基于最小生成树的大规模数据分类模型及其MapReduce实现

黄 鑫[1, 2] 罗 军[1]

[1]（中国科学院深圳先进技术研究院 深圳 518055）
[2]（中国科学院大学 北京 100049）

摘 要 数据的快速增长，为我们提供了更多的信息，然而，也对传统信息获取技术提出了挑战。这篇论文提出了MCMM算法，它是基于MapReduce的大规模数据分类模型的最小生成树（MST）的算法。它可以看做是介于传统的KNN方法和基于聚类分类方法之间的模型，旨在克服这两种方法的不足并能处理大规模的数据。在这一模型中，训练集作为有权重的无向完全图来处理。顶点是对象，两点之间边的权重是对象间的距离。这一距离，不同于欧几里得距离，它是一个特定的距离度量。这样，可以找到图中最小生成树集，其中，图中每棵树代表一个类。为了降低时间复杂度，提取了每棵树中最具代表性的点来代表该树。这些压缩了的点集，可以通过计算无标签对象和它们之间的距离，来进行分类。MCMM模型基于MapReduce实现并且部署在Hadoop平台。该模型可扩展处理大规模的数据，是因为Hadoop支持数据密集分布应用，并且这些应用可以和数以千计的节点和数据一起运作。另外，MapReduce 和Hadoop能在由商品机组成的集群上很好的运行。MCMM模型使用云平台并且通过使用MapReduce 和Hadoop进行云计算是有益处的。实验采用的数据集包括从UCI数据库得到的真实数据和一些模拟数据，实验使用了4000个集群。实验表明，MCMM模型在精确度和扩展性上优于KNN和其他一些经常使用的基础分类方法。

关键词 最小生成树；分类；MapReduce；云计算；图挖掘

A Classification Model for Massive Data Based on Minimum Spanning Tree with MapReduce Implementation

HUANG Xin[1,2] LUO Jun[1]

[1](*Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen* 518055, *China*)
[2](*University of Chinese Academy of Sciences, Beijing* 100049, *China*)

Abstract The Rapid growth of data has provided us with more information, yet challenges the traditional techniques to extract the useful knowledge. In this paper, MCMM, a Minimum spanning tree (MST) based Classification model for Massive data with MapReduce implementation is proposed. It can be viewed as an intermediate model between the traditional K nearest neighbor method and cluster based classification method, aiming to overcome their disadvantages and cope with large amount of data. In this model, we treat the training set as weighted undirected complete graph. The vertices are objects and the weight of an edge between two objects is their distance, which could be a certain distance metric other than Euclidean distance. Then we find a minimum spanning tree forest of the graph, in which each tree represents a class. In order to reduce the computing time, we extract the most representative points of each tree to represent that tree. The shrunk point sets can be used for classification by computing the distances from unlabeled objects to them.MCMM model is implemented on Hadoop platform, using its MapReduce programming framework. Since Hadoop supports data intensive distributed applications and enables applications to work with thousands of nodes and petabytes of data, MCMM model is scalable to deal with massive data. In addition, MapReduce and Hadoop work well on cluster composed of commodity machines. Therefore there is no special need for particular hardware or architecture. This is actually the feature of cloud

作者简介：黄鑫，硕士研究生，研究方向为计算机应用技术；罗军，副研究员，研究方向为算法分析与设计、数据挖掘、GIS，E-mail:jun.luo@siat.ac.cn。

computing. MCMM model is used on cloud platform and could benefit from cloud computing by using Hadoop and MapReduce. Experiments had been carried out on several data sets including real world data from UCI repository and synthetic data, using Downing 4000 cluster, installed with Hadoop. The results show that MCMM model outperforms KNN and some other classification methods on a general basis with respect to accuracy and scalability.

Keywords minimum spanning tree; classification; MapReduce; cloud computing; graph-based mining

1 Introduction

Classification is one of the most active research fields in data mining and machine learning, which is widely used in many application areas, including e-commerce, WWW, bioinformatics, scientific simulation, customer relationship management, business intelligence etc. With the development of hardware and software, it is becoming normal that the size of databases goes to Gigabytes or even larger. This raises new challenge to data mining techniques, including classification and so on.

There are many techniques for classification. If full knowledge of underlying distribution is available, Bayes analysis yields an optimal decision procedure and minimum probability of error[4]. Sadly this kind of knowledge is always hard or impossible to get, in which cases many algorithms make use of distance or similarity among samples to classify them. The K nearest neighbor technique[4] falls into this category and is widely used in a lot of areas for its simplicity to handle and generally has high accuracy. However, KNN classifier faces the problem that it may decrease the precision because of the uneven density of training data. Also KNN has to compute the distances between an unlabeled object and every object in training set. When the size of data set goes to several gigabytes, which is common in today's information explosive world, the time for classifying becomes unacceptable. The cluster based classification extends the basic idea of KNN[30]. It first performs clustering on the training set, and each cluster belongs to a particular class, and then uses certain kind of center point to represent each cluster. Classification stage is similar to KNN. The only difference is that the training set is composed of those clusters' centers. Although this method reduces the total points of training set, it may lose too much information and is only suitable for convex group, which can be sufficiently substituted by a center point. What's more, traditional clustering methods on huge amount of data consume too much time or even can't be applied to massive data due to memory limitation.

In this paper, we present a classification model which tries to find an intermediate model between the above two extremes, aiming at benefiting from their advantages, and removing some of the drawbacks. One direct and simple way is to use a certain kind of subtree to represent each cluster which is obtained by clustering on training set, and then classify using idea similar to KNN. To achieve this goal, we firstly use minimum spanning tree (MST) for clustering, which is a simple and effective way compared with other traditional clustering method. Each cluster we get is actually a subtree, whose majority nodes are of the same class. Next step, we extract the most representative points of each subtree, and then get shrunk subtrees (actually they are subsets of nodes. We use subtree for convenience in the remaining text). By calculating the distances between an unlabeled object and each shrunk subtree, we can select the nearest subtree and classify the object into this subtree. The reason why shrunk subtrees are used is to reduce the quantity of training set which overwhelms KNN, yet without losing too much useful information. Using subtree is a vital feature when the cluster is not convex or of irregular shape.

Another notable feature of our classification model is that it can cope with a huge amount of data from modeling to classification in an effective way, especially in the period of clustering, because we use MapReduce distributed programming framework, which has the ability to processe huge amounts of data in parallel, using hundreds of machines. We have done experiments on Downing 4000 cluster installed with Hadoop, an open source implementation of MapReduce. It shows that

our model outperforms KNN and some other traditional classification methods both in accuracy and efficiency. And the nature of MapReduce's distributed computing ability endows our model with good scalability.

The rest part of this paper is organized as follows. Section 2 presents the background knowledge and related work. Section 3 describes our classification model in details. How to implement the model with MapReduce is presented in Section 4. Experiments and results are shown in Section 5, and we conclude the paper in Section 6. Pay attention that we may use minimum spanning tree and MST interchangeably in this paper, and they mean the same thing.

2 Background and Related Work

In this section, we give some background knowledge and a brief description of existing classification algorithms related to our model.

