Multiclass Multifeature Split Decision Tree Construction in a Distributed Environment

Jie Ouyang
Intelligent Information Engineering Lab
Department of Computer Science and Engineering
Oakland University, Rochester, MI 48309
jouyang@oakland.edu

Nilesh Patel
Intelligent Information Engineering Lab
Department of Computer Science and Engineering
Oakland University, Rochester, MI 48309
npatel@oakland.edu

Ishwar Sethi
Intelligent Information Engineering Lab
Department of Computer Science and Engineering
Oakland University, Rochester, MI 48309
isethi@oakland.edu

ABSTRACT
The decision tree-based classification is a popular approach for pattern recognition and data mining. Most decision tree induction methods assume training data being present at one central location. Given the growth in distributed databases at geographically dispersed locations, the methods for decision tree induction in distributed settings are gaining importance. This paper describes one such method that generates compact trees using multifeature splits in place of single feature split decision trees generated by most existing methods for distributed data. Our method is based on Fisher’s linear discriminant function, and is capable of dealing with multiple classes in the data. For homogeneously distributed data, the decision trees produced by our method are identical to decision trees generated using the Fisher’s linear discriminant function with centrally stored data. For heterogeneously distributed data, a certain approximation is involved with a small change in performance with respect to the tree generated with centrally stored data. Experimental results for several well known data sets are presented and compared with decision trees generated using the Fisher’s linear discriminant function with centrally stored data.

Categories and Subject Descriptors
I.5 [Computing Methodologies]: PATTERN RECOGNITION

General Terms
Algorithms, Performance

1. INTRODUCTION
The decision tree-based classification is a popular approach for pattern recognition and data mining. Methods for the induction of decision trees have been studied for past several decades. Once built, a decision tree can be used to classify previously unseen instances of patterns or to characterize patterns of different classes in the form of rules to constitute a knowledge base for decision support. The decision tree methodology has been applied to numerous applications in different domains. A comprehensive survey of decision tree induction methods from multiple disciplines and their applications is provided by Murthy[12]. Almost all methods reviewed in this survey are based on the assumption that the training data for building the decision tree is present at one central site.

With increasing globalization of businesses, advances in technology, and increasing concerns about the loss of privacy due to data mining, the interest in data mining methods that can operate in a distributed data environment has been growing. While it is always possible to move data to one central location for mining, the costs for such a move can be high. Consequently, methods for decision tree induction in a distributed data environment without large scale movement of data have started receiving attention. A distributed data environment can be homogeneous or heterogeneous. In a homogeneous environment, different sites record similar information albeit for different objects. The distributed data in such situations is also termed as horizontally partitioned data. In a heterogeneous environment, all sites record information for the same set of objects albeit different aspects of the information. Such data is referred as vertically partitioned data. Of course in reality, data may be both horizontally and vertically partitioned.

Two basic approaches to distributed decision tree induction are possible. One approach is to have an ensemble of decision trees with each data site contributing its own local decision tree to the ensemble. Examples of some popular ensemble methods are boosting, bagging, and random forests. Any of these methods can be employed in a distributed environment. One problem with ensemble approach is the difficulty of converting the ensemble decision to rules to form a knowledge base. Another drawback is that it applies only to the horizontally partitioned data scenario. A novel work in this type of approach is the orthogonal decision trees due to Kargupta et al [10] where the Fourier transformation of trees is used to combine them to arrive at a final decision tree. The other approach to distributed decision tree induction is to develop only one single decision tree by organizing the computation for a traditional decision tree induction method at one site in such a way that traffic between different data sites is minimized. The resulting decision tree then can be used by each site independently if so desired. This approach is applicable to both the horizontally and the vertically partitioned data scenarios. Furthermore, the resulting single decision tree is useful for a knowledge base across the entire enterprise. Examples of this approach are exemplified by the works described in [5][2]. In both cases, single feature decision trees are built for discrete features using entropy or Gini index as splitting criteria. Since building single feature decision trees with discrete
features needs only different counts to evaluate different features as potential split candidates, the extension to the distributed data environment is achieved easily. These methods are, however, difficult to extend even to single feature decision trees when data consists of continuous features. In such situations, choosing a split requires searching through potential cutoff values across different features. Thus, there does not appear any efficient method of computation with distributed data that can deal with continuous features.

