The k-Nearest Neighbor Algorithm Using MapReduce Paradigm

Prajesh P Anchalia  
Department of Computer Science and Engineering  
RV College of Engineering  
Bangalore, India.

Kaushik Roy  
Department of Computer Science and Engineering  
RV College of Engineering  
Bangalore, India.

Abstract— Data in any form is a valuable resource but more often than not data collected in the real world is completely random and unstructured. Hence, to utilize the true potential of data as a resource we must transform it in such a manner so as to retrieve meaningful information from it. Data mining fulfills this need. Today there is not only a need for efficient data mining techniques to process large volume of data but also a need for a means to meet the computational requirements to process such huge volume of data. In this paper we implement an effective data mining technique known as the k-Nearest Neighbor method on a distributed computing environment running Apache Hadoop that uses the MapReduce paradigm to process high volume data.

Keywords— k-Nearest Neighbor, Distributed Computing, Hadoop, MapReduce, Data Mining.

I. INTRODUCTION

Given the huge volume of data these days it is almost impossible for human analysts to derive meaningful conclusions in a short time frame. Hence data mining techniques are looked upon as tools that can be used to automate the process of knowledge discovery and define relationships and patterns of resemblance given a completely random and raw data set. The majority of data collected for analysis is unsupervised. This gives rise to the need of effective techniques that can process such unsupervised data sets and convert what might seem to be completely random and meaningless into something sensible and valuable. For instance, given a collection of hourly or daily temperatures of a place, it is possible to determine the demography of that place by grouping the temperatures based on a similarity metric. Here, we use what might seem to be a random set of temperatures to determine the demography of a place by simply grouping temperatures and finding the mode temperature of each group. Hence processing data and data mining techniques play a critically important role in technological development.

Data collected is not only more often than not random but also voluminous. This poses the problem of meeting computational demand. The problem is addressed efficiently by distributed computing. Distributed computing deals with hardware and software components that are physically separate but communicate and co-ordinate with each other and work like a single system on the whole. Hence, we divide the processing of huge volume data amongst computers that are networked and work together towards a common goal.

Distributed computing environment basically consists of computers connected over a network and interact with each other constantly to achieve a common goal. Individually each computer connected on the network is called a node and collectively all nodes form a cluster.

Apache™ Hadoop[1][2][3] is an open source application that runs on a distributed computing environment and supports processing of high volume data intensive applications. It has evolved from The Google File System[4]. Hadoop uses MapReduce[3][7] programming style that gives it the flexibility and capabilities required to process petabytes of data. Since Hadoop uses the MapReduce paradigm it achieves its goal of processing by downscaling the given data set and consequent integration of the data processed individually at separate nodes that are networked to form a cluster. Hence the computational demands are met collectively by all the nodes that form the cluster and this gives rise to increase in efficiency which is in harmony with the low costs involved in setting up the cluster.

This paper contains 8 sections. The First section gives an introduction to data mining, Hadoop and Distributed Computing. The Second section gives a detailed explanation of Data Mining and popular Data Mining techniques. The Third section describes the K-Nearest Neighbor Technique. The Fourth Section describes the MapReduce Paradigm. The Fifth section talks about The Hadoop Framework and its functioning. The sixth section gives the Algorithmic Design of the K-Nearest Neighbor Technique. The Seventh section illustrates the experimental results and the observations from the implementation of the KNN Algorithm. The Eighth section concludes the paper.

II. DATA MINING AND DATA MINING TECHNIQUES

A. Definition of Data Mining

Data mining is a method that automates the detection of relevant patterns and relationships given a dataset. It uses defined approaches and algorithms to scan the given data set and predict the data trend. Data Mining is not particularly new. Statisticians have been using similar manual approaches to review data and propose a trend. However, what has changed is the volume of data today has increased immensely and thus giving rise to automated data mining techniques that investigate data trends very quickly. Users can also determine the outcome of the data analysis by the parameters they choose, thus automated data mining gives users such flexibility.

B. Data Mining Techniques

Data Mining has a broad spectrum of application and so there are may techniques of data mining in existence. Most commonly used techniques are Artificial Neural Networks,
Decision Trees and The Nearest-Neighbor method. Each of these techniques analyzes data differently and is discussed below.

