Distributed Decision Tree Learning in Peer-to-Peer Environments using a Randomized Approach

Submitted for Blind Review

Abstract

Learning decision trees from distributed data has been a challenge since straightforward distributed implementation of popular centralized decision tree learning algorithms suffer from high communication cost resulting from the greedy selection of attributes at every stage of the tree. This paper offers an alternate communication-efficient randomized technique that is appropriate for large asynchronous peer-to-peer (P2P) networks. It first presents a brief overview of a P2P distributed data mining system called PADMini that can be used for P2P classifier learning among others. The proposed distributed randomized decision tree learning algorithm is then presented in the context of this system and the related applications. The paper offers a set of analytical results regarding the properties of randomized decision trees used in this paper following the construction proposed elsewhere [8]. The paper also presents a detailed description of the distributed algorithm and extensive experimental results.

Keywords Random Decision Tree, Random Forest, Boosting, Bagging, P2P

1 Introduction

Distributed classifier learning from data stored at different locations is an important problem with many applications. Large peer-to-peer networks are creating many such applications where centralized data collection and subsequent learning of classifiers are extremely difficult, if not impossible. Decision trees [5] are popular classifier learning techniques widely used for many applications. Distributed versions of decision tree learning algorithms are therefore of high interest. However, most decision tree learning algorithms work by selecting an attribute in a greedy manner at every stage of the tree construction. This usually results in synchronized construction of the tree by exchanging the information stored at different nodes. This causes high communication overhead and synchronous algorithms that do not scale very well in large distributed environments like the P2P networks.

This paper motivates the current work by introducing the readers with the PADMini system that offers distributed classifier learning among other capabilities over P2P networks. The paper poses the distributed decision tree learning problem in that context. In order to avoid the root cause of synchronized communication, the paper explores a class of randomized decision trees introduced elsewhere [8] that do not require greedy selection of the attributes at every stage of the tree-construction. It investigates analytical properties of those trees and adapts those to design a distributed asynchronous decision tree learning algorithm for P2P networks.

The paper is organized as follows. Section 2 describes the PADMini system in order to motivate the current work. Section 3 reviews the existing literature. Section 4 defines the distributed decision tree learning problem. Section 5 explores various analytical properties of randomized decision trees. Section 6 presents a detailed description of the distributed decision tree learning algorithm. Section 7 offers the experimental results. Finally, section 8 concludes this paper.

2 Motivation

Popular internet document repositories store large amount of unstructured texts and image data that are frequently accessed by a large number of users. Users' input through collaborative tagging have proven to be very useful in classifying such online documents. A web-user interested to participate in this collaborative tagging process can be thought of a node in the P2P network (the internet). In such a P2P network, the nodes have their own (local) training dataset. In order to learn a global classifier based on all the training data on every node, one possible but not a wise way is to transfer all the training data to a central server and let the server to learn the classifier. The communication cost for transferring the data and single point failure are the reasons for it. This motivates us to develop some distributed classifier learning algorithms without centralizing the training data.

There are existing P2P data mining systems that implement such collaborative text classifiers. For example, a collaborative TagLearner has been implemented on top of the P2P distributed data mining system **PADMini**, where the underlying classifier is an LP classifier [7]. However, the distributed simplex algorithm used for classification sometimes may have issues with convergence, since the worst-case complexity of the simplex algorithm is exponential time [11]. Also, currently the **PADMini** LP based TagLearner supports only binary classification. In this paper, we shall focus on the P2P TagLearner implemented on the same **PADMini** system, but using a different P2P classifier, which is implemented using the distributed random decision tree (DRDT) learning algorithm proposed in this paper (as an extension of RDT algorithm by [8]). The DRDT classifier learnt is an n-class classifier and the P2P learning is fully asynchronous in nature, hence faster. The underlying peer-to-peer network of **PADMini** system is implemented using the Distributed Data Mining Toolkit (**DDMT**) forms the backbone of the computation network using which it is possible to implement any P2P distributed data mining algorithm in our system.

A web-user interested in collaborative tagging has to download and install the Firefox plugin provided on the **PADMini** website. The plugin allows the user to login to the system, join a tagging group and label texts using the tags that are currently supported by the tagging group and then it converts the user-labeled text data to a set of feature vectors using the set of features provided by the tagging group. When the system has sufficient inputs from users, it automatically triggers the underlying P2P classifier learning algorithm.

Furthermore, if we consider the online document repository as a huge training data set, web user's tagging process can be viewed as (virtual) random sampling (since each document is equally likely to be selected) from the large training data set. Hence, every web user will have a subset of the training dataset by randomly sampling, which is required for DRDT.