In this paper, we suggest a method for decision tree induction in a distributed environment. Our method is based on Fisher’s linear discriminant function\cite{7} to generate multifeature splits for decision tree nodes. Since Fisher’s discriminant function is meant for two class problems, we suggest a method to extend it to multiclass problems. For homogeneously distributed data, the decision trees produced by our method are identical to decision trees generated using the Fisher’s linear discriminant function with centrally stored data. For heterogeneously distributed data, a certain approximation is involved with a small change in performance with respect to the tree generated with centrally stored data. The organization of the paper is as follows. Section 2 reviews some related work. Section 3 discusses distributed fisher’s linear discriminant and shows different relationships in the partitioned data that are used for building the decision tree in a distributed environment. Section 4 discusses extension to multiple classes by clustering classes into two super classes. Section 5 presents the final algorithm followed by experimental results in Section 6. Finally a summary of the work is provided in Section 7.

2. RELATED WORK

The most common approach for building a decision tree classifier in a distributed setting uses the tree aggregation technique which combines the locally trained classifiers into a single decision tree. Researchers have suggested numerous tree ensembling techniques. Lazarevic et al\cite{11} suggested a distributed boosting algorithm for homogeneously distributed data which combines the local classifiers into a weighted voting ensemble on each disjoint data set. Prodromidis et al\cite{14} proposed a distributed meta-learning environment, in which the local classifiers are transferred to a central site. The predictions of local classifiers on the central data is used to form the training set for meta learner. Finally an arbiter or combiner is used to determine the final classification results. In a different work, Park et al\cite{13} proposed a decision tree method for heterogeneously partitioned data. He combined the boosting and tree ensemble techniques where a measure was defined to select misclassified instances to train decision trees. Finally a Fourier transformation based method is used to ensemble decision trees from local data sites. Caragea et al\cite{4,5} proposed a different approach for distributed learning. He decomposed the learning tasks into two components: (1) hypothesis generation and (2) information extraction. The information extraction part is extended to a distributed environment while the hypothesis generation part is kept central. The information extracted by the information extractors is fed to the hypothesis generator. As long as the information extracted is the same in distributed and centralized settings, the hypothesis generator produces the same hypothesis. Hence their distributed algorithms are apodictic to their centralized counterpart, which is termed exact distributed learning. Caragea suggested the distributed information extraction based on sufficient statistics. He further demonstrated applicability of his approach on decision tree induction for both horizontally and vertically partitioned data. A slight extension of Caragea’s work is also found in [4], where he addressed the issue of counting examples from heterogeneous and autonomous data sources by the query system INDUS\cite{6}. These approaches generate single feature decision trees which are more appropriate for categorical features rather than numerical features.

Giannella et al\cite{8} proposed a method for decision tree induction for a heterogeneous environment. His method uses the feature splitting criteria which tries to reduce the coordination cost for determining instances of current tree node. Instead of keeping a copy of node assignment at the local site, the algorithm tries to determine the tree node instances when it is to be split. The procedure is based on the dot product of binary vectors which are node assignment indicators of different sites. To reduce the communication cost of the binary vectors, a random projection method is used to reduce their dimensionality. The information loss caused by the random projection is also analyzed. Baik et al\cite{1} suggested a different method for reducing the intra node communication. He proposed encoding of node assignment bit vectors. Both these approaches also generates a single feature split decision trees.

Bar-Or et al\cite{2} suggested a distributed decision tree algorithm for data sites organized in a hierarchical structure. Such structure moves the information through a path from the leaf data site to the root data site. In principle, this algorithm is more suitable for horizontally partitioned data. The method defines the lower and upper bound of the gain function which is further used to efficiently collect gain values using the hierarchical structure. Their approach is suitable for any high dimension data, provided that the correlations in it are sparse.

3. DISTRIBUTED DECISION TREE INDUCTION USING FISHER’S LINEAR DISCRIMINANT

Decision Tree is among the most important non-parametric technique for building classifiers. Decision tree algorithms such as C4.5\cite{15} or CART\cite{3} have attracted many researchers because its ability to provide intuitive interpretations. The standard decision tree build process follows a top-down divide and conquer technique, in which the induction starts with a root node containing all data. The tree nodes are then recursively partitioned until a stopping condition is satisfied. Early decision tree induction methods followed a single feature split technique. In this technique, one feature with the greatest gain function value is picked for splitting the tree. Single feature split decision trees could grow very large leading to suboptimal accuracy. Multifeature decision tree induction techniques are introduced to overcome the size and performance issues. The later technique splits the data using combination of features. If the combination criterion is linear, the tree is called linear discriminant tree. One method to determine the feature combination coefficients is using perceptron learning\cite{16}. Another method is to use Fisher’s Linear Discriminant(FLD)\cite{7} as suggested in [17].