- **Artificial Neural Networks:** These are non-linear predictive models that learn through training. The fundamental processing element of a neural network is a neuron. This building block of human awareness encompasses a few general capabilities. Basically, a biological neuron receives inputs from other sources, combines them in some way, performs a generally nonlinear operation on the result, and then outputs the final result. This model is replicated electronically forming the Artificial Neural Network.

- **Decision Trees:** These are tree-like graphs or models of decisions and their possible outcomes. They consist of flow-chart-like structure that has nodes to represent a test on attributes and the branches of the node represent the outcome of the test. They are often used in data analysis as they depict every possible scenario pictorially and hence patterns can easily be identified in a data set.

- **Nearest Neighbor:** This method classifies data based on a majority vote of its neighbors and it is assigned to the class most common amongst all its neighbors. This method is discussed in detail in later sections of this paper.

III. THE k-NEAREST NEIGHBOR METHOD

A. An Introduction

kNN is a non-parametric lazy learning algorithm. Being a non-parametric algorithm it does not make any assumptions on the underlying data distribution. This is a major advantage because majority of the practical data does not obey theoretical assumptions made and this is where non-parametric algorithms like kNN come to the rescue. kNN is also a lazy algorithm this implies that it does not use the training data points to do any generalization. So, the training phase is pretty fast. Lack of generalization means that kNN keeps all the training data. kNN makes decision based on the entire training data set.

The k-Nearest Neighbor Algorithm finds applications in some of the fascinating fields like Nearest Neighbor based Content Retrieval, Gene Expressions, Protein-Protein interaction and 3-D Structure predictions are to name a few.

B. Requisites for k-Nearest Neighbor Algorithm

- kNN assumes that the data is in a feature space. More exactly, the data points are in a metric space.
- The data can be scalars or possibly even multidimensional vectors. Since the points are in feature space, they have a notion of distance – This need not necessarily be Euclidean distance although it is one of the commonly used methods.
- The kNN uses training data as reference to classify the new data points collectively called testing dataset
- Each of the training data consists of a set of vectors and class label associated with each vector. In the simplest case, it will be either + or – (for positive or negative classes). But kNN, can work equally well with arbitrary number of classes.
- We are also given a single number ‘k’. This number decides how many neighbors (where neighbors is defined based on the distance metric) influence the classification. This is usually an odd number if the number of classes is 2. If k=1, then the algorithm is simply called the nearest neighbor algorithm.

C. The k – Nearest Neighbor Algorithm

The k-Nearest Neighbor Algorithm involves two phases.

- The Training Phase
- The Testing Phase

These Phases are discussed in detail in the following sub-sections.

1) The Training Phase

Given data points for testing, our aim is to find the class label for the new point. The algorithm is discussed for k=1 or 1 Nearest Neighbor rule and then extended for k=k or k-Nearest Neighbor rule.

a) $k=1$ or 1 Nearest Neighbor

This is the simplest scenario for classification. Let ‘x’ be the point to be labeled.

- Find the point closest to ‘x’ in the training data set. Let this closest point be ‘y’.
- Now nearest neighbor rule asks to assign the label of ‘y’ to ‘x’.

This seems too simplistic and some times even counter intuitive. This reasoning holds only when the number of data points is not very large.

If the number of data points is very large, then there is a very high chance that label of x and y are same.
MapReduce is best suited to deal with large datasets and therefore ideal for mining larger datasets of petabytes size that do not fit into a physical memory.

V. THE HADOOP FRAMEWORK

Hadoop is an open source software framework that can run large data-intensive, distributed applications. Hadoop comes with its own file system called the Hadoop Distributed File System (HDFS) and a strong infrastructural support of managing and processing huge petabytes of data.

A HDFS cluster consists of one unique server known as the Namenode that manages the namespace of the file system, determined the mapping of blocks to Datanodes, and regulates file access. Each node in the HDFS cluster is a Datanode that has the task of managing the storage it holds. It is these Datanodes that serve the client’s read/write requests and perform block operations, creation, deletion and replication instructions from the Namenode.

HDFS is designed to deal with huge files by breaking them into blocks and replicating them across the network. The common practice is to replicate every file thrice and place two copies on different nodes on the same rack and the third copy on a separate rack. Since HDFS is based on the Google File System and so it follows the write–once–read–many access model that does not support continuous updates of data.