3 Related Work

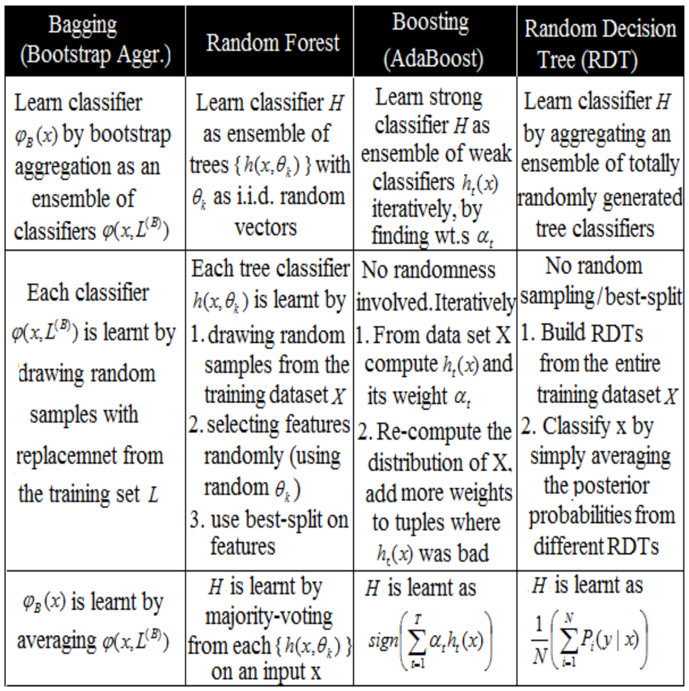
Computing a decision tree in such large distributed systems using standard centralized algorithms can be very communication-expensive and impractical because of the synchronization requirements. A P2P distributed scalable decision tree induction technique was proposed as an alternative in [2] that works in a completely asynchronous manner in distributed environments and offers low communication overhead, which uses P2P misclassification error minimization technique for greedy selection of features, based on the impurity measure. However, this is a distributed extension of ID3 [13] and C4.5 [5] like single greedy decision induction technique and in this paper we shall concentrate on distributed learning of ensemble tree classifier, by generating a set of completely random decision trees [8], thereby completely getting rid of the greedy feature choice, the one that causes major bottleneck in communication. A number of "multiple random decision tree approaches" have been developed since mid 90's. Multiple decision trees are frequently referred to as "ensemble" classifiers and the different techniques proposed were "bagging" [3], "boosting" [9], "random forests" [4]. Amit et. al. [1] described these methods as "holographic" methods because each data point in the analysis can be resampled many times, under different circumstances and therefore have the ability to contribute to the shaping of a multi-dimensional view of the data. In bagging, the approach is typically to form different random subsamples of the source data. In boosting, the approach is normally to re-weight the contribution of subsampled observations. Observations are reweighted based on how frequently they have been misclassified in previous runs in the multi-tree sequence. Observations that have been poorly classified are given more weight than those that have been well classified. On the contrary, Random forests are a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest [4]. Fan et. al. proposed a completely random decision tree algorithm achieving much higher accuracy than the single best hypothesis, with the advantages being its training efficiency as well as minimal memory requirement [8]. The following figure 1 compares the different techniques proposed for ensemble based learning.

We note that the random decision tree model resembles the random forest model with only difference between two being that the forest technique uses random samples to learn the decision trees but the one proposed by Fan et. al. learn the random decision trees from the entire data (no sampling is done), also the random forest technique selects the features randomly at once by choosing a random vector, while the random decision tree picks up the features iteratively.

4 Problem Definition: The Distributed Classifier Learning Problem

We have the dataset D (the population), with $|D|$ tuples, each tuple with $n = |F|$ features F= $\{X_1, \ldots, X_n\}$ and target class $Y \in C = \{c_1, \ldots c_{|C|}\}$. In the P2P environment, we have N nodes $\aleph_1 \ldots \aleph_N$, node i with dataset $D_i,\ i = 1 \ldots N$, where each $D_i \subseteq D$ is chosen by selecting $|D_i| = N_{train}$ random samples (with replacement) from the population D, each of them will serve as training data in the corresponding nodes.

Now, for a given $k,\ 1 < k < n$, at each node D_i, we shall learn a decision tree $T_i,\ i = 1 \ldots N$ of height k locally, by choosing the features totally randomly by using the RDT learning technique proposed in [8]. Finally, given any test dataset $D_{test} \subseteq D$ and a test tuple $x \in D_{test}$ (with unknown target class), we shall send x to each node and pre-



Bagging (Bootstrap Aggr.)	Random Forest	Boosting (AdaBoost)	Random Decision Tree (RDT)
Learn classifier $\varphi_B(x)$ by bootstrap aggregation as an ensemble of classifiers $\varphi(x, L^{(B)})$	Learn classifier H as ensemble of trees $\{h(x,\theta_k)\}$ with θ_k as i.i.d. random vectors	Learn strong classifier H as ensemble of weak classifiers $h_t(x)$ iteratively, by finding wt.s α_t	Learn classifier H by aggregating an ensemble of totally randomly generated tree classifiers
Each classifier $\varphi(x, L^{(B)})$ is learnt by drawing random samples with replacemnet from the training set L	Each tree classifier $h(x,\theta_k)$ is learnt by 1. drawing random samples from the training dataset X 2. selecting features randomly (using random θ_k) 3. use best-split on features	No randomness involved. Iteratively 1. From data set X compute $h_t(x)$ and its weight α_t 2. Re-compute the distribution of X. add more weights to tuples where $h_t(x)$ was bad	No random sampling/best-split 1. Build RDTs from the entire training dataset X 2. Classify x by simply averaging the posterior probabilities from different RDTs
$\varphi_B(x)$ is learnt by averaging $\varphi(x, L^{(B)})$	H is learnt by majority-voting from each $\{h(x,\theta_k)\}$ on an input x	H is learnt as $sign\left(\sum_{t=1}^{T}\alpha_t h_t(x)\right)$	H is learnt as $\frac{1}{N}\left(\sum_{i=1}^{N}P_i(y\mid x)\right)$

Figure 1. Different ensemble-based classification techniques using random decision trees

dict the target class label of x by the ensemble of the RDT classifiers $\{T_1 \ldots T_N\}$, by simply computing an average (by P2P distributed asynchronous average) of the posterior class probability distribution and choosing the class with maximum average probability, by $\underset{j \in C}{argmax}\frac{1}{N}\sum_{i=1}^{N} P_i(y_j|x)$.

Table 4 summarizes the terminology used in this paper.