3.1 Fisher’s Linear Discriminant(FLD)

Fisher’s Linear Discriminant aims at finding the projection vector $W$(combination coefficient vector) such that the projection of the original data has the best discriminant ability. Given a set of augmented training vectors, $(X_1, X_2, ..., X_n)$, from two classes $C_1$ and $C_2$, FLD tries to determine $W$ such that it maximizes -

$$W = \max_{k} \frac{W^T(m_1 - m_2)(m_1 - m_2)^TW}{W^T(S_1 + S_2)W}, \tag{1}$$

where $m_1$ and $S_1$ are the mean and the scatter matrix of class $i$ respectively. The solution of Eq.1 can be represented as -

$$W = (S_1 + S_2)^{-1}(m_1 - m_2). \tag{2}$$
Assuming the projection of each class to be one dimensional Gaussian, the splitting threshold can be easily computed using the joint point, \( M \), of two Gaussian Probability Distribution Functions (PDFs). One well-known issue with FLD is possible singularity of \((S_1+S_2)\), making it difficult to obtain its inverse. This difficulty is often triggered by mismatch between number of data point instances with respect to its dimensionality. Principle Component Analysis (PCA) is used as one common practice to overcome this problem. Specifically, PCA is used to reduce the dimensionality of input data before FLD is applied. It should be noted that FLD is more suitable for numerical data, although the non-numeric data can also be trained to generate a decision tree using suitable preprocessing (data transformation) technique.

### 3.2 Statistics Required by FLD

The decision tree algorithm proposed in this paper is based on performing FLD in a distributed environment. As established in the subsection 3.1, this requires computation of class means and scatter and thereby computing the projection vector that separates the two classes of data. In this section we establish the theoretical foundation to compute the class means and scatter in a homogeneous and heterogeneous distributed environment. Without loss generality, we consider the case of two data sites. Let's further consider one of these two sites to be a coordinating or a master site which also contains one partition of the data. Hence the dataset \( D_{\text{total}} \) is divided among two sites, consider these to be \( X \) and \( Y \). Let's define few terms which are central to our algorithm.

- **Global Mean**\((m)\): Represents the mean of the total dataset.
- **Global Scatter**\((S)\): Represents the scatter (covariance) of the total dataset.
- **Global Class Mean**\((m_i)\): Represents the mean of the data belonging to class \( i \).
- **Global Class Scatter**\((S_{ij})\): Represents the scatter (covariance) of the data belonging to class \( i \).
- **Local Class Mean**\((m_{ij})\): Represents the mean of the data belonging to class \( i \) available at site \( j \).
- **Local Class Scatter**\((S_{ij})\): Represents the scatter (covariance) of the data belonging to class \( i \) available at site \( j \).

Table 1 summarize the denotations of these different statistics. In the following subsection we introduce the theoretical foundation to compute these statistics in a homogeneously and heterogeneously partitioned data.

#### 3.2.1 Computing FLD Statistics in Homogeneous Setting

Consider data \( D_{\text{total}} \) being homogeneously partitioned between two sites. Let \( X \) and \( Y \) be the data partitions of \( d \) dimensions where \( n = n_X + n_Y \), \( D_{\text{total}} = \left[ \begin{array}{c} X \\ Y \end{array} \right] \). Given this distribution we can easily compute class mean \((m_i)\) and global mean \((m)\) as -

\[
\begin{align*}
m_{ij} &= \frac{\sum_{i \in j} X_{ij}}{n_{ij}} \quad (3) \\
m_i &= \frac{\sum_i m_i n_{ij}}{\sum_i n_i} \quad (4) \\
m &= \frac{\sum_i m_i n_i}{\sum_i n_i} \quad (5)
\end{align*}
\]