Hadoop’s infrastructure is built on the assumption that data will be distributed on hardware that is subject to failure. Therefore, when tasks are in progress on multiple nodes
concurrently then it can automatically detect a failure and restart the failed tasks on other healthy nodes. When a block or a node is lost, it automatically creates a copy of the missing data from available replicas. The only single point failure is the presence of a single Namenode, it does not provide real redundancy there.

Hadoop gives higher priority to high throughput than low latency and is suitable for batch processing. Hadoop is designed on the premise that moving computation is more efficient than moving data itself. Therefore, HDFS provides interfaces to move the application closer to the data – either on the same node or on the same rack thereby improving I/O bandwidth.

Hadoop is not the same as relational databases as Hadoop facilitates applications such as text analysis, image processing, that require analysis of all the available data. Hadoop allows users to run executables as mappers and reducers. This is based on fault tolerant, share-nothing architecture where tasks are independent of each other. The only exception in dependence is that the reducer awaits the results of the mapper under Hadoop control.

There are many open source projects built on top of Hadoop. Hive[5] is a data warehouse framework that finds its application in adhoc querying and complex analysis. Pig[7] is a data flow and execution framework produces a sequence of MapReduce programs. Mahout is used to build scalable machine learning libraries that focus on clustering, classification, mining frequent item-sets and evolutionary programming. Pydoop is a python package that provides and API for Hadoop MapReduce and HDFS.

VI. ALGORITHMIC DESIGN OF THE K-NEAREST NEIGHBOR TECHNIQUE IN THE MAPREDUCE PARADIGM

Prior to implementing the K-Nearest Neighbor technique lets see how to handle the input and output for the implementation. Since we are using the MapReduce paradigm we must make sure that the input is form of a <key, value> pair.

The Map routine performs the function of calculating the distance of each data point with the classes and lists it out. The Reduce routine then chooses the first ‘k’ Neighbors in increasing order of distances and conducts a majority vote. After which it sets the data point’s label as the label of the class with the majority vote count.

Now, we need to organize the input and output directories. To do this let us name the directory that holds the data vectors as vectors and the training and testing data as trainFile and testFile.

Having organized our input/output directories and training and testing data ready, we can apply the k-Nearest Neighbor technique in a distributed environment by following the algorithms discussed below to design the Map and the Reduce functions for the k-Nearest Neighbor Technique.

A. Algorithm for the Map Function

<table>
<thead>
<tr>
<th>Algorithm 1 Mapper design for k-NN</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: procedure K-NN MAPDESIGN</td>
</tr>
<tr>
<td>0: Create list to maintain data points in the testing data-set</td>
</tr>
<tr>
<td>0: testList = new testList</td>
</tr>
<tr>
<td>0: Load file containing testing data-set</td>
</tr>
<tr>
<td>0: load testFile</td>
</tr>
<tr>
<td>0: Update list with data points from file</td>
</tr>
<tr>
<td>0: testList &lt;= testFile</td>
</tr>
<tr>
<td>0: Open file containing training data set</td>
</tr>
<tr>
<td>0: OPEN trainFile</td>
</tr>
<tr>
<td>0: Load training data points one at a time and compute distance with every testing data point</td>
</tr>
<tr>
<td>0: distance (trainData, testData)</td>
</tr>
<tr>
<td>0: Write the distance of test data points from all the training data points with their respective class labels in ascending order of distances</td>
</tr>
<tr>
<td>0: testData &lt;= testData(dist, label)</td>
</tr>
<tr>
<td>0: Call Reducer</td>
</tr>
<tr>
<td>0: end procedure=0</td>
</tr>
</tbody>
</table>