Symbols	Definitions
RDT	Random Decision Tree
$N_R(k)$	Number of all possible RDTs of height k
N	Number of RDTs to be generated = number of nodes for distributed version
$T_1, \ldots T_N$	Generated RDTs

5 Randomized Decision Tree Ensembles

As discussed before, there are different existing randomization techniques for ensemble based learning 1. For instance, BootStrap Aggregation [3] uses random sampling with replacement to setup the training data for each of the classifier. Random Forest [4] technique uses yet another randomization on top of it, in the form of independently chosen random vectors with same distribution and the individual trees are constructed by choosing features from this vector. Breiman showed that Bagging always improves the accuracy of the individual classifier if the individual classifiers are not stable. Another important result by Breiman is to show that these ensemble based classifiers never overfit, rather the generalization error approaches to a limit if we add more and more trees (contrary to Occam's razor). In a generalized setting, Breiman also showed that the generalization error of the random forest ensemble classifier depends upon the individual classifier strengths and the correlation of the raw margin functions [4]. In the following section we theoretically investigate the RDT technique proposed by [8], where we try to answer the question: why ensemble of purely random decision trees (RDT) gives high accuracy and low generalization error. For the sake of simplicity we present the analysis for decision trees with binary features and binary target class labels ($f_T : \{0, 1\}^n \rightarrow 0, 1$), but it can be easily extended to the generic case.

5.1 Accuracy (Strength) of the individual RDT classifiers

Let us consider random (binary) decision trees of depth k. Total number of possible such random decision trees

$$= N_R(k) = \binom{n}{1} \times \binom{n-1}{2}.2! \times \binom{n-1-2}{2^2}.(2^2)! \times \ldots$$
$$\ldots \times \binom{n-1-2-\ldots-2^{k-1}}{2^k}.(2^k!)$$
$$= \prod_{i=0}^{k} \binom{n-2^i+1}{2^i} = \frac{n!}{(n-2^{k+1}+1)!}$$

which form the population of all possible RDTs of height k.

Now, let us concentrate on the accuracy of the individual RDT classifiers. As in [3], let us denote an RDT predictor T predicting the target class $j \in \{0, 1\}$ (equivalently $+$ and $-$ target class labels) of a test tuple x by $\phi(x, T)$. Similarly define $Q(j|x) = P(\phi(x, T) = j)$ and the event $C_x = \bigcup_{j=0}^{1} ((\phi(x, T) = j) \wedge$ (target class for x is j)), denoting the event that the predictor T classifies x correctly, so that

$$P(C_x) = \sum_j Q(j|x)P(j|x) = \sum_j P(\phi(x, T) = j)P(j|x)$$

Also, let's define the set F_x of **"essential features"** for the prediction of a tuple x as

Definition 1

A minimum (nonempty) subset of features, needed to classify x correctly.

We argue that there always exists such a subset of features (not necessarily unique) for every tuple x, s.t., in order to predict the target class label of the tuple x correctly, we need all off these l features $X_{i_1}, X_{i_2}, \ldots, X_{i_l}$ from these subset F_x to be considered by our classifier (i.e., all of them must be present in at least one path in the random decision tree (as fixed bits)), this l will vary depending upon data (explained in next figure).

Moreover, this set is **"irreducibly essential"** or minimal in the sense that if we do not consider any feature present for classification of x, then x can never be classified correctly by any classifier. As explained in figure 5.1, X_2 is such an essential feature for the given dataset.

Probability that an RDT T_i classifies x correctly

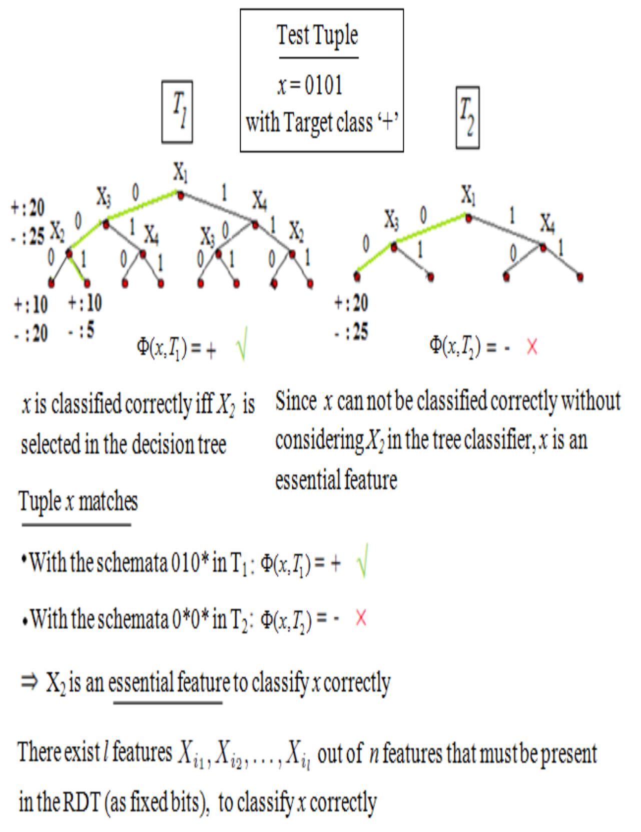

Test Tuple
x = 0101
with Target class '+'
T1
T2
+ : 20
- : 25
+ : 10
- : 20
+ : 10
- : 5
Φ(x,T1) = + √
+ : 20
- : 25
Φ(x,T2) = - X
x is classified correctly iff X2 is selected in the decision tree
Since x can not be classified correctly without considering X2 in the tree classifier, x is an essential feature
Tuple x matches
•With the schemata 010* in T1: Φ(x,T1) = + √
•With the schemata 0*0* in T2: Φ(x,T2) = - X
⇒ X2 is an essential feature to classify x correctly
There exist l features Xi1, Xi2, …, Xil out of n features that must be present in the RDT (as fixed bits), to classify x correctly

Figure 2. Essential Features

$= P(C_x) = P(F_x \in T_i) =$ Probability that all the essential features are present (as a schemata) in a (at least one) path in T_i.

Now, probability that all of the l essential features are chosen in a given path (schemata) in T_i will be $= P(S_i) = \frac{\binom{n-l}{k-l}}{\binom{n}{k}}$.

Hence, the probability that T_i classifies x correctly $= P(C_x) = \bigcup_{i=1}^{2^k} P(S_i) = p$ (let), which is similar to the "strength" of the individual classifiers defined by Breiman et al. [4], following which the strength of the ensemble of RDTs can be defined in terms of the margin function as

$$s = E_{x,y} mrdt(x, y)$$

where

$$mrdt(x, y) = P(T(x) = y) - \max_{j \neq y} P(T(x) = j)$$

where

$$T \in \{T_1, T_2, \ldots, T_N\}, \text{ set of RDT classifiers.}$$

Also, since probability that T_i classifies x correctly $= p$ and wrongly with probability $= q = 1 - p$, we can think of $N_R(k)$ independent Bernoulli trials (one for each RDT, from the population of all possible RDTs of depth k) with probability of success (i.e., probability of correct classification) $= p$.

Now, $P(T(x) = y)$ is the proportion of the trees expected to vote correctly $\Rightarrow P(T(x) = y) = E[I(T(x) = y)] = p$ and $P(T(x) \neq y) = q$, for binary classification, we have

$$mrdt(x, y) = p - q = 2.\left(p - \frac{1}{2}\right)$$

$$s = E_{x,y} mrdt(x, y) = 2.\left(p - \frac{1}{2}\right)$$

It can be easily seen that the strength (s) of the individual RDT classifiers will be high in general (when we have $p > \frac{1}{2}$) and will keep on increasing rapidly as the depth k of the generated RDTs increases.

Additionally, we notice that in Bagging like techniques [3], the individual classifiers are constructed from the random samples taken with replacement from the population, but in case of RDT construction, the entire population is used for the same, hence strength of the individual classifiers must be higher.

5.2 Accuracy of the ensemble classifier (the population mean)

As explained in the earlier section, for a given test tuple x, we can consider predicting the target class of x

an RDT classifier as a Bernoulli trial (a coin toss) with success probability (probability to classify x correctly) as p.

Let Y_i denote the random variable denoting the output of each trial with

$$Y_i = \begin{cases} 1 & x \text{ correctly classified by } T_i \\ 0 & x \text{ wrongly classified by } T_i \end{cases}$$

Since each Y_i follows Bernoulli distribution with success probability p, $Y = \sum_{i=1}^{N} Y_i \sim B(N,p)$ follows a Binomial distribution (population size = number of RDTs of depth k is N), with (population) mean Np.

The aggregate predictor (for the ensemble of the RDT classifiers) can be computed based on majority voting as [3]

$$\phi_A(x) = \underset{j}{argmax}\; Q(j|x)$$

or equivalently by choosing the class label corresponding to the maximum average posterior probabilities given x (for all N RDTs in the population of all possible depth k binary decision trees) as (population mean) [8]

$$\underset{j}{argmax} \frac{1}{N}\left(\sum_{i=1}^{N} P_i(y_j|x)\right)$$

The aggregate classifier gives the correct output iff majority ($\lfloor \frac{N}{2} \rfloor + 1$) of the N individual binary classifiers agree on their vote for the target class label for the test tuple x correctly.

Hence, the aggregate classifier gives the correct output with probability

$$\begin{aligned} &= P(Y \geq \left\lfloor \frac{N}{2} \right\rfloor + 1) \\ &= 1 - P(Y \leq \left\lfloor \frac{N}{2} \right\rfloor + 1) \\ &= 1 - F\left(\left\lfloor \frac{N}{2} \right\rfloor + 1; N, p\right) \\ &\geq 1 - \exp\left(-\frac{1}{2p}\frac{\left(Np - \lfloor \frac{N}{2} \rfloor + 1\right)^2}{N}\right) \\ &= 1 - p_e \end{aligned}$$

where $F(k;N,p) \leq \exp\left(-\frac{1}{2p}\frac{(Np-k)^2}{N}\right)$ by Chernoff's bound.

Since p_e is likely to be small we have a high accuracy for the aggregate classifier and hence taking the true "population mean" of all the RDTs explains the reason for the high accuracy of the ensemble classifier.

5.3 Accuracy of the ensemble classifier (the sample mean)

The space of all possible RDTs of depth k is exponential, not all of them can be chosen, hence only N from $N_R(k)$ RDTs (the set of all possible RDTs of depth k) are chosen as individual classifiers (sample size N).

As explained in [8], the construction of the purely random decision trees can be thought of random sampling of RDTs from the population of all possible RDTs of size $N_R(k)$ and by Strong Law of Large Numbers (SLLN) we know that sample aggregate $\hat{\phi}(A)$ is going to converge to the population aggregate $\phi(A)$, a.s.

Also, by applying the Central Limit Theorem (CLT) [8], we can argue that class label predicted by the sample aggregate ensemble classifier is going to approach the class label predicted by the population aggregate classifier, the later is able to predict the target class with high accuracy, as we have already shown.

The population mean posterior class distribution vector is obtained by averaging the posterior class distributions of all $N_R(k)$ possible RDTs of depth k by $\mu = \frac{1}{N_R(k)} \sum_{i=1}^{N_R(k)} P_i(y|x)$. Similarly, the sample mean posterior class distribution obtained by averaging the posterior class distributions of the N chosen depth-k RDT samples from the depth-k RDTs' population by $\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} P_i(y|x)$.