Eq. 4 and Eq. 5 are computed at the master site. The communication cost for transferring \( m_{ij} \) and \( n_i \) in the distributed environment of \( K \) sites is \( O(KCd^2) \) where \( C \) is the number of classes. Assuming \( D \) being zero mean, which can be achieved by shifting all the data by its global mean, the scatter of \( D \) can be calculated as -

\[
\begin{align*}
S_D &= \left[ X^T \right] [X] \\
&= [X^TX + Y^TY] \\
&= S_X + S_Y \quad (6)
\end{align*}
\]

Generalizing Eq 6 to \( K \) sites leads us to the solution -

\[
\begin{align*}
S_i &= \sum_j S_{ij} \quad (7) \\
S &= \sum_{i,j} S_{ij} \quad (8)
\end{align*}
\]

The communication cost for transferring \( S_{ij} \) in an environment of \( K \) sites is \( O(KCd^2) \). Assuming \( d \ll n \), \( K \ll n \), and \( C \ll n \), the total communication cost \( KCd^2 << nd \).

#### 3.2.2 Computing FLD Statistics in a Heterogeneous Setting

It is conceivable that distributed data is not always partitioned homogeneously (data split by samples). Many situations necessitates data split by its dimensionality. Such distribution is also referred as vertical or heterogeneous data partitioning. Let \( X \) and \( Y \) be the data partitions with the condition \( d = d_X + d_Y \). The data matrix \( D \) is denoted by \( D = [X \ Y] \). Again, without loss of generality we can compute the means as -

\[
m_i = \left[ m_{i1}, m_{i2}, \ldots, m_{ik} \right] \\
m = \frac{\sum_i m_i n_i}{\sum_i n_i} \quad (9)
\]

\[
\begin{align*}
S_D &= \left[ X \ Y \right]^T [X \ Y] \\
&= [X^TX \ X^TY] \quad (11) \\
&= [S_X \ X^TY] \\
&= [X^TY \ S_Y]
\end{align*}
\]

To calculate the nonlocal (off diagonal) items in Eq.11, one of the data partitions needs to be transferred to the master site, which is undesirable. Hence \( S_D \) has to be approximated. We can accomplish this in two ways which are both based on approximating original
data using truncated singular value decomposition (SVD)[9]. Assume data \( X \) needs to be transferred to where \( Y \) is located. We can factorize \( X \) to its singular values as:

\[
X = U\Sigma V^T = U_1\Sigma_1 V_1^T + \ldots + U_d\Sigma_d V_d^T,
\]

(12)

where the matrix \( V \) (right eigen vector matrix) contains a set of orthonormal basis vector directions for \( X \), the matrix \( U \) (left eigen vector matrix) contains a set of orthonormal output basis vector directions for \( X \), and the matrix \( \Sigma \) contains the singular values. The singular values are also considered scalar gain controls. The first item in the expansion, the basis vector directions for \( X \), contains the most information of \( X \); we can use \( U_i\Sigma_i V_i^T \) only to approximate \( X \), denoted by \( \hat{X} = U_i\Sigma_i V_i^T \).

Our first method approximating \( S_d \) aims computing as many exact entries (entries with the same values in \( S_d \) and its approximation) as possible. Hence the sub-matrices \( X^T Y \) and \( Y^T X \) are respectively approximated by \( \hat{X}^T Y \) and \( Y^T \hat{X} \) while \( S_k \) is an exact entry computed by \( V^*\Sigma^* + \Sigma + V \). This approximation requires transmission of only \( U_i, \Sigma, \) and \( V \), keeping the communication cost down to \( O(n + d^2) \) for two sites environment. Thus for an environment of \( K \) sites, the cost is scaled to \( O(Kn + Kd^2) \), or \( O(Kn) \) when \( d << n \). The approximation of \( X^T Y \) and \( Y^T X \) loses some information. The lose of information may lead to drop in overall classifier performance, which can be addressed by transferring more than 1, say \( p \), vectors from \( U \). The cost of communication then would be \( O(Kpn) \).

The second method is based on approximating the entire dataset. In this method, every local site performs SVD on locally stored data and sends its own \( U_i, \Sigma, \) and \( V \) to the master site. The master site then forms an approximation of the original dataset, and computes the scatter in a centralized manner. The difference of this method is that only local items are exact entries in \( S_d \), e.g., \( S_k \) will be replaced with its approximation. Ironically, both techniques have the same communication cost model.

The information transfer in method 1 can be done in a serial or parallel manner. In the serial approach, every site needs storage space for the information from the previous site only. However in method 2, the master site needs storage space to hold information from all local sites. On the contrary, method 2 requires one time information transfer during the whole tree induction process while method 1 requires information transfer for every tree node split. The implementation implication is the trade-off between space and transmission.

4. MULTICLASS CLASSIFICATION USING HIERARCHICAL SUPER CLASSING

The Fisher’s Linear Discriminant algorithm was originally proposed for a two class problem. Over these years, researchers have proposed numerous FLD extensions to solve the multiclass problems. For example, the authors of [17] have suggested the use of exchange method for reducing multiclass problems to two-class problems. However the exchange method does not fit well for a distributed environment. To cope with multiclass problems, we propose use of hierarchical structuring of the data and apply traditional class method on this pre-organized data. Our solution keeps the problem formation simple in a distributed environment. The hierarchical structuring orders the data to transform the multiclass problem to a two class problem. We generate this hierarchy by estimating two super classes. This is accomplished by hierarchically clustering the class means at the master site. The super class assignment is then sent to each local sites to get the statistics of the super classes. The super class means and scatters are obtained using the process outlined in previous section using Eq.3 and Eq.7. One is-sue of the super class generation is what class mean to use. The class mean can be computed in the original space, in the full scale principle component space, or in the reduced principle component space. We performed the experiment using each to evaluate the difference in classifier performance. Our empirical study did not observe significant performance difference between each method. The experiment results reported in this paper uses the class means in the full scale principle component space.

In short, Principal Component Analysis, Hierarchical Super Classing, and Fisher’s Linear Discriminant constitute three key components of our algorithm. These three steps are carried out iteratively in the following manner:

1. Compute PCA to gets a full scale eigen vector matrix.
2. Calculate Global class means in the eigen vector space.
3. Generate Super classes using the class means from step [2].
4. Compute Super class means and scatters(\( S_{sc1}, S_{sc2} \)).
5. Asses the singularity of \( S_{sc1} + S_{sc2} \).
6. Remove the least significant eigen vector and repeat step 4 and 5, if assessed close to singular; otherwise performance FLD to compute projection vector \( W \).

It shall be noted that the repetition of step 4 does not require extra communication. It is performed by removing certain vectors from the means and scatters.

5. INDUCTION OF DISTRIBUTED MULTI-FEATURE DECISION TREE (DMDT)

The generalized Distributed Multifeature Decision Tree solution is depicted in Figure 1. Consider the data, containing \( L \) classes, being distributed among \( K \) sites denoted as \( LS_1, LS_2, ..., LS_K \). Let us denote the central or master site to be \( MS \). Although our algorithm specifically suggest separate master site, it is equally applicable in a peer to peer setting where there is no functional coordinator. In such situation, any one of the local site can fulfill this role master site. Clearly, each participating site is responsible for computing its local class means (\( m_{ij} \)) and local class scatter (\( S_{ij} \)). Furthermore, the local sites are also responsible for maintaining the decision tree. The master site is responsible for performing PCA, dividing the
data among super classes, and computing the projection plane using the FLD. Following pseudo code captures the steps involved in induction of a multifeature decision tree in a distributed environment. For completeness, we outline three pseudo code algorithms, one for homogenous and two for heterogeneous data distributions. The use of \( \rightarrow \) signifies the data transmission direction.

**Algorithm 1 DMDT in Homogeneous Environment**

1: \( \forall j \in K \) and \( \forall i \in L \), Compute\((m_{i,j}, n_{i,j}) \rightarrow MS\)
2: \( LS_j \leftarrow \text{Compute}(n)\)
3: \( \text{Compute}(S_{n,j}) \rightarrow MS\)
4: \( \text{Compute}(S) \rightarrow MS\)
5: \( \text{PCA} @ MS\)
6: \( LS_j \leftarrow \text{GenerateSuperclass}(SC_1, SC_2)\)
7: \( LS_j \leftarrow \text{Compute}(m_{i,j}, n_{i,j})\)
8: \( \text{Compute}(S_{m,j}, S_{n,j}) \rightarrow MS\)
9: \( \text{Compute}(S_{m,j}, S_{n,j}) @ MS\)
10: \( \text{while Singular}(S_{m,j} + S_{n,j}) \text{ do} \)
11: \( \text{Remove the least significant principle component.} \)
12: \( \text{end while} \)
13: \( \text{FLD} @ MS\)
14: \( LS_j \leftarrow (W, W_0)\)

Please note that the transmission of \( m \) to \( LS_j \) can be saved in implementation as the global data can be made zero mean by making each local dataset zero mean. The two algorithms for the heterogeneous environment correspond to the two approximation solutions described in section 3.2. The first pseudo code describes the computation steps carried out at each non-leaf tree node, while the second piece of pseudo code describes the tree induction process.

**Algorithm 2 DMDT in Heterogeneous Environment 1**

1: \( \forall j \in K \) and \( \forall i \in L \), Compute\((m_{i,j}, n_{i,j}) \rightarrow MS\)
2: \( \forall j \in K \), SVD\((U_{i,j}, \Sigma_j, V_j) \rightarrow MS\)
3: \( \text{Compute}(m_{i,j}, S) @ MS\)
4: \( \text{PCA} @ MS\)
5: \( \text{GenerateSuperclass}(SC_1, SC_2) @ MS\)
6: \( \text{Compute}(m_{i,j}, m_{n,j}, S_{n,j}, S_{m,j}) @ MS\)
7: \( \text{while Singular}(S_{m,j} + S_{n,j}) \text{ do} \)
8: \( \text{Remove the least significant principle component.} \)
9: \( \text{end while} \)
10: \( \text{FLD} @ MS\)
11: \( LS_j \leftarrow (W, W_0)\)

**Algorithm 3 DMDT in Heterogeneous Environment 2**

1: \( \forall j \in K \), SVD\((U_{i,j}, \Sigma_j, V_j) \rightarrow MS\)
2: \( \forall j \in K \), Compute\((U_{i,j}, \Sigma_j * V_j) @ MS\)
3: \( \text{Compute}(F_{LDT}) @ MS\)

Once the tree induction process is complete, we perform the classification to assess the performance of our classifier. Classification of the new instance for homogenous distributed environment is no different than the centralized setting. Since each site carries a copy of the decision tree, new data is classified immediately without any communication overhead. It is not the same for the heterogeneous distribution of data. It requires the data transformation before classification. Consider \( x \) be the feature partition of a new instance at site \( j \). Let \( \Sigma_j \) and \( V_j \) be the components obtained by SVD of the training samples at \( j \). The row in \( U \) for \( x \) can be obtained from \( x \times V_j \times (\Sigma_j)^{-1} \). The transformation of \( x \) is computed as -

\[
\tilde{x} = (x \times V_j \times (\Sigma_j)^{-1})_i+p \times (I_j \times V_j)
\]

\( \tilde{x} \) is then sent to the master site to fill in its position in the approximated instances. The approximated instance is then classified at the master site.

6. ACCURACY EXPERIMENTAL RESULTS

To assess the performance of our decision tree constructed in a distributed environment, we compared its accuracy against a centrally build decision tree. The experiments are conducted over 13 UCI datasets. The data sizes range from couple hundreds to several thousands and the dimensions range from 5 to 65. To test the ability of DMDT to process multiclass problems, the class numbers of these datasets range from 2 to 10. In addition, the complexity of the datasets varies as the accuracy of some popular decision tree algorithms on them ranges from around 50% to around 90%. In order to check the accuracy of our classifier, we performed two runs of five folds cross validation on each dataset.

The first experiment measures the accuracy of distributed classifier in both homogeneous and heterogeneous environments with respect to the centralized version. To simulate a homogeneous distribution of 4 sites, we first permuted the data instances and then partitioned it in 4 random segments. Each segment is then distributed to four virtual processing sites. Due to relatively small dimensionality, we simulate the heterogeneous distribution using only 3 sites. In this setting, the features are evenly partitioned and distributed among available 3 sites. In addition, each site receives a copy of the class labels. It shall be noted that heterogeneous distribution is only tested on the datasets having greater than six features. The constraint was necessary to ensure that each site receives at least two features. The experimental results are tabulated in Table 2. Column with title \textit{Hetero 1} shows the results from algorithm2 while the column titled \textit{Hetero 2} corresponds to algorithm3. Both are obtained with the number of left eigen vectors, denoted by \( p \) = 1.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Classes</th>
<th>Instances</th>
<th>Features</th>
<th>准确率</th>
<th>准确率</th>
<th>准确率</th>
<th>准确率</th>
<th>准确率</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breast</td>
<td>2</td>
<td>699</td>
<td>30</td>
<td>90.65±4.86</td>
<td>90.54±4.78</td>
<td>90.54±4.76</td>
<td>89.76±4.62</td>
<td></td>
</tr>
<tr>
<td>Hepatitis</td>
<td>2</td>
<td>507</td>
<td>1</td>
<td>87.8±2.25</td>
<td>89.9±3.79</td>
<td>82.0±5.15</td>
<td>84.5±4.30</td>
<td></td>
</tr>
<tr>
<td>Vote</td>
<td>2</td>
<td>845</td>
<td>17</td>
<td>93.6±6.90</td>
<td>95.6±8.01</td>
<td>91.0±6.39</td>
<td>94.7±6.81</td>
<td></td>
</tr>
<tr>
<td>Ionosphere</td>
<td>2</td>
<td>131</td>
<td>35</td>
<td>99.0±6.36</td>
<td>88.1±8.03</td>
<td>88.1±8.90</td>
<td>84.0±2.35</td>
<td></td>
</tr>
<tr>
<td>Wine</td>
<td>3</td>
<td>178</td>
<td>14</td>
<td>99.3±1.44</td>
<td>98.3±2.45</td>
<td>98.3±4.49</td>
<td>98.3±1.27</td>
<td></td>
</tr>
<tr>
<td>Iris</td>
<td>3</td>
<td>150</td>
<td>5</td>
<td>97.3±4.62</td>
<td>97.3±4.62</td>
<td>97.3±4.62</td>
<td>97.3±4.62</td>
<td></td>
</tr>
<tr>
<td>Car</td>
<td>4</td>
<td>172</td>
<td>7</td>
<td>85.3±6.02</td>
<td>80.3±4.10</td>
<td>76.3±3.10</td>
<td>78.2±4.73</td>
<td></td>
</tr>
<tr>
<td>Dermatology</td>
<td>6</td>
<td>305</td>
<td>35</td>
<td>97.4±8.10</td>
<td>99.4±8.35</td>
<td>94.1±4.65</td>
<td>94.0±2.93</td>
<td></td>
</tr>
<tr>
<td>Segment</td>
<td>7</td>
<td>2310</td>
<td>20</td>
<td>91.16±6.2</td>
<td>90.6±6.32</td>
<td>84.0±8.96</td>
<td>84.0±5.33</td>
<td></td>
</tr>
<tr>
<td>Zoo</td>
<td>7</td>
<td>108</td>
<td>18</td>
<td>92.10±5.86</td>
<td>91.1±3.92</td>
<td>81.0±5.51</td>
<td>80.0±5.41</td>
<td></td>
</tr>
<tr>
<td>Glass</td>
<td>7</td>
<td>214</td>
<td>10</td>
<td>87.8±8.47</td>
<td>78.4±9.21</td>
<td>64.0±5.53</td>
<td>61.0±9.84</td>
<td></td>
</tr>
<tr>
<td>Opinions</td>
<td>10</td>
<td>5823</td>
<td>65</td>
<td>94.45±4.30</td>
<td>95.19±4.97</td>
<td>95.4±8.57</td>
<td>95.4±10.19</td>
<td></td>
</tr>
<tr>
<td>Pendigits</td>
<td>10</td>
<td>7494</td>
<td>17</td>
<td>99.26±4.40</td>
<td>97.8±7.62</td>
<td>96.8±11.37</td>
<td>95.3±10.18</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Performance of Distributed Multifeature Decision Tree
oberved that the performance drops do not have any strong correlation with the average number of features per site. Algorithm3 has systematically outperformed algorithm2. The reason for that is observed in the consistency of approximated scatter matrix, which was found to be better in algorithm3 than algorithm2.

The second experiment measures the impact of $p$ on the performance. In this case, the results are obtained only using heterogeneous settings as Table2 shows algorithm3 outperforms algorithm2. The column titled Features represents the average number of features at each site. The value of $p$ is increased by one for each trial. The experiment was stopped as soon as the accuracy of certain $p$ value is close to the accuracy of the centralized version. Table3 captures the results of our second experiment run measuring the impact of $p$. It is apparent that in most cases (9 datasets out of 12) our method reports comparable results with less than half of the communication bandwidth requirement.