2.1 Hadoop and MapReduce

Hadoop is the Apache Software Foundation top-level project. It provides and supports the development of open source software that supplies a framework for the development of highly scalable distributed computing applications. The two fundamental parts of Hadoop Core are MapReduce framework, the cloud computing environment, and Hadoop Distributed File System (HDFS). It also provides other projects, such as HBase, Hadoop's distributed, column-oriented database efficiently storing and handling semi-structured data as Google's Big Table storage system, and PIG, a high level language for data analysis[32]. Hadoop is drawing more and more attention due to its simplicity and power for the development of distributed applications on cloud.

Hadoop MapReduce is a software framework for easily writing applications which process vast amounts of data (multi-terabyte data sets) in parallel on large clusters of commodity hardware (e.g. cloud platform) in a reliable, fault-tolerant manner[23]. It is based on two distinct steps. First step is map: the framework sequentially passes over the input file and output (key, value) pairs, in which individual input records can be processed in parallel. Second step is reduce: it firstly groups all values by key, then processes the values with the same key and outputs the final result. The framework shields the programmer from the details about the data distribution, replication, fault-tolerance, load balancing, etc. So the programmer only needs to provide two functions, a map and a reduce. Yet it's powerful enough to process more complicated problems than just word counting. It can perform sorting, joining and many other operations on massive data in an efficient way[26]. Kang et al. present PEGASUS in [13], which is a tool for large scale graph mining applications. The key functions of PEGASUS are all implemented by MapReduce, including finding connect components, calculating pageranks, estimating diameter. Karloff et al. [15] prove that it is possible to find the minimum spanning tree of a huge graph by using MapReduce when traditional algorithms for computing MST consume intolerable time or exceeds the limitation of single machine's memory.

2.2 Clustering Based Classification

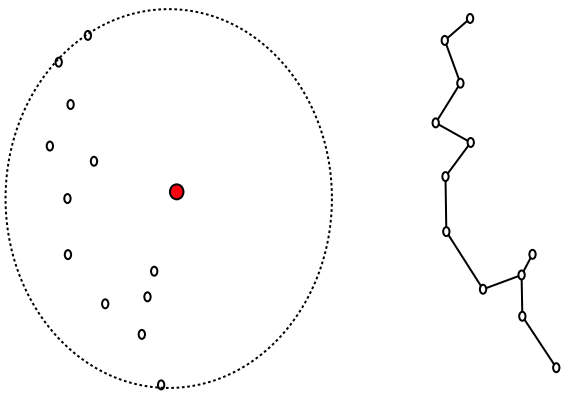
Clustering methods have been applied to supervised classification problems[30, 17, 10]. In [20], Mui et al. illustrate building a cluster tree classification model using the k-means clustering method. The problem with their model is that only small numeric data could be classified and every time only two sub-clusters are formed. In [10], Huang et al. proposed a new interactive approach to build a decision cluster classification model, in which the k-prototypes clustering algorithm is used to partition the training data. But the above two methods are not adequate for high dimensional data with noise. In [18], a variable weighting k-means algorithm to build cluster-based classification models automatically is proposed, which can reduce the impact of noisy attributes by assigning smaller weights to them in clustering. However, all of the above methods adopt the basic idea of traditional k-means clustering to cluster. When data set goes beyond Giga-, Tera- or Peta- bytes, those methods become too much time-consuming or even can not be used because of single memory's limitation. While in our proposed model, minimum spanning tree clustering can eliminate noise by cutting long edges with the number of nodes under a threshold. In addition, our MST clustering algorithm benefits from distributed system and parallel computing

by using MapReduce framework and Hadoop's distributed file system (HDFS).

2.3 Minimum Spanning Tree for Clustering and Classification

Given an edge weighted graph, minimum spanning tree (MST) is a tree spanning all the vertices, whose total weight is minimal. It has been extensively researched and is widely used in many areas. Here we briefly state its usage in two branches of data mining.

MST for clustering Minimum spanning tree has been used for clustering in some applications[29, 24, 22, 9]. It is a variation in the family of clustering algorithms based on graph theory. The purpose of clustering algorithm based on graph theory is to take advantage of the simplicity of tree structure, which can facilitate efficient implementations of much more sophisticated clustering algorithms. It is widely used in the field of computer vision where the data are all in very high dimension space. In general, the idea of graph algorithm is as follows: firstly, it constructs a weighted graph upon the points in the X-dimensional space, with each point being a node, and the similarity/distance value between two points being the weight of the edge connecting the two points. Then, it decomposes the graph into connected components (e.g. subtrees) in some way, and calls those components as clusters. As MST based clustering algorithm, it does not depend on detailed geometric shape of a cluster, it overcomes many of the problems faced by classical clustering algorithms. Figure 1 shows that the tree can represent a non-convex cluster more accurately than a center.



(a) Center representation (b) Tree representation

Figure 1. White dots are objects. (a) Using the center (red dot) of these objects to represent it. (b) Using MST to represent this cluster

MST for classification Minimum spanning tree can also be used for classification. Piotr Juszczak et al. propose a minimum spanning tree based one-class classifier in [11]. This classifier builds on the structure of the minimum spanning tree constructed on the target training set only. The classification of a test object relies on its distance to the closest edge of that tree, hence the proposed method is an example of a distance-based one-class classifier.

Our proposed model combines clustering based classification with MST clustering and MST classification. It aims to take advantage of them all. More importantly, we use MapReduce distributed programming framework, which enables our model to handle massive data efficiently in a distributed way.

3 MST Classification Model for Massive Data with MapReduce Framework

In this section, we present how to use MST clustering algorithm to find clusters of the training set, shrink these clusters to reduce the computational complexity and apply these MST clusters to classification.

3.1 Definitions

For a training set, objects from the same class tend to be spatially close in the data space. By clustering on the training data, objects in the same cluster have similar behaviors or properties and tend to be in the same class [10]. The distances between every two objects are calculated by a distance metric function, which is also used in the final classification phase. There are many distance metrics, such as Euclidean distance, Cosine distance, Hamming distance, Manhattan distance, Tanimoto distance, etc. Usually the choice of distance metric has a great impact on the classification accuracy.

Let X be a training set of n labeled objects. Each object in X has m attributes and a label suggesting its class. Without loss of generality, missing values of attributes are permitted.

Definition 1. A MST-clustering forest of X is a partition of X into k sets $T_1,\cdots,T_k$, where T_i is a minimum spanning tree connecting all the nodes within it, which satisfies:

$$T_i \neq \varnothing, i = 1,\cdots,k \ with \bigcup T_i = X \ and \ T_i \bigcap T_j = \varnothing$$

Definition 2. The dominant class of a MST is the class that the majority of nodes are labeled to. And the tree is

labeled by dominant class.

Definition 3. The first principal path of a MST is the path between two vertices that yields maximum length. The second principal path is obtained by excluding edges from the first principal path and finding the longest path in the remain, and so on [11].

By using N principal paths, the tree representation of the data can therefore be simplified by considering only a few principle paths. Figure 2 shows the first and second principal paths in a MST.