B. Algorithm for the Reduce function

<table>
<thead>
<tr>
<th>Algorithm 2 Reducer design for k-NN</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: procedure K-NN REDUCEDESIGN</td>
</tr>
<tr>
<td>0: Load the value of 'k'</td>
</tr>
<tr>
<td>0: Load testFile</td>
</tr>
<tr>
<td>0: OPEN testFile</td>
</tr>
<tr>
<td>0: Load test data points one at a time</td>
</tr>
<tr>
<td>0: READ testDataPoint</td>
</tr>
<tr>
<td>0: Initialize counters for all class labels</td>
</tr>
<tr>
<td>0: SET counters to ZERO</td>
</tr>
<tr>
<td>0: Look through top k distance for the respective test data point and increment the corresponding class label counter</td>
</tr>
<tr>
<td>0: for i = 0 to k</td>
</tr>
<tr>
<td>0: COUNTERk +=</td>
</tr>
<tr>
<td>0: Assign the class label with the highest count for the testDataPoint in question</td>
</tr>
<tr>
<td>0: testDataPoint = classLabel(COUNTERmax)</td>
</tr>
<tr>
<td>0: Update output file with classified test data point</td>
</tr>
<tr>
<td>0: outFile = outFile + testDataPoint</td>
</tr>
<tr>
<td>0: end procedure=0</td>
</tr>
</tbody>
</table>
C. Algorithm to Implement the Map and Reduce Functions

Algorithm 3 Implementing kNN Function

0: procedure KNN FUNCTION
0: Read the value of ’k’
0: SET ‘k’
0: Set paths for training and testing data directories
0: SET trainFile
0: SET testFile
0: Create new JOB
0: SET MAPPER to map class defined
0: SET REDUCER to reduce class define
0: Set paths for output directory
0: SUBMIT JOB
0: end procedure

VII. THE EXPERIMENTAL RESULTS AND ANALYSIS

A. Experimental Setup

The k-Nearest Neighbor technique was implemented on a setup consisting four nodes connected over a private LAN. One node was used as a Namenode and Job Tracker, the other three other nodes were used as Datanodes and Task Trackers. All the four nodes had Intel i3 processors with 2.40Ghz and 2 GB memory. The Operating System running on all the 4 nodes was Ubuntu version 12.04. The programming language used to code the kNN Algorithm in both sequential and MapReduce Implementation was JAVA. Apache Hadoop version 1.0.4 was installed on all the nodes and the single node and consequent multinode configurations were done according to the guide found at [8][9].

B. Implementation Of Sequential and MapReduce kNN

The training data points were stored in a file called trainFile and the testing data points were stored in testFile. Once the data points were ready, the required statistical parameters, such as, iteration counters and timers were initialised. Next, the required files were copied into the Hadoop Distributed File System (HDFS). The Hadoop MapReduce iteratively until all the data points in the testing dataset were classified. We started with 500 data points of about 5KB in size and gradually increased the number of points up to 13 million about 125MB in size. Repeating the experiment five times helped us to reliably monitor the time taken to classify, ensure we could replicate the result with stability and identify any unexplainable circumstances.

The Algorithm worked well with the data-set chosen for experimentation. While the main purpose was to identify the suitability of using MapReduce for kNN, there was also a need to study its effectiveness. Therefore, a simple sequential kNN classification program to classify the same set of data points without MapReduce was also implemented. In this method, we stored the data points both training and testing in local files. The sequential kNN was then applied to assign the testing data points to the closest class label. We noted the time taken to converge to the final classification, number of iterations and the final classes.

C. Experimental Results and Observations

![Figure 2: The Experimental Setup](image)

![Figure 3: Graph of Sequential kNN vs MapRed kNN for a small sized dataset.](image)
The first observation made from the experimental results was that the simple sequential kNN classification took much lesser time compared to the MapReduce kNN classification for smaller datasets. This fact is illustrated by the graph in Figure 3.

The graph in Figure 4 shows a comparative plot of the time taken for up to 12 million points using MapReduce and sequential kNN.

![Figure 4: Graph of Sequential kNN vs MapRed kNN for a large sized dataset.](image)

It shows that Hadoop MapReduce performed better than sequential kNN from 8 million data points which is about 60MB in size. By 12 million points and 90MB size, it was found that Hadoop completed kNN classification 150 seconds ahead of sequential kNN.

VIII. CONCLUSION AND ONGOING RESEARCH

In today’s fast paced world, the rate of information exchange is tremendously high. We look for quick and robust methods to enhance and improve the way in which information is exchanged keeping in mind overall user experience. The most difficult task is to strike a balance between efficient performance and a high degree of user-friendliness. To achieve these it is important to understand the data collected so as to device an efficient way to handle this such data. This is where efficient data mining techniques are required. Data Mining can be thought of as a tool to extract diamonds of knