementioned approaches. In all cases, the errors are reported on the dataset D with an ε value of 0.05 (where applicable).



(a) DATA1 (b) DATA2 (c) DATA3

Figure 4: Red represents A, blue represents B, green represents C and black represents D. Positive and negative examples (for all datasets) are denoted by '+'s and '∘'s, respectively.

The above methods have been evaluated on three synthetically generated datasets (DATA1, DATA2, DATA3). For all datasets, both A and B contain 500 data points each (250 positive and 250 negative). Figure 3 pictorially depicts the data.

Method	DATA1		DATA2		DATA3	
	Acc	Cost	Acc	Cost	Acc	Cost
NAIVE	100%	500	100%	500	100%	500
VOTING	100%	500	100%	500	50%	500
RANDOM	100%	65	100%	65	99.62%	65
MAXMARG	**100%**	**4**	**100%**	**4**	**100%**	**12**
MEDIAN	**100%**	**6**	**100%**	**6**	**100%**	**10**

Table 2: Accuracy (Acc) and communication cost (Cost) of different methods for 2-*dimensional* noiseless datasets.

Table 2 compares the accuracies and communication costs of the aforementioned methods for the dataset in 2-dimensions. For all datasets, MAXMARG and MEDIAN required the least amount of communication to learn an optimal classifier. For cases when it is easy to separate the positive from the negative samples (e.g. DATA1 and DATA2) MAXMARG converges faster than MEDIAN. However, DATA3 show that there exists difficult datasets where MEDIAN requires less communication than MAXMARG. This reinforces our theoretical convergence claims for MEDIAN that hold for *any* input dataset. DATA3 in Table 2 shows that there exists cases when both VOTING and RANDOM perform worse than MEDIAN and with a much higher communication overhead; for DATA3, VOTING performs as bad as random guessing. Finally, neither VOTING nor MAXMARG provide any provable error guarantees.

Method	DATA1		DATA2		DATA3	
	Acc	Cost	Acc	Cost	Acc	Cost
NAIVE	100%	500	100%	500	100%	500
VOTING	100%	500	100%	500	81.8%	500
RANDOM	100%	100	100%	100	99.1%	100
MAXMARG	**100%**	**4**	**100%**	**4**	**98.27%**	**40**

Table 3: Accuracy (Acc) and communication cost (Cost) of different methods for 10-*dimensional* noiseless datasets.

Table 3 presents results for DATA1, DATA2, DATA3 extended to dimension = 10. As can be seen, our proposed heuristic MAXMARG outperforms all other baselines in terms communication cost while having comparable accuracies.

6.2 k-Party Results

The aforementioned methods have been appropriately modified for the multiparty scenario. For NAIVE, VOTING and RANDOM, a node is fixed as the *coordinator* and the remaining $(k-1)$ nodes send their information to the coordinator node which aggregates all the received information. For MAXMARG and MEDIAN, in each epoch, one of the k-players takes a turn to act as the *coordinator* and updates its state by receiving information from each of the remaining $(k-1)$ nodes. We experiment with a k value of 4 (i.e., four nodes A,B,C,D). As earlier, for all datasets each of A,B,C,D, contain 500 examples (250 positive and 250 negative). The datasets are shown in Figure 4.

Method	DATA1		DATA2		DATA3	
	Acc	Cost	Acc	Cost	Acc	Cost
NAIVE	100%	1500	100%	1500	100%	1500
VOTING	98.75%	1500	100%	1500	50%	1500
RANDOM	100%	195	100%	195	99.76%	195
MAXMARG	**97.61%**	**14**	**100%**	**2**	**97.38%**	**38**
MEDIAN	**99.0%**	**36**	**100%**	**6**	**98.75%**	**29**

Table 4: Accuracy (Acc) and communication cost (Cost) of different methods for 2-*dimensional* noiseless datasets.

As shown in Table 4, for the k-party case, ITERATIVESUPPORTS substantially outperforms the baselines on all datasets. As earlier, for the difficult dataset DATA3, MEDIAN incurs less communication cost as compared to MAXMARG. We observed that for DATA1 and DATA2, both MAXMARG and MEDIAN require the same number of iterations to converge. However, the cost for MEDIAN is higher due to its quadratic dependency on k. One of our future goals is to get rid of an extra k factor and reduce the dependency from quadratic to linear in k (ref. Section 7.2).