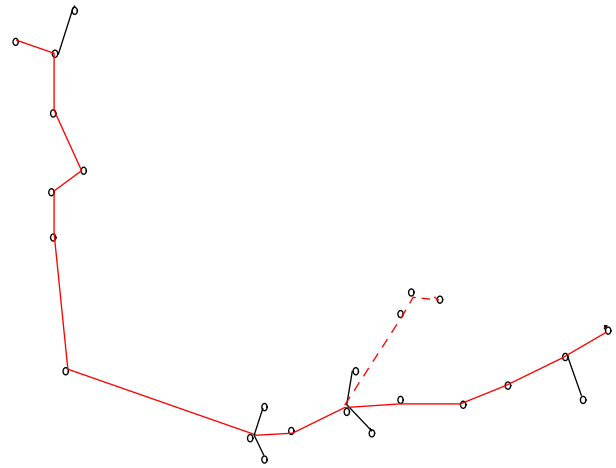


Figure 2. The first principal path of the MST is marked by red solid line, and the second principal path is marked by orange dashed line

Definition 4. The key points of a MST are the representative points in dense parts and backbone points. Here dense part means that all nodes in this part can reach each other within a predefined short distance and there are a variety of ways of choosing representative points, such as the one with most neighbors. Backbone points are those whose neighbors are all far from them.

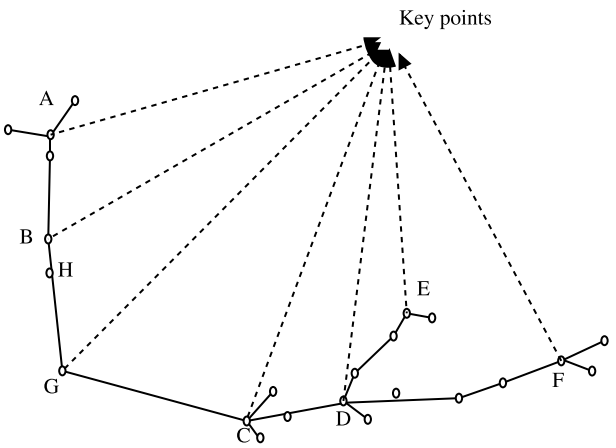


Figure 3. (a)A,C,D,E,F are in the dense part of the tree because they can reach all their neighbors in a short distance; B or H is also a key point, because the dense part can be confined within {BH}. (b)G is a backbone point, since its neighbors are all far from it

By using key points, the tree representation of the data can therefore be simplified by considering only key points. Figure 3 shows the key points in a MST.

In general, a MST-clustering forest of train set can be used for classification. It is actually a cluster based classification model, only the representation of a cluster is a tree. However for a training set of large quantity of objects, it is time-consuming to include all the nodes for classification. So using some kind of way to shrink every MST is important. N principal paths or key points are the choices which we select in our proposed model.

3.2 Generating MST-clustering forest

This is the most challenging part in our modeling process, since the traditional algorithms for finding MST in a graph are not applicable when the numbers of edges are huge. We implement a distributed algorithm to find MST and then construct MST-clustering forest with MapReduce. The implementation details will be stated in section 4. This section can be further decomposed into the following parts:

Calculating similarity matrix For a training set X of n objects, its similarity matrix is:

	0	1	…	(n-1)
0	d_{00}	d_{01}	…	$d_{0(n-1)}$
1	d_{10}	d_{11}	…	$d_{1(n-1)}$
…	…	…	…	…
(n-1)	$d_{(n-1)0}$	$d_{(n-1)1}$	…	$d_{(n-1)(n-1)}$

where d_{ij} is the distance between objects i and j. The function calculating d_{ij} is determined by the chosen distance metric.

Finding MST in the graph The cost of constructing a minimum spanning tree with classical sequential algorithms is O(mlog(n)) [21], where m is the number of edges in the graph, n is the number of vertices. More efficient algorithms for constructing MST have also been extensively researched in [16, 14, 7]. These algorithms promise close to linear time complexity under different assumptions. However there is no guarantee that they can be efficient under any condition. With the increase of vertex number, sequential algorithm also faces the problem of memory limitation. To fix this, many parallel or distributed algorithms are put forward [6, 2, 5]. But most of them are too complicated to implement due to too

many message-passings and perform well only on special graph with regular structure[5]. Moreover, traditional parallel algorithms have specific requirements on the machine they run on, such as SMP or supercomputer, which is not always available. For a distributed algorithm running on traditional distributed system, the overhead of sending messages between processors, including time cost and bandwidth limitation, facilities and time for synchronization, may reduce the performance of algorithm severely. Furthermore, programmer should possess special knowledge when implementing parallel algorithm in the above environment. The most prevalent model for writing parallel algorithms is PRAM, in which an arbitrary number of processors, sharing an unboundedly large memory, operate synchronously on a shared input to produce some output. However, building a large computer with a large robust shared memory is rather difficult and actually fully shared memory machines with large numbers of processors do not exist today.

For all these reasons, there is still large room with respect to optimizing the algorithm of finding MST, especially distributed algorithms, as the data needed to be processed has been growing rapidly. It has been demonstrated that a large class of PRAM algorithms can be efficiently simulated via MapReduce. In our classification model, we adopt a novel distributed algorithm to generate MST using MapReduce framework which is presented in [15]. It can compute MST of a dense graph in only two rounds, as opposed to log *n* rounds needed in the standard PRAM model[15]. The strength of MapReduce lies in the fact that it uses both sequential and parallel computation. In addition, it runs on Hadoop cluster, which can be set up by commodity machines with the installation of Hadoop related software (actually it is a platform for cloud computing). Therefore there is no need for SMP, or supercomputer etc.

Denote the graph, vertex set, and edge set by G, V, and E. The procedure of generating MST can be described as follows [15]:

Step 1: partition the vertex set V into k equally sized subsets: $V = V_1 \bigcup V_2 \bigcup \cdots \bigcup V_k$, with $V_i \bigcap V_j = \varnothing$ for $i \neq j$ and $|V_i| = N/k$. For every pair (i,j), let $E_{i,j} \subseteq E$ be the edge set induced by vertex set $V_i \bigcup V_j$, that is $E_{i,j} = \{(u,v) \in E \mid u,v \in V_i \bigcup V_j\}$, and denote the resulting sub graph by $G_{i,j} = (V_i \bigcup V_j, E_{i,j})$;

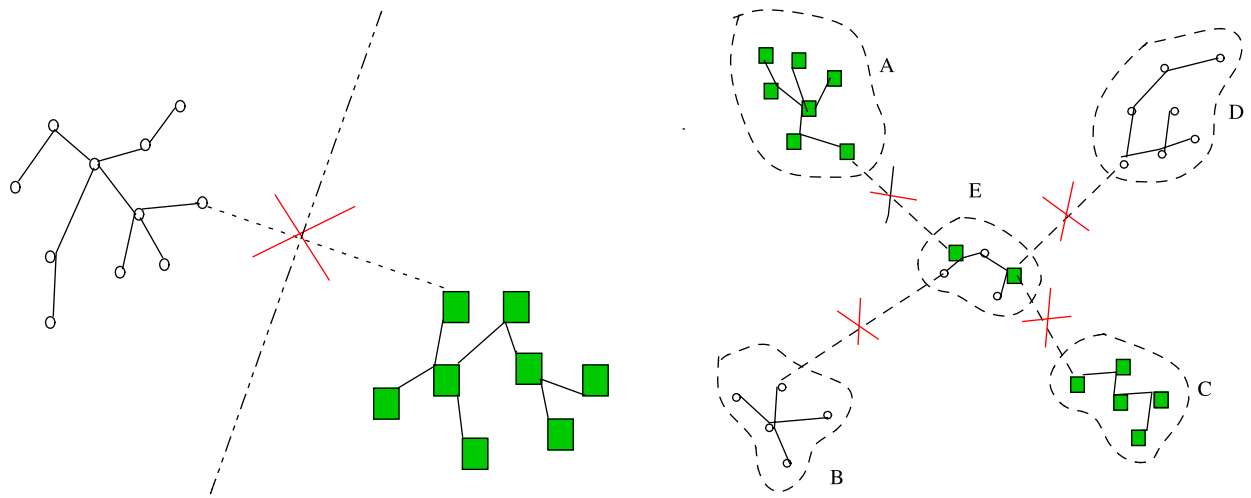
Step 2: for each of the $\binom{k}{2}$ sub graphs $G_{i,j}$, compute the unique minimum spanning tree $M_{i,j}$. Then let H be the graph consisting all of the edges present in $M_{i,j}$, so $H = (V, \bigcup_{i,j} M_{i,j})$;

Step 3: compute M, the minimum spanning tree of H.