Now, since the sample RDTs are independently chosen, by the Central Limit Theorem, we have

$$\left(\frac{\mu - \hat{\mu}}{\sigma/\sqrt{N}}\right) \xrightarrow{D} \aleph(0,1)$$

$$P\left(\left|\frac{\mu - \hat{\mu}}{\sigma/\sqrt{N}}\right| < \epsilon\right) = 1 - 2\Phi\left(\frac{\mu - \hat{\mu}}{\sigma/\sqrt{N}}\right)$$

Also, by Chebyshev's inequality, we have

$$\Pr\left(|\mu - \hat{\mu}| \geq \frac{\sigma}{\sqrt{N}}\right) \leq N$$

.

These inequalities can be used to find minimum number (N_{min}) of RDTs required to be generated to ensure that sample mean does not deviate much from the population mean, as described in the next section.

5.4 Number of RDTs to be generated

5.4.1 Number (N_{min}) of RDTs required for the ensemble classifier to produce at most ϵ error in classification

As shown earlier, the aggregate classifier classifies a test tuple x correctly with probability at least $1 - p_e$ (can be lower bounded), hence it classifies x wrongly with at most $\epsilon = p_e$ probability. If we want to choose N RDTs, we shall have $Y \sim B(N,p)$ and $\exp\left(-\frac{1}{2p}\frac{\left(Np-\lfloor\frac{N}{2}\rfloor+1\right)^2}{N}\right) \leq \epsilon$. Hence, given ϵ, we can solve for N by

$$\left(Np - \left\lfloor\frac{N}{2}\right\rfloor + 1\right)^2 \geq 2pN.log_e\,\epsilon$$

the least number (N_{min}) of RDTs required to classify x wrongly with probability at most ϵ.

5.4.2 Representing the decision trees: the "sufficient representation" and the "fixed bit" heuristic

Since there are n different features and each can have 3 possible values in a given schema (namely 0, 1 or *), along with 2 different possible target class labels (+ or -, since binary classifier), we have $3^n.2$ different concepts. Also, we need to have at least one name for each concept and no two concepts can share a name, in order to have a "representation" for the class of concepts [12]. Hence we need to have at least $3^n.2$ different names, in order to have a "representation" of the class of concepts, which is exponential in number of features. Now, we shall define sufficient representation by considering the presence of all of n features in the set of N RDTs generated.

Let us first start with an example. Consider an arbitray schemata "$*1*$" for $k = 3$ and also additionally assume that none of the RDTs generated posses a schemata (a path) that has $X_2 = 1$ as a fixed bit (X_2 was never selected while construction of any of the random trees, hence all of them have $X_2 = *$). If the feature X_2 is a noise and does not have a major role in target classification, then the accuracy of our ensemble classifier is not going to change with the presence or absence of this particular feature in any of the schematas presented by all RDTs. But in the average case, if the feature X_2 has contribution in target classification, the accuracy will certainly be affected for some of the test tuples, if we do not have the feature is not present at all in any of the RDTs.

In the above example, the set of RDTs does not have the X_2 at all (as fixed bit) and hence we treat the collection to be an insufficient representation. In other words, we define the "sufficient representation" (this has nothing to do with the traditional machine learning representation defined by [12]) by the notion that for every attribute $X_i,\ i = 1 \ldots n$, the class of N random trees generated must have at least one schemata (path) having a "fixed bit" corresponding to that attribute.

Definition 2

A collection of random decision trees $\{T_i : i = 1 \ldots N\}$ provides a "sufficient representation" of the underlying domain iff for each feature $X_i,\ i = 1 \ldots n$, it contains at least one schemata with a "fixed bit" corresponding to that feature. Also, by 100% confidence in this context we mean that the collection is a totally sufficient representation.

Lemma 1

The minimum number (N_{min}) of random decision trees required to ensure representation sufficiency to a degree of $\delta\%$ confidence is given by $\frac{log\left(\frac{1-\delta}{n}\right)}{log\ p\prime}$, where $p\prime = \left(1-\frac{1}{n}\right)\left(1-\frac{1}{n-1}\right)^2 \ldots \left(1-\frac{1}{n-k+1}\right)^{2^{k-1}}$.

Proof

As seen from figure 5.4.2

$$P(X_i \text{ not selected at level 0}) = \frac{1}{n}$$
$$P(X_i \text{ not selected at level 1}|X_i \text{ not selected at level 0}) = \frac{1}{(n-1)^2}$$
$$P(X_i \text{ not selected at level 2}|X_i \text{ not selected at level 0 and 1}) = \frac{1}{(n-2)^4}$$
$$\ldots$$
$$P(X_i \text{ not selected in the RDT of height } k) = \bigcap_{j=0}^{k-1} P(X_i \text{ not selected at level j})$$
$$= \prod_{j=1}^{k-1} P(X_i \text{ not selected at level j}|X_i \text{ not selected at level 0} \ldots \text{j - 1})$$
$$\times P(X_i \text{ not selected at level 0})$$
$$= \left(1-\frac{1}{n}\right)\left(1-\frac{1}{n-1}\right)^2 \ldots \left(1-\frac{1}{n-k+1}\right)^{2^{k-1}}$$

Probability that a given feature X_i was never selected while building a given random tree

$= p\prime = \left(1-\frac{1}{n}\right)\left(1-\frac{1}{n-1}\right)^2 \ldots \left(1-\frac{1}{n-k+1}\right)^{2^{k-1}}$

Since the trees are constructed independently, probability that a given feature X_i was never selected while building any of the N generated random trees $= p\prime^N$.

Now let's define the event S_i to be the event denoting that the attribute X_i was selected in at least one of the random decision trees generated (and hence present as a "fixed bit" in at least one of the schematas). Hence,

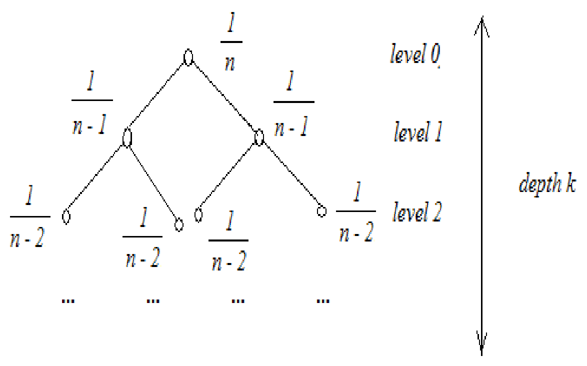


Figure 3. Number of ways for sufficient representation

$P(\bar{S_i}) = p\prime^N$. Also, by our earlier definition of functional completeness, we are interested in lower-bounding the following probability (all attributes are present as "fixed bits" in at least one of the schematas in at least one of the random decision trees):

$$P\left(\bigcap_{i=1}^{n} S_i\right) = 1 - P\left(\bigcup_{i=1}^{n} \bar{S_i}\right) \geq 1 - \sum_{i=1}^{n} P(\bar{S_i}) = 1 - n.p\prime^N,$$

by union bound.

If we call this our confidence δ, then the above inequality gives us a very straightforward formula to find N_{min}, the (minimum) number of random decision trees required to be generated in order to guaranty a $\delta\%$ confidence is

$$N_{min} = \frac{log\left(\frac{1-\delta}{n}\right)}{log\ p\prime}.$$

For instance, if we want $\delta = 0.9 \Rightarrow N_{min} \geq \frac{1+log(n)}{log(1/p\prime)}$, from which we can compute the minimum number of random decision trees required (N_{min}) to ensure representation sufficiency.

5.4.3 The "diversity" heuristic

The generalization error of a forest of tree classifiers depends on the strength of the individual trees in the forest and the correlation between them ([4], theorem 2.3). Since the attributes are chosen randomly (two randomly chosen vectors are orthogonal) and independently, hence they are highly uncorrelated and the error will be upper bound by a small value. This supports the "diversity" heuristic that turned out to be very efficient: if the random decision trees are diverse in nature, the accuracy increases.

Lemma 2

Assuming that each decision tree is chosen in totally random manner (e.g., by tossing coin), the number of expected trees to have N different (to maintain diversity heuristic [8]) random decison trees $= \theta(rlog(r))$, by the randomized Coupon Collector problem.

Proof

Follows directly from coupon collector problem.

5.4.4 Random Decision Trees and Functional Completeness

We note that any decision tree function is nothing but a set of schematas (or equivalent classes). Since the random decision trees generated are iso-height (all of same height $k < n$) and full binary trees of depth k, we have 2^k different schematas (equivalent classes) that represent a single random decision tree (with binary features and binary target class labels). So equivalently our desired target classifier can be represented as $f : \{h_1, h_2, \ldots h_{2^k}\} \rightarrow \{0, 1\}$.

Also, each (partial) schemata consists of any of the 2 fixed values $(1,\ \ 0)$ or a variable value $(*)$ for a given feature. Hence any arbitrary decision tree f can be represented by the linear combination of the N random trees essentially means that any arbitrary schemata can be represented as linear combination of the schematas present in the randomly generated trees. We note that a single tree is enough to represent all possible schemata.

Definition 2

The family of functions $\{f_i\ :\ i = 1 \ldots r\}$ are functionally complete iff they form a linearly independent (basis) set of functions in the space of decision tree functions.

Definition 3

The set of functions $\{f_i\ :\ i = 1 \ldots r\}$ are linearly independent iff $\forall x \in X^n, \sum_{i=1}^{r} \lambda_i.f_i(x) = 0 \Rightarrow \lambda_i = 0,\ \forall i.$

We shall now be interested to answer the following question: how many random decision trees are needed to achieve functional completeness. We note that by function completeness, we mean that any arbitrary decision tree function f should be expressed as a linear combination of the random decision tree function vectors $(f_i(x),\ i = 1 \ldots r)$ only.

Post's Functional Completeness Theorem

This theorem defines 5 different classes C_i for boolean functions $(\beta, \gamma, alternating, A : a, self - dual)$ and states that a set $F = \{f_i | i = 1 \ldots r\}$ of truth functions (a set of N RDTs, which are also boolean functions) will be functionally complete iff $\forall i \in \{1 \ldots 5\},\ \exists f \in F | f \notin C_i$.

$$P(\text{F is functionally complete}) = P(\bigcap_{i=1}^{5} (\exists f \in F | f \notin C_i))$$

$$= \prod_{i=1}^{5} (1 - P(\forall f \in F | f \in C_i)) = \prod_{i=1}^{5}(1 - p_i)$$

Since p_i s are likely to be small, the set of random decision trees generated are likely to be functionally complete.

5.5 Random Decision Trees: Orthogonal?

We can compute the Fourier coefficients w_j od a RDT function f by [10]

$$w_j = \frac{1}{|\wedge|} \sum_{h \in \wedge} f(h) \psi_j(h)$$

As shown in the same paper [10], in a depth k RDT, all the Fourier coefficients of order $\geq k$ are zero.

Also, $w_j \neq 0 \Rightarrow \exists$ a schema $h \in \wedge$ in the RDT s.t. h has a fixed bit in all the positions where w_j has a 1 bit.

Let w be a Fourier Coefficient and $o(w) = l$. Now consider depth k binary RDT that has 2^k different paths. Define $I_{P_i},\ i = 1 \ldots 2^k$ to be the event that the schemata represented by P_i does not fixed bits in all the positions where w_j has set bits.

Hence, $P(I_{P_i}) = 1 - \frac{l}{n} \Rightarrow P(w \neq 0) = P\left(\bigcap_{i=1}^{2^k} I_{P_i}\right) = \left(1 - \frac{l}{n}\right)^{2^k}$, which is very small.

Hence, most of the Fourier coefficients for the RDTs are likely to be zeros, thereby yielding a very low inner-product between any two Fourier-coefficient vectors, hence any two RDTs will be orthogonal to a high degree. Experimental results show the inner product lies in between $0.1 - 0.2$.

5.6 Generalization Error of the Ensemble Classifier

From the analysis of the random forest paper [4], we see that the generalization error

$$PE^* \leq \bar{\rho} \frac{1 - s^2}{s^2}$$

which is dependent upon

1. The mean value of the correlation coefficient between margin functions for the tree classifiers (since the RDTs constructed are of small heights (shallow) [8], intuitively $\bar{\rho}$ should be low) thereby decreasing the lower bound of the generalization error, as shown previously.

2. The strength s of the individual random decision tree classifiers, which is high enough (as per the lower bound proof shown in the earlier section).

As discussed in [4], since the growing depth of the classifiers (individual RDTs) cause increase in the strength and increase in correlation simultaneously, this is a tradeoff. The half-full heuristic proposed by Fan et al. gives good results as explained in [8].

6 DRDT: Distributed Learning of the RDTs

6.1 The Algorithm

The distributed extension of the RDT algorithm is pretty straight-forward. Instead of using the entire training data (population) for the construction of the individual RDTs, we use random samples (with replacement) as in [3], [4] to construct RDTs at different sites (nodes), one RDT per each node. For testing purpose, a testing tuple is sent to all the sites where the probability distribution vector for its class label is predicted by the corresponding RDT. The mean vector of the probability distribution vectors is computed by distributed average computation, in order to obtain the target class label. As shown in 6.1, the distributed RDT learning has the following straightforward steps:

- First randomly select N_{train} samples from the repositories (population), in case of collaborative text tagging this happens when a web-user selects a text to tag using his browser. Also note that this step is different from Fan et al.'s centralized counterpart in the sense that it uses the entire training population to learn each RDT.

- Build and Train each RDT locally at each of the nodes using the local training data.

Algorithm 1 DRDT: The Distributed Random Decision Tree Ensemble Classification

```
1: for each node parallelly do
2:     Randomly (with replacement) select $N$ data tuples from the training data (population).
3:     Locally generate a random decision tree structure of height $k$ by using build RDT technique by Fan et al.
4:     Locally compute leaf node class distribution for the RDT using ComputeStatistics technique by Fan et al.
5: end for
6: Given a test tuple $x$, send $x$ to every node.
7: At each node parallelly compute the posterior class probability distribution vector locally.
8: Asynchronously (using gossip) compute the distributed average posterior class probability distribution vector from the individual posterior class probability distributions, to obtain the target class probability distribution.
9: Output the target class label corresponding to the highest probability in the mean vector computed.
```

- To Test the ensemble classifier, send the test tuple x to all nodes and compute the average posterior class probability distributions in a distributed manner.

6.2 Analysis

Similar to analysis in section 5.

6.3 Weighted Extension to RDTs: Choosing Weights for Random Decision Trees

So far we were only concerned about simple averaging of the posterior class probability distributions generated by the RDTs given a test tuple x, i.e., we have assigned equal weights (importance) to the predictions from each of the individual RDT classifiers. But some of the classifiers may not be as good as others and hence it makes sense to assign weights to the individual classifiers prediction, in order to improve accuracy of the ensemble classifier.

In this section we propose some technique to assign weights to the decision trees generated and instead of simple averaging we do a weighted aggregation to compute the ensemble classifier.

6.4 Choosing Weights by Solving an Optimization Problem

We can find the weights to the random decision trees generated and maximize the ensemble classification accuracy by the following quadratic minimization problem:

$$\underset{w}{min}\left(P(y=+|x)-\sum_{i=1}^{r} w_i.P(y_i=+|x)\right)$$
$$s.t.\ x\in\{0,1,*\}^n,\ w_i\in[0,1]$$

6.5 A Weighted Heuristic

We also tried some heuristics to assign weights to the individual RDTs, based upon their performance (accuracies). The idea is very simple and illustrated as follows:

- Iteratively do the following steps whenever a new test sample is to be tested: Start with uniform weights for each of the RDTs
  Test a new set of (possibly unseen) data and (re) evaluate the individual RDT accuracies
  In the next round (re) assign weights to the RDTs proportional to the individual accuracies obtained

The experimental results found are shown in 7. As we see, even if we change the weights of the individual RDTs proportional to their accuracies, the ensemble accuracy does not always improve.

7 Experimental Results for DRDT

We ran a series of experiments each time varying a set of parameters affecting the accuracy of our distributed ensemble classifier. We tried our experiment on the NSF abstract dataset and some of UCI Machine Learning datasets as well and noted the accuracy by changing different parameters. We compared the results with the centralized counter as well.

A few observations that we obtained:

- As seen from the figures, we found that if we go on increasing the training sample size, DRDT and RDT converge to the same accuracy (theoretically validated by SLLN, since sample mean approaches population mean a.s.).

- The discretization of the continuous features (even discreate features with a large number of different values) has a huge impact on accuracy. The accuracy drastically improves if binning is proper (could use sophisticated histogram binning techniques or the techniques proposed in [6] to improve the quality of binning).

- The ensemble classifier most of the times outperforms all the individual RDT classifiers in terms of accuracy.

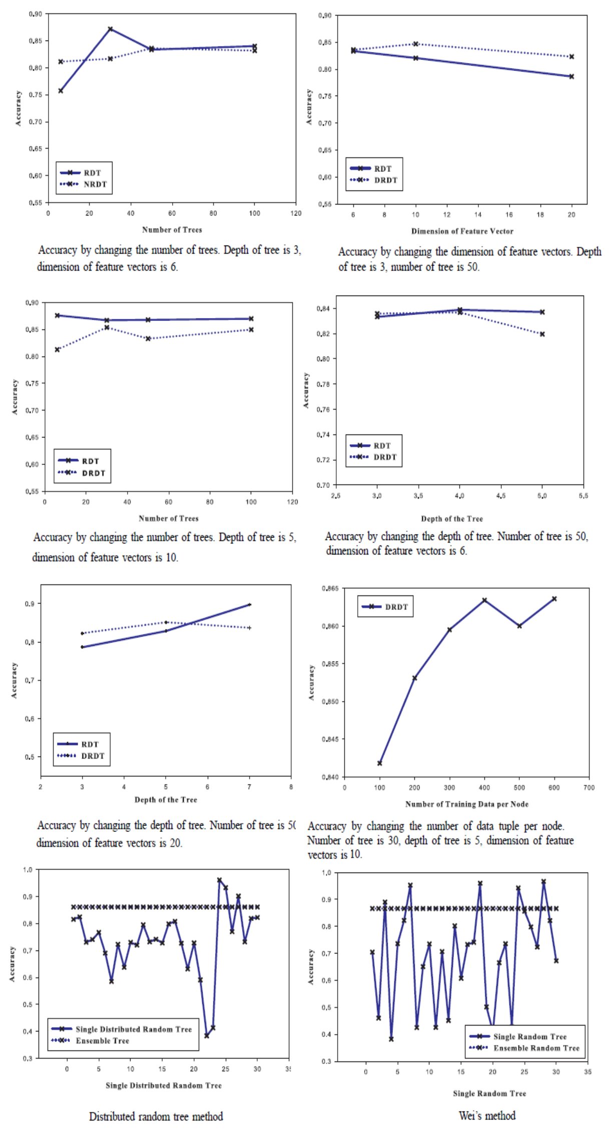


Figure 4. Experimental Results for DRDT with NSF abstract dataset and comparison with RDT

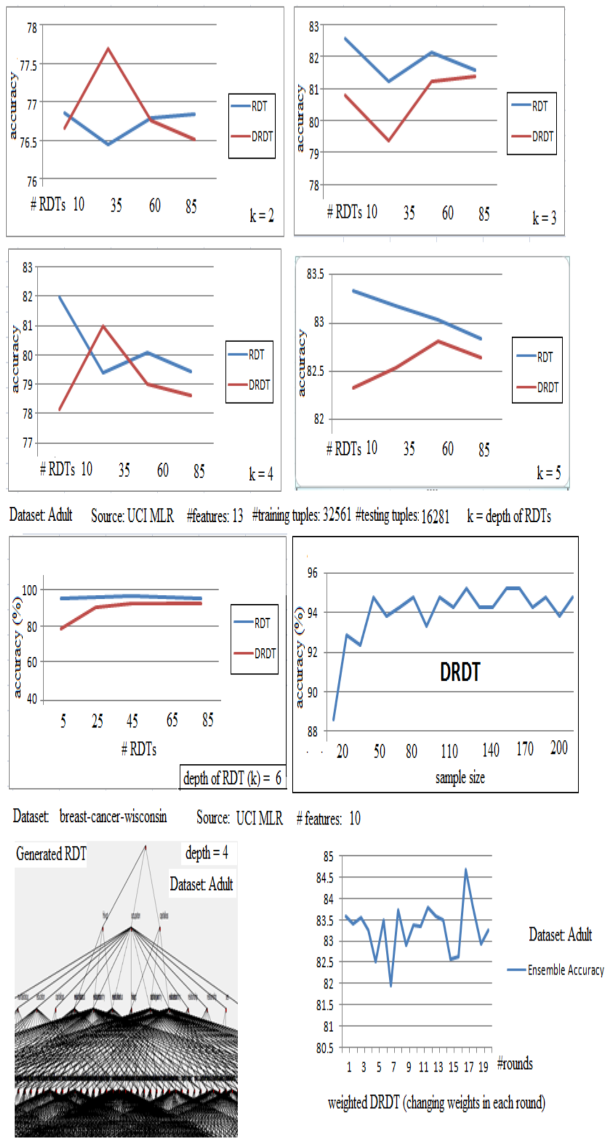


Figure 5. Experimental Results for DRDT with NSF abstract dataset and comparison with RDT

8 Conclusion

9 Acknowledgements

References

[1] Y. Amit and D. Geman. Shape quantization and recognition with randomized trees. *Neural Computation*, 9(7):1545–1588, 1997.

[2] K. Bhaduri, R. Wolff, C. Giannella, and H. Kargupta. Distributed decision-tree induction in peer-to-peer systems. *Statistical Analysis and Data Mining*, 1(2):85–103, 2008.

[3] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[4] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[5] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, 1984.

[6] S. Dey, V. P. Janeja, and A. Gangopadhyay. Temporal neighborhood discovery using markov models. In *ICDM*, pages 110–119, 2009.

[7] H. Dutta, X. Zhu, T. Mahule, H. Kargupta, K. D. Borne, C. Lauth, F. Holz, and G. Heyer. Taglearner: A p2p classifier learning system from collaboratively tagged text documents. In *ICDM Workshops*, pages 495–500, 2009.

[8] W. Fan, H. Wang, P. S. Yu, and S. Ma. Is random model better? on its accuracy and efficiency. In *ICDM*, pages 51–58, 2003.

[9] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *EuroCOLT*, pages 23–37, 1995.

[10] H. Kargupta, B.-H. Park, and H. Dutta. Orthogonal decision trees. *IEEE Trans. Knowl. Data Eng.*, 18(8):1028–1042, 2006.

[11] V. Klee and G. Minty. How good is the simplex algorithm? In *O. Shisha, editor, Inequalities*, volume III, page 159175, 1972.

[12] B. K. Natarajan. *Machine Learning: A Theoretical Approach*. Springer Netherlands.

[13] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.