Third, we compare the performance of DMDT with two popular decision tree algorithms, ID3 and CART. We only refer to the experimental results of their centralized versions since the theoretial proof described in [4][5] shows that the performance of exact distributed learning will be the same as its centralized counterpart. The comparison is shown in Table4. We quote the accuracy of ID3 and CART from [17]. The performance of implementing algorithm for heterogeneous environment is proportional to the total number of data instances, we empirically proved that our approach could reach the performance of a centralized version while saving more than half of the communication cost. Also we empirically demonstrated distributed multifeature decision trees can reach similar or better performance than the distributed single feature decision trees with low communication cost in both homogeneous and heterogeneous environments.

### Table 3: Impact of $p$ on Heterogeneous Data Distribution

<table>
<thead>
<tr>
<th>Dataset</th>
<th>$p=1$</th>
<th>$p=2$</th>
<th>$p=3$</th>
<th>$p=6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breast</td>
<td>96.6±1.12</td>
<td>94.9±1.44</td>
<td>95.9±1.78</td>
<td>97.8±0.82</td>
</tr>
<tr>
<td>Bupa</td>
<td>62.3±8.58</td>
<td>67.1±7.43</td>
<td>66.9±5.64</td>
<td>64.5±5.40</td>
</tr>
<tr>
<td>Vote</td>
<td>94.9±1.11</td>
<td>90.3±3.22</td>
<td>95.1±1.81</td>
<td>94.7±1.81</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>87.6±3.32</td>
<td>90.6±4.01</td>
<td>89.2±2.69</td>
<td>92.8±2.55</td>
</tr>
<tr>
<td>Wine</td>
<td>93.7±3.97</td>
<td>97.3±4.43</td>
<td>98.3±2.45</td>
<td>94.1±0.27</td>
</tr>
<tr>
<td>Iris</td>
<td>93.9±2.84</td>
<td>93.9±3.44</td>
<td>97.3±3.62</td>
<td>92.1±2.62</td>
</tr>
<tr>
<td>Dermatology</td>
<td>92.2±2.44</td>
<td>90.4±6.49</td>
<td>97.4±2.35</td>
<td>94.0±0.43</td>
</tr>
<tr>
<td>Segment</td>
<td>91.5±1.82</td>
<td>90.6±1.67</td>
<td>92.6±0.42</td>
<td>94.2±0.30</td>
</tr>
<tr>
<td>Zoo</td>
<td>94.8±1.51</td>
<td>90.6±1.57</td>
<td>92.2±0.54</td>
<td>94.8±0.54</td>
</tr>
<tr>
<td>Glass</td>
<td>93.3±2.53</td>
<td>93.2±2.41</td>
<td>94.6±0.21</td>
<td>93.0±0.19</td>
</tr>
<tr>
<td>Opinions</td>
<td>85.4±3.85</td>
<td>91.4±3.13</td>
<td>92.6±0.39</td>
<td>94.0±0.99</td>
</tr>
</tbody>
</table>

### Table 4: Performance Comparison

The algorithm systematically performs better than ID3 and CART in homogeneous settings. This is because DMDT’s more suitable for describing oblique decision surfaces. Also fisher’s linear discriminant analysis increases the generality of the decision tree. The performance of algorithm3 is comparable to ID3 and CART for most of the datasets even with the smallest possible $p$ setting. For those three datasets(Dermatology, Segment, and Pendigits) with performance loss for more than 5 percent, similar or better performance of DMDT can be reached when $p$ is set to 2 (see Table3), which is a small number compared to the dimensionality.

### 7. CONCLUSION

In this paper, we propose a multifeature decision tree algorithm for distributed environment. Our approach extends the popular Fisher’s Linear Discriminant in a straightforward fashion to deal with homogeneous data distribution. The correctness of the theoretical proof showing the equivalency between centralized and distributed algorithm for homogeneously distributed data is further verified using the experimental results. For heterogeneously distributed data, we introduced the data approximation technique. Two data approximation approaches are suggested and compared for their relative performances. Our experimental results show that the consistency in estimating scatter matrix considerably improves the classification performance. Although, the communication cost of implementing algorithm for homogeneous environment is proportional to the total number of data instances, we empirically proved that our approach could reach the performance of a centralized version while saving more than half of the communication cost. Also we empirically demonstrated distributed multifeature decision trees can reach similar or better performance than the distributed single feature decision trees with low communication cost in both homogeneous and heterogeneous environments.

### 8. REFERENCES