The correctness of the above algorithm is proved in [15].

Cutting long edges to get MST forest of the graph

In the case of clustering with MST, in order to produce



(a) There are objects of two classes and they have a clear boundary. If we cut the longest edge (on which we put a cross), we can get two clusters. Each of them is composed of objects from the same class. This is the final result
(b) The objects from two classes have some overlap. So even there are only two classes, we can not build a model of only two MST clusters. More edges should be cut to form purer clusters, on which we put a cross. And the truly mixed area E can be removed. We get A;B;C;D four MST clusters finally

Figure 4. Two classes of objects with and without clear boundary

clusters after MST of the whole graph is generated, we can sort the edges of MST in descending order, and remove the first $k-1$ longest edges [29, 1]. The value of k should be preset and usually it is pretty difficult, because the number of classes is unknown.

However, in the case of classification, k can be set to ($C-1$) initially, where C is the number of classes. At best, when each class is separated from each other by a clear boundary, the cutting phase can stop (see Figure 4(a)). However, in some cases, classes are mixed inherently, such as the case in Figure 4(b). In order to adapt our model to it, the cutting phase of our model is as follows:

Step 1: cut ($k-1$) longest edges to produce k subtrees, where k is the number of classes.

Step 2: for each subtree T_i, calculate its purity P_i and total count TC_i .

If $P_i < Purity$, store T_i and remove it from clusters

Else if $TC_i \leq IsolateNum$, (it may be in an mixed area, which is no good for classification), delete this MST ;

Else cut the longest edge, and go to the beginning of Step 2.

IsolateNum is a preset integer which denotes the vertex number of the smallest subtree. Usually it's a very small value, i.e. one or two, for the purpose to eliminate truly mixed MST. P_i is calculated by Nmc_i / TC_i, where Nmc_i is the number of objects with majority class in T_i, and *Purity* is a preset value to control the accuracy of the model. The larger *Purity* is set, the less non-majority class objects in a subtree will be. After the above operations, we get the MST forest of the original graph. Note that the total count of all MST forests' vertices may be less than that of the graph because of the elimination of mixed clusters in Step 2.

3.3　Shrink MST in the MST-clustering forest

MST clustering forest can describe clustering structure of a graph, especially for a non-convex cluster. But sometimes it's unnecessary to include all the vertices of the MSTs into the classification model. Some representative vertices could be enough. Otherwise, a model with complete vertices may reduce efficiency or have the problem of over fitting to the training set. Thus we can adopt either one of the following two methods to shrink MST, with the goal of eliminating some unnecessary vertices.

Using N principal path The definition of N principal path has been given by definition 3. The tree representation of data can be simplified by using a few principal paths. The algorithms 1 and 2 can be used to find the first principal path [3].

Let G=(V;E;w) be a graph. For a vertex v, the eccentricity of v is the maximum of the distance to any vertex in the graph, which can be computed by algorithm 1.

Algorithm 1 : Eccent(r)

Input T_r : a tree T_r = ($V;E;w$) rooted at r

Output *Eccent*(r): the eccentricity of r in T_r ;

```
if r is a leaf then
    return 0;
end if
for each child s of r do
    compute Eccent(s) recursively;
end for
return Eccent(r)=max(Eccent(s) + w(r, s)).
```

Lemma 1. Let r be any vertex in a tree T. If v is the farthest vertex to r, the eccentricity of v is the length of the longest path (first principal path) of T.

Algorithm 2 : First Principal Path

Input T_r : a tree T = ($V;E;w$)

Output *PPath*: the first principal path of T;

```
root T at an arbitrary vertex r;
Use Eccent to find the farthest vertex v to r;
Root T at v;
Use Eccent to find the eccentricity path of v;
for each vertex vi along eccentricity of v do
    PPath.add(vi);
end for
return PPath;
```

Lemma 1 has been demonstrated in [1]. Algorithm 2 uses this property to find the first principal path of a tree.

The second principal path can be calculated by using Algorithm 2 after deleting the edges in first principal path, and so on.

Using Key Points Apart from N principal paths policy, we also applied key points method (see definition 4) to our classification model. The following procedure describes how to find key points in a tree.

Denote the tree of which we want to find key points by T,

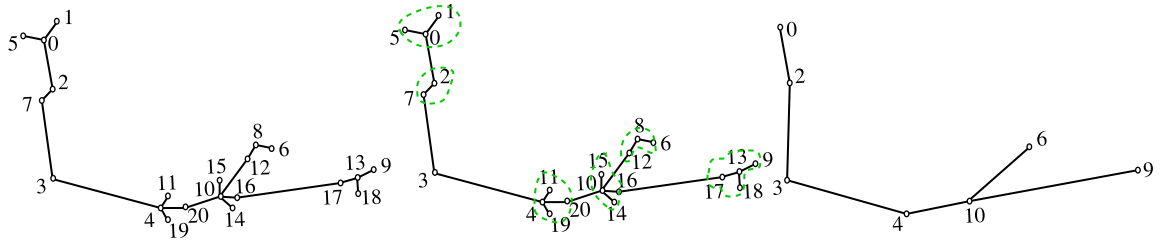


Figure 5. Process of finding key points

which has k vertices.

Step 1: label the T's vertices with integers 1, 2 $\cdots k$.

Step 2: collect the edge of the tree whose weight is smaller than a predefined value. These edges are organized by neighborhood relationship, and then we get several neighborhood lists. All vertices within a list are neighbors and close enough to each other.

Step 3: for each neighborhood list, we preserve the vertex with the smallest label.

Step 4: for edges whose weight is larger than the predefined value, we preserve both ends.

Figure 5 illustrates this process. Figure 5(a): first label the vertex of a tree by integers, suppose there are 21 vertices, which are labeled by integer from 0 to 20. Figure 5(b): collect edges whose weights are smaller than a threshold, which are (0,1), (0,5), (2,7), (4,11), (4,19), (4,20), (10,14), (10,15), (10,16), (6,8), (8,12), (9,13), (13,17), (13,18), where an edge is presented by its two end nodes in a bracket. We put the node with smaller label in the front, just for consistency. These edges are organized by neighborhood relationship: (0,1,5), (2,7), (4,11,19,20), (10,14,15,16), (6,8,12), (9,13,17,18). Figure 5(c): within each neighborhood relationship, use the node with the smallest label is used to represent all its neighbors. Hence we get 0,2,4,6,9,10. Node 3 is also collected. Although it's not collected in (b), it's probably a representative node of the tree since all of their neighbors are far.

If an algorithm needs a preset value, usually this is a tricky part, such as the value of k in k-means clustering. Recall in step 2, a preset value is required. There is no strict standard on this setting, but we suggest it to be set to 1/2 to 1/3 of the largest edge weight initially and later to be adjusted to control the quantity of key points.