7 Discussion

This paper introduces the problem of learning classifiers across distributed data where the communication between datasets is the bottleneck to be optimized. This model focus on real-world communication bottlenecks is increasingly prevalent for massive distributed datasets. In addition, this paper identifies several very general solutions within this framework and introduces new techniques which provide provable exponential improvement by harnessing two-way communication.

7.1 Comparison with Related Approaches

As mentioned earlier, techniques like classifier *voting* [Bauer and Kohavi, 1999] and *mixing* [McDonald et al., 2010, Mann et al., 2009] are often used in a distributed setting to obtain global classifiers. Interestingly, we have shown that if the different classifiers are restricted to train on disjoint data subsets then there exist specific examples (under the adversarial model) where voting will *always* yield sub-optimal results. We have presented such examples in Section 6. Additionally, parameter mixing (or averaging [Collins, 2002]), which has been primarily proposed for maximum entropy (MaxEnt) models [Mann et al., 2009] and structured perceptrons [McDonald et al., 2010, Collins, 2002], have shown to admit convergence results but lack any bounds on the communication. Indeed, parameter-mixing for structured perceptrons uses an iterative strategy that performs a large amount of communication.

The body of literature that lies closest to our proposed model relates to prior work on label compression bounds [Floyd and Warmuth, 1995, Helmbold and Warmuth, 1995]. In the label compression model, both A and B have the same data but only A knows the labels. The goal is to efficiently communicate labels from A to B. Whereas in our model, each player (A and B) have "disjoint labeled" datasets and the goal is to efficiently communicate so as to learn a combined final ε-optimal classifier on $D_A \cup D_B$. Indeed some of our *one-way* results derive bounds similar to the cited work, as they all build on the theory of ε-nets. In particular, there exists a label compression method [Helmbold and Warmuth, 1995] based on boosting, which gives $O(\log 1/\varepsilon)$ size set for *any concept* that can be represented as a majority vote over a fixed number of concepts. However, in our model, we show that for certain concept classes (with *one-way* communication) we need a linear amount of communication (ref. Theorem 3.3). Furthermore, we demonstrate that using a *two-way* communication model can provide an exponential improvement (ref. Theorem 4.1) in communication cost.

7.2 Future Extentions

Although we have provided many core techniques for designing protocols for minimizing communication in learning classifiers on distributed data, still many intriguing extensions remain. Thus we conclude by outlining three important directions to extend this work and provide outlines of how one might proceed.

Higher dimensions. We provide several results for high-dimensions: for axis-aligned rectangles, for bounded VC-dimension families of classifiers, and a heuristic for linear separators. But for the most common high-dimensional setting–SVMs computing linear separators on data lifted to a high-dimensional feature space–our results either have polynomial dependence on $1/\varepsilon$ or have no guarantees. For this setting, it would be ideal to extend out MEDIAN routine which requires $O(\log 1/\varepsilon)$ communication in $\mathbb{R}^2$ to work in $\mathbb{R}^d$. The key insight required is extending our choice of a *median point* to higher dimensions. Unfortunately, the natural geometric generalization - a *centerpoint* - does not provide the desired properties, but we are hopeful that a clever analysis of a constant size *net* or *cutting* of the space of linear separators will provide the desired bounds.

Noisy setting. Most of the results presented in this paper generalize to noisy data. In fact, Theorem 2.1 and Theorem 3.1 have straight-forward extensions to the noisy case by an ε-sample argument [Har-peled, 2011]. This would increase the communication from $O(1/\varepsilon)$ to about $O(1/\varepsilon^2)$. It would of course be better to use communication only logarithmic in $1/\varepsilon$. We suggest modifying ITERATIVESUPPORTS to work with noisy data with the following heuristic, and defer any formal analysis. In implementing SUPPORT (based either on MAXMARG or MEDIAN) we suggest sending over support points of linear separators that allow for classifiers with exactly ε-error. That is, players never propose classifiers with 0-error, even if one exists; or at least they provide margins on classifiers allowing ε-error. This would seem to describe the proper family of classifiers tolerating ε-error of which we seek to find an example.

Efficient two-way k-party protocols. All simple one-way protocols we present generalize naturally and efficiently to k-players; that is, with only a factor k increase in communication. In fact, a distributed random sample of size $t = O((\nu/\varepsilon)\log(\nu/\varepsilon))$ can be drawn with only $O(t+k)$ communication [Huang et al., 2011], so under a different two-way *coordinator model* some results for the one-way *chain model* we study could immediately be improved. However, again it would be preferable to achieve protocols for linear separators with communication linear in k and logarithmic in $1/\varepsilon$; our protocols are quadratic in k. In particular, our protocol seems slightly wasteful in that each player is essentially analyzing its improvements independently of improvements obtained by the other players. To improve the quadratic to linear dependence on k, we would need to coordinate this analysis (and potentially the protocol) to show that the joint space of linear separators must decrease by a constant factor for each player's turn as coordinator, at least in expectation.

Acknowledgements

This work was sponsored in part by the NSF grants CCF-0953066 and CCF-0841185 and in part by the DARPA CSSG grant N11AP20022. This work was also partially supported by the sub-award CIF-A-32 to the University of Utah under NSF award 1019343 to CRA. All the authors gratefully acknowledge the support of the grants. Any opinions, findings, and conclusion or recommendation expressed in this material are those of the author(s) and do not necessarily reflect the view of the funding agencies or the U.S. government.

References

http://metaoptimize.com/qa/questions/1885/suppose-your-training-and-test-set-are-generated-by-a-cunning-adversary, 2010.

Martin Anthony and Peter L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, New York, NY, USA, 1st edition, 2009.

Eric Bauer and Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36(1-2), 1999.

Ron Bekkerman, Mikhail Bilenko, and John Langford. Scaling up machine learning: Parallel and distributed approaches. www.cs.umass.edu/ ronb/scaling_up_machine_learning.htm, 2011.

Nicolò Cesa-Bianchi, Claudio Gentile, and Francesco Orabona. Robust bounds for classification via selective sampling. In *ICML*, Montreal, Canada, 2009.

Michael Collins. Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In *EMNLP*, Stroudsburg, USA, 2002.

Ofer Dekel, Claudio Gentile, and Karthik Sridharan. Robust selective sampling from single and multiple teachers. In *COLT*, Haifa, Israel, 2010.

Sally Floyd and Manfred Warmuth. Sample compression, learnability, and the vapnik-chervonenkis dimension. *Machine Learning*, 50:269–304, 1995.

Sariel Har-peled. *Geometric Approximation Algorithms*. American Mathematical Society, 2011. ISBN 0-8218-4911-5.

David P. Helmbold and Manfred K. Warmuth. On weak learning. *Journal of Computer and System Sciences*, 50: 551–573, 1995.

Daniel Hsu and John Langford. The end of the beginning of active learning. http://hunch.net/?p=1800, 2011.

Zengfeng Huang, Ke Yi, Yunhao Liu, and Guihai Chen. Optimal sampling algorithms for frequency estimation in distributed data. In *The 30th IEEE International Conference on Computer Communications*, 2011.

Michael Kearns and Umesh Vazirani. *An introduction to computational learning theory*. MIT Press, Cambridge, MA, USA, 1994. ISBN 0262111934.

Pavel Laskov and Richard Lippmann. Machine learning in adversarial environments. *Machine Learning*, 81(2), 2010.

Aleksandar Lazarevic and Zoran Obradovic. The distributed boosting algorithm. In *KDD*, San Francisco, USA, 2001.

Gideon Mann, Ryan McDonald, Mehryar Mohri, Nathan Silberman, and Dan Walker. Efficient large-scale distributed training of conditional maximum entropy models. In *NIPS*, Vancouver, Canada, 2009.

Ryan McDonald, Keith Hall, and Gideon Mann. Distributed training strategies for the structured perceptron. In *NAACL HLT*, Los Angeles, California, 2010.

Joel B. Predd, Sanjeev R. Kulkarni, and H. Vincent Poor. Distributed learning in wireless sensor networks. *IEEE Signal Processing Magazine*, 2006.

Burr Settles. Active learning literature survey. In *Computer Sciences Technical Report 1648*, University of Wisconsin-Madison, 2009.

Jeffrey Scott Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software*, 11:37–57, 1985.