Note that the algorithm for finding key points proposed above may not be an optimal one. However it's straightforward and can reflect the backbone of a tree to some extent. In addition, it can be implemented with MapReduce framework by simply adjusting the format of input file which represents the tree. For each subtree in the MST-clustering forest, either principal path or key points policy can be adopted to shrink it. In our classification model, we use both respectively and compare them. The result can be viewed in section 5.

3.4 Classification

The basic idea of classification is inherited from KNN. First, we compute the distance between unlabeled object and each subtree in MST forest. These subtrees are different from the ones which are directly generated by MST clustering, because they have been processed by the shrinking policy described previously. Then we find the shortest distance and corresponding subtree T_i. The unlabeled object is classified to the class of T_i. Although the classification idea is straightforward, there are two parts needing further explanation: distance metric selection and distance definition.

In our model, we decide to use Euclidean distance for numerical attribute and Hamming distance for categorical distance, after comparisons with several other distance metrics, including Cosine distance, Manhattan distance and Tanimoto distance. The distance from a point to a tree can be defined as the smallest distance between the point and all edges of the tree [11]. However, this distance involves both the computing of projection and point-to-point distance. Thus it is too complicated for practical use, especially when the dimension is very large. We consider the distance between x and tree T_i as the min (distance(x; x_i)), where x_i is a node of tree T_i. And in order to speed up the classification, we implemented it with MapReduce.

4 MapReduce Implementation

In this section, the details of how to use MapReduce

framework to implement our model and classify objects are presented. Since we have described the related algorithms in previous sections, here we only focus on implementation.

4.1 Finding MST with MapReduce

This phase can be further divided into the following steps.

Step 1: Generating similarity matrix of the training set.

Step 2: Finding MST in the undirected complete graph corresponding to the similarity matrix.

The power of MapReduce framework lies in its ability of distributed computing. In order to benefit from this, as we have illustrated in section 3, we should properly partition the input file, allowing each map to operate on a partition of more or less the same size. Considering a graph can be expressed by a matrix, we can partition the matrix by row and column to several blocks and control the size of each block by defining the total number of the blocks. Assume there are n objects in the training file and we labele them by node IDs from 0 to (n−1). The matrix below shows how each object is re-labeled with partition ID if the training set is partitioned into k parts.

	0⋯ *n*/*k*-1	*n*/*k*⋯2n/k-1	…	(k-1)*n*/*k*⋯n-1
0⋯*n*/*k*-1	[0][0]	[0][1]	…	[0][k-1]
n/*k*⋯2*n*/*k*-1	[1][0]	[1][1]	…	[1][k-1]
…	…	….	…	…
(k-1)*n*/*k*⋯n-1	[k-1][0]	[k-1][1]	…	[k-1][k-1]

The first row and column of the matrix is the node IDs. They are divided into k groups, and these groups are separated by solid lines. Assume the intersection of a row and a column is an edge induced by the row node and column node. Then the solid lines between groups can partition the edges into k^2 blocks, with the partition ID displayed in the above matrix. The graph we use is an undirected complete graph, so actually it is enough to consider the upper triangular part. Denote partition ID by PID, which is in the form of [Row Element][Column Element] as shown in the above figure, then edge e_{ij} will go to partition pid, if $pid.[RowElement] \equiv \lfloor ik/n \rfloor$ and $pid.[ColumnElement] \equiv \lfloor jk/n \rfloor$. This can be accomplished by one map method.

(*Inputkey, Inputvalue*) : (*ID, information*)

$$(OutputKey;OutputValue\ information) : \begin{cases} ([ID * k=n][j]; information) \\ ([j][ID * k=n]; information) \end{cases} \quad j=0,1,\cdots,k$$

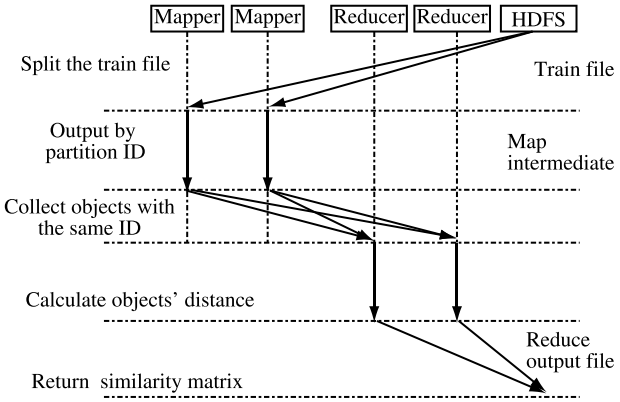


Figure 6. Generating similarity matrix

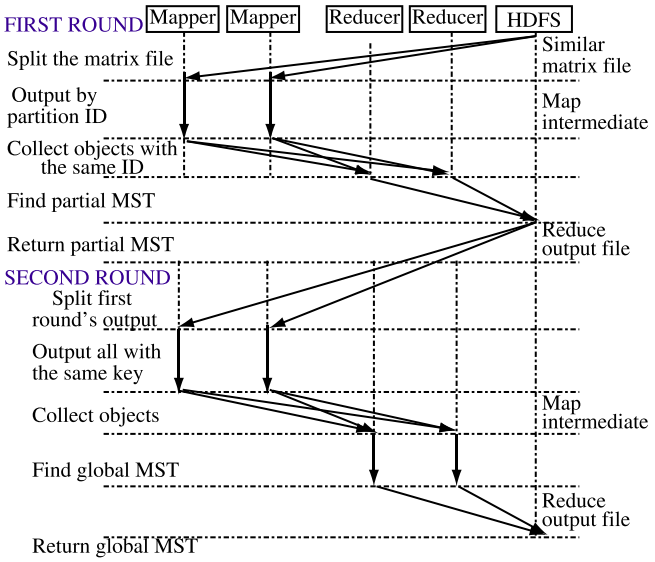


Figure 7. Finding MST

When generating similarity matrix, the corresponding reduce method will calculate distances between nodes with the same PID. One PID is processed by one reduce.

(*OutputKey, OutputValue*) : (*PID*, (*StartNodeID EndNodeID Distance*))

where (*StartNodeID EndNodeID Distance*) is actually the edge in the undirected complete graph, which will be used for finding MST in the following step (see Figure 6).

The next step is generating MST. It consists of two rounds. During the first round, map method will pass the key and value from previous job to the reduce after certain processing, and the corresponding reduce method will use Kruskal's algorithm to find MST within edges with the same PID. One PID is processed by one reduce. (*OutputKey, OutputValue*) : (*PID*, (*StartNodeID EndNodeID Distance*)) where (*StartNodeID EndNodeID Distance*)) is actually the MST edge. In the second

round, map collects all the partial MSTs' edges by setting the PID to the same value, and the reduce does the same thing as the first round (see Figure 7).

4.2 Finding key points in the MST with MapReduce

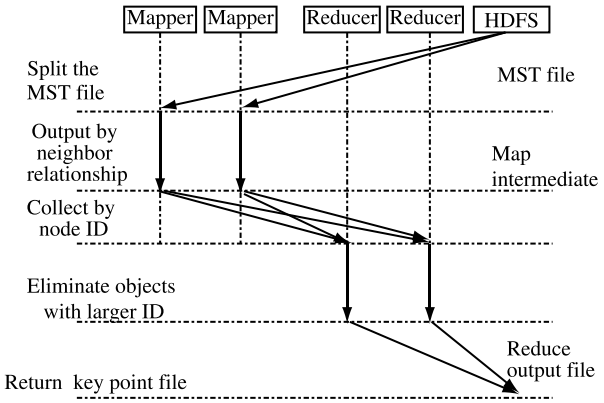


Figure 8. Finding key points

Map

The input (key, value) pair is: ("*MST*", (*StartNodeID EndNodeID Distance*)), which is the output of the previous MST generation job. The output (key, value) pair is:

For the edge whose weight is smaller than a threshold:

(*StartNodeID EndNodeID*) and (*EndNodeID StartNodeID*)

For the edge whose weight is larger than a threshold:

(*StartNodeID EndNodeID*) and ((*EndNodeID*, MAX_VALUE))

Reduce

Only collect the key in the (*key, List < value >*) whose key is smaller than all values in List < *value* >. The complete job procedure can be seen in Figure 8.

4.3 Classification with MapReduce

Map

The input file is training set after modeling, i.e. the shrunk MSTs, with the form of (*nodeID nodeInformation&Class*)

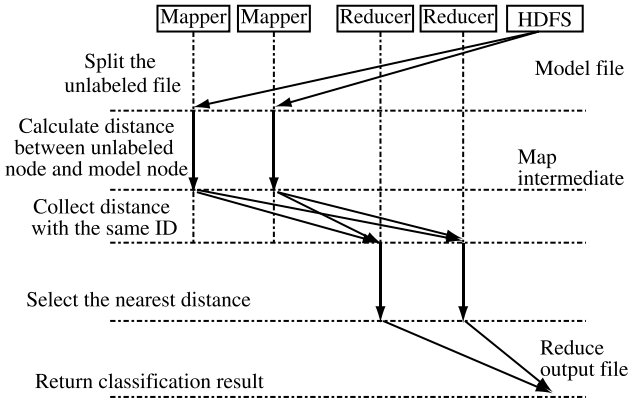


Figure 9. Classification

per line. By using Hadoop API's Text Input Format, input file is passed to map line by line. The distance between this node and unlabeled node is calculated by map. The output (key, value) pair is: (*unlabeledNodeID distance&Class*)

Reduce

According to MapReduce framework, the output of map with the same key is organized as a list, which is passed to a reduce method. The task of reduce method in classification is to select the smallest distance within a list and hence get the class for the unlabeled node. The complete job procedure can be seen in Figure 9.

5 Experiments and Results

In order to verify the accuracy and effectiveness of our proposed model, we have done experiments using data sets from UCI Machine Learning Repository [34]. A brief description of the data sets chosen is listed in Table 1. We try to select data sets of different type, such as with numerical attributes only, with categorical attributes only, with combined attributes, to get a comprehensive conclusion.

Table 1 Data set information

Data Set	Class	attribute number	attribute type(s)	train/test number
Breast Cancer	2	10	Real	450/249
Car Evaluation	4	6	Categorical	1130/578
Credit Approval	2	15	Real, Categorical	450/240
Iris	3	4	Real	90/60
Letter Recognition	26	16	Real	14040/5960
Vote	2	10	Categorical	250/185

Our experiments are completed on Dawning 4000 cluster, which is set up by 10 separate nodes. Each node has eight 2 GHz Dual Core AMD Opteron Processors and 8 G memory, running Linux. And we use the latest version of Hadoop package, hadoop-0.20.2.

5.1 Choosing Distance Metric

Since our classification model can be regarded as an intermediate model between KNN and clustering based classification method, whose accuracies both greatly rely on the choice of distance metric, it's necessary to select a proper metric. Although most distance based algorithms use Euclidean distance, there is no guarantee that it performs well in every model. There has been a lot of study on distance metric learning [28, 25]. But for simplicity and practicability, we only choose several commonly used basic metrics for numerical attributes. We use Hamming distance for categorical attribute. The MST shrinking policy we choose is principal path. The result is given in Table 2. From above we can draw the conclusion that Euclidean distance outperforms others on a general basis, which can be chosen as the distance metric for our model. The entry which is labeled as bad means that it's very low and there is no need to list it.

Table 2　Accuracy for different distance metric

Data Set	Euclidean	Manhattan	Cosine	Tanimoto
Breast Cancer	0.9476	0.9679	0.6426	0.1454
Car Evaluation	0.6765	0.5882	0.5862	0.5862
Credit Approval	0.7750	07250	0.3312	0.6375
Iris	0.9167	0.9167	0.1245	0.1221
Letter Recognition	0.6970	0.7133	0.4213	0.3997
Vote	0.9459	0.9405	0.2326	0.2258

5.2 Comparison of Reduction Rates

By using the MST-shrinking methods we proposed in the previous section, we can significantly reduce the number of samples used for classification compared with the case in KNN, which should include all of the samples in training set. Table 3 shows the reduction rate of our model compared with KNN. The first column of the table indicates the shrinking policy. Each row lists the number of remaining training samples and the reduction rates (in brackets) in our models after shrinking policy. The row begins with "KNN" actually lists the number of points in the training set.

Table 3　Reduction rates

	Breas Cancer	Car Evaluation	Credit Approval	Iris	Letter	Vote
Key Point	110(75.6%)	102(91%)	81(82%)	32(64.4%)	3874(72.4%)	54(78.4%)
Principal Path	113(74.9%)	617(45.4%)	200(55.6%)	45(50%)	7423(47.1%)	50(80%)
KNN	450	1130	450	90	14040	250

5.3 Accuracy of MCMM

In Table 4, the accuracies of MCMM on six different data sets are shown. The description of the six data sets and how they are partitioned to training and test set are given in Table 1. Note that MCMM adopts the idea of clustering based classification method, which uses MST to clustering the whole training set first and then cut long edges to form MST clustering forest. However, as mentioned above, at the best case all objects from the same class are in the same MST because they are closer to each other than to the ones of a different class. Since we know the class of every object in the training set, an alternative to building the model is constructing only one MST for all objects of the same class, and then perform the similar operation as MCMM. We call this mode SMCMM, meaning Separate MCMM. In Table 4, the row begins with "Separate MST" is the accuracy of SMCMM, and the row begins with "Global MST" is the accuracy of MCMM.

In Table 4, we can see that for shrinking MST, the policy of key points is better than principal path policy, but not significantly. When it comes to the way of constructing MST clustering forest, the accuracies of the two have no

Table 4　Accuracy of MCMM and SMCMM

	Key Points						First Principal Path					
	Breast Cancer	Car Evaluation	Credit Approval	Iris	Letter	Vote	Breast Cancer	Car Evaluation	Credit Approval	Iris	Letter	Vote
Separate MST	0.9438	0.8327	0.75	0.9333	0.9307	0.9243	0.9476	0.6765	0.775	0.9167	0.6970	0.9459
Global MST	0.9759	0.7977	0.7857	0.95	0.8998	0.8919	0.9839	0.7093	0.8083	0.9333	0.8844	0.9081

Table 5 Accuracy of algorithms from Weka

	Breast Cancer	Car Evaluation	Credit Approval	Iris	Letter	Vote
Naive Bayes	0.9759	0.5920	0.775	0.9833	0.6364	0.9081
KNN	0.9719	0.7676	0.7583	0.95	0.955	0.9005
BFTree	0.9871	0.7007	0.8167	0.95	-	0.9243
NNge	0.9839	0.5618	0.7917	0.9667	-	0.9459
StackingC	0.6345	0.6856	0.5542	-	-	-
FLR	0.9116	-	-	0.95	-	-

significant difference. This may be because that the data sets we choose are inherently convex, and the advantage of global MST clustering isn't shown clearly. For comparison, we apply some other common classification algorithms in Weka [33] to the six data sets, and result is shown in Table 5.

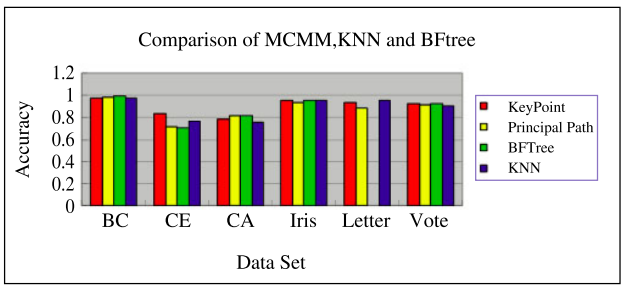


Figure 10. Horizontal axis lists the data set name (BA=Breast Cancer, CE=Car Evaluation, CA=Credit Approval), and vertical axis represents the accuracy

From the comparison of Table 4 and 5, we can draw the conclusion that our model is better and has no significant difference from the best algorithm which we adopt from weka regarding to accuracy. Note that, the BFTree algorithm and KNN outperform other traditional classification algorithms in weka in general, and our model has similar accuracy to it, if not better. From Figure 10 we can get a more direct view of comparison. However, BFTree is decision tree-based classification algorithm. When the size of training set is very large, the memory can not store the whole tree structure and hence can not be used for classification, just as the case of "Letter Recognition" data set. And from Figure11 we can see that our model has greatly reduced the number of objects used for classification compared to KNN. For example, in the case of "car evaluation", 90% of the training objects have been removed in our model, yet it yields higher accuracy than KNN, which needs all of the training objects for classification.

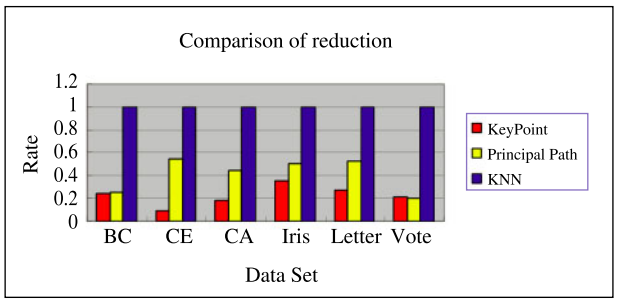


Figure 11. zzHorizontal axis lists the data set name (BA=Breast Cancer, CE=Car Evaluation, CA=Credit Approval), and vertical axis is the percentage of the object number in classification model against whole training set. KNN use all objects in the training set, hence the height of its rectangle is always 1

5.4 Testing on Large Data

So far, we have described the prototype of MCMM and presented some of its features by analyzing experiment results. However, we haven't referred to one of its most notable features—the ability to deal with massive data in a distributed way by using MapReduce framework. In this part, we will discuss it in detail.

To better test the scalability and efficiency of our model on large data, we developed a data generator, as it can produce data of various sizes. And the format of data is similar to weka's .arff file, which is one record per line, and each line contains the record's all attributes and label, separated by commas. Here we generated data set record of six real types and one class attribute and there are four kinds of classes in all. For example:

1.647, 1.06, 1.78, 1.92, 1.57, 0.39, A

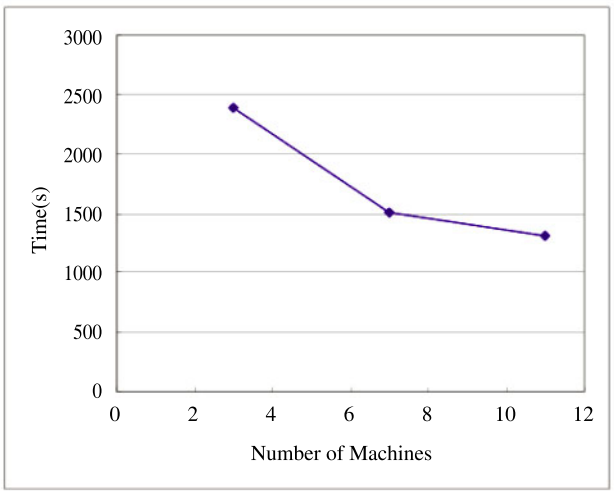
0.012, 0.278, 1.25, 0.453, 0.105, 0.843, B

First we show how the modeling time for MCMM changes as we add more machines.

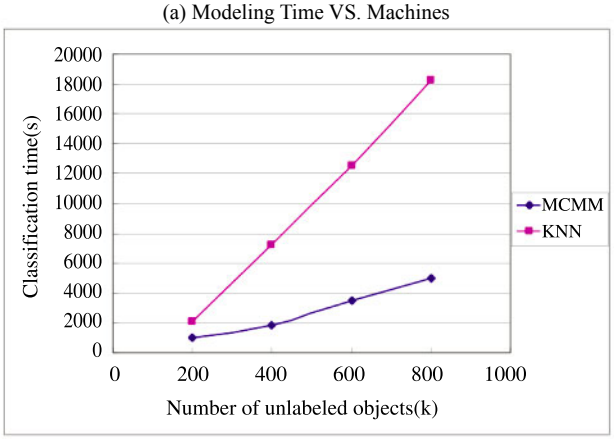
We construct MCMM using a training set of 20000 objects (equivalent to a complete graph with 200002 = 4×108 edges) on 3, 7, 11 machines. Figure12 (a) shows that the time of constructing MCMM decreases as we add more machines. This benefits from the distributed computing of MapReduce framework and indicates our model's scalability.

After the model is constructed, it then comes to its real function — classification. Figure 12(b) shows how the classification time changes with the input size of test set using the MCMM model constructed above. In Figure12

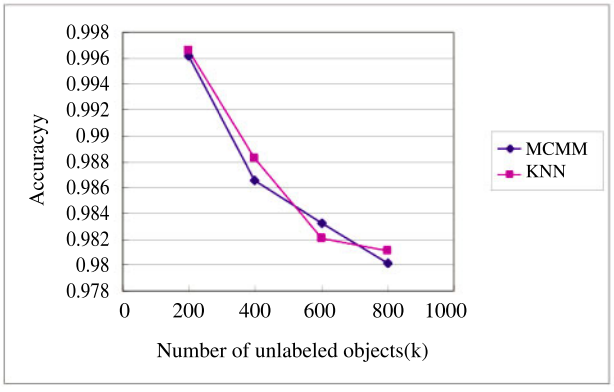
(b) we also give the classification time of KNN using the same training set. Note that we also implement KNN with MapReduce and run it on our Hadoop cluster. It is actually similar to MCMM classification. The only difference is that KNN uses all the objects of training set. However, our MCMM classification uses a subset of training set. We can't use traditional KNN algorithm, since the data used and produced by KNN is too large to fit in a single machine's memory (e.g. we try to classify the same test sets using Weka's KNN algorithm, but always get "memory overflow" error). Figure 12 (b) shows the classification time of MCMM is much better than that of KNN, yet achieving nearly the same accuracy (Figure12(c)).



(a) Modeling Time VS. Machines



(b) Classification Time VS. Number of Objects



(c) Accuracy

Figure 12. (a) Modeling time decreases as we add more machines; (b) MCMM outperforms KNN by a large margin, and achieves nearly the same accuracy (shown in (c))

6 Conclusion

In this paper, we present MCMM, a minimum spanning tree-based classification model with MapReduce, which is an intermediate model between *k* nearest neighbor and cluster-based classification. MST of the training set is computed and by cutting long edges several subtrees are obtained, which are used to represent each cluster. We propose two policies, key points and N principal paths, to cut superfluous edges of the subtrees. Benefiting from this, a more concise model is built and hence classification speeds up.

Another contribution is that we implement the model in a distributed way by using MapReduce framework. Thus our model is capable of dealing with huge amount of data in an efficient way. In addition, the classification phase also uses MapReduce. We run our model on a cluster of ten nodes, installed with the Hadoop package, to test on several data sets from UCI machine learning repository [34]. For comparison, we have also used weka [33] to classify the same data sets. The experiment results show that MCMM has advantage in classifying large data of multiple classes and high dimension, both in accuracy and time. The scalability of MCMM is proved by experimenting on synthetic graphs of different sizes.

A tree based model can be altered by cutting, adding or adjusting some of its edges, without complete information of original data set. So MCMM has the ability of incremental learning and hence may be suitable for stream mining. With rapid growth of stream data and its widely use in many areas, the application of MCMM to this field deserves extensive study in the future.

参 考 文 献

[1] Asano T, Bhattacharya B, Keil M, et al. Clustering algorithms based on minimum and maximum spanning trees [C] // In Proceedings of the 4th Annual Symposiumon Computational

Geometry, 1988: 252-257.

[2] Awerbuch B. Optimal distributed algorithms for minimum weight spanning tree, counting, leader election and related problems [C] // Proceedings of 19th Symposium on Theory of Computing, 1987: 230-240.

[3] Cormen T H, Leiserson C E, Rivest R L. Introduction to Algorithms [M]. The MIT Press: Massachusetts, 1990.

[4] Cover T M, Hart P E. Nearest Neighbor Pattern Classification [J]. IEEE Transactions on Information Theory, 1967, IT-13: 21-27.

[5] Dehne F, Götz S. Practical parallel algorithms for minimum spanning trees [C] // In Workshop on Advances in Parallel and Distributed Systems West Lafayette, 1988: 366-371.

[6] Faloutsos M. Corrections, improvements, simulations and optimstic algorithms for the distributed minimum spanning tree problem [D]. Master's Thesis University of Toronto, 1995.

[7] Gabow H, Spencer T, Rarjan R. Efficient algorithms for finding minimum spanning trees in undirected and directed graphs [J]. Combinatorica, 1986, 6(2): 109- 122.

[8] Gower J C, Ross G J S. Minimum spanning trees and single-linkage cluster analysis [J]. Applied Statistics,1969, 18: 54-64.

[9] Grygorash O, Zhou Y, Jorgensen Z. Minimum spanning tree based clustering algorithms [C] // Proceedings of the 18th IEEE International conference on tools with Arti_cial Intelligence, 2006: 73-81.

[10] Huang Z X, Ng M K, Lin T, et al. An interactive approach to building classification models by clustering and cluster validation [C] // Proceedings of the Second International Conference on Intelligent Data Engineering and Automated Learning, Data Mining, Financial Engineering, and Intelligent Agents, 2000: 23-28.

[11] Juszczak P, Tax D M J, Pekalska E, et al. Minimum spanning tree based one-class classifier [J]. Neurocomputing, 2009, 72:1859-1869.

[12] Kang U, Tsourakakis C, Appel A, et al. Hadi: Fast Diameter Estimation and Mining in Massive Graphs with Hadoop [M]. Carnegie Mellon University, 2008.

[13] Kang U, Tsourakakis C E, Faloutsos C. PEGASUS: a peta-scale graph mining system implementation and observations [C] // Proceedings of the 2009 Ninth IEEE International Conference on Data Mining, 2009: 229-238.

[14] Karger D, Klein P, Tarjan R. A randomized linear-time algorithm to find minimum spanning trees [J]. Journal of the ACM, 1995, 42(2): 321-328.

[15] Karloff H, Suri S, Vassilvitskii S. A model of computation for mapreduce [J]. Symposium on Discrete Algorithms, 2010.

[16] Kruskal J. On the shortest spanning subtree and the traveling salesman problem [J]. American Mathematical Society, 1956: 48-50.

[17] Kyriakopoulou A, Kalamboukis T. Text classification using clustering [C] // Proceedings of ECML-PKDD Discovery Challenge Workshop, 2006.

[18] Li Y, Hung E, Chung K, et al. Building a decision cluster classication model by a variable weighting k-means method [C] // In: 21st Australasian Joint Conference on AI, LNCS, 2008: 337-347.

[19] Liang S S, Liu Y, Wang C, et al. A CUDA-based parallel implementation of knearest neighbor algorithm [C] // International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, 2009: 291-296.

[20] Mui J K, Fu K S. Automated classification of nucleated blood cells using a binary tree classifier [J]. IEEE Transactions on Pattern Analysis and Machine Intel- ligence, 1980, 2(4): 429-443.

[21] Preparata F, Shamos M. Computational Geometry: An Introduction [M]. Springer-Verlag: New York, USA, 1985.

[22] Vathy-Fogarassy A, Feil B, Abonyi J. Minimal Spanning Tree Based Fuzzy Clustering [J]. ENFORMATIKA Transactions on Engineering, Computing and Technology, 2005, 8: 7-12.

[23] Venner J. Pro Hadoop [M].1st Edition, Apress, 2009.

[24] Victor S P, Peter S J. A novel minimum spanning tree based clustering algorithm for image mining [J]. European Journal of Scientific Research, 2010, 40(4): 540-546.

[25] Weinberger K Q, Blitzer J, Saul L. Distance metric learning for large margin nearest neighbor classification [J]. The Journal of Machine Learning Research, 2009, 10(12): 207-244.

[26] White T. Hadoop: The Definitive Guide [M]. O'Reilly Media, 2009.

[27] Wu B Y, Chao K M. Spanning Trees and Optimization Problems [M]. Chapman & Hall/CRC Press: USA, 2004.

[28] Xing E P, Ng A Y, Jordan M I, et al. Distance metric learning, with application to clustering with side-information [C] // Proceedings of Neural Information Processing Systems, 2002: 505-512.

[29] Xu Y, Olman V, Xu D. Minimum spanning trees for gene expression data clustering [J]. Genome Informatics, 2001, 12: 24-33.

[30] Zhang B, Srihari S N. Fast k-nearest neighbor classification using cluster-based trees [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2004, 26(4): 525-528.

[31] Zhou L, Wang L, Ge X, et al. A clustering-Based KNN improved algorithm CLKNN for text classification [C] // 2nd International Asia Conference on Informatics in Control, Automation and Robotics, 2010: 212-215.

[32] Hadoop [EB/OL]. http://hadoop.apache.org/.

[33] WEKA. Weka 3: Data mining software in java [EB/OL]. http://www.cs.waikato.ac.nz/ml/weka/.

[34] Hettich S, Blake C L, Merz C J. UCI repository of machine learning databases [EB/OL]. http://www.ics.uci.edu/ mlearn/MLRepository.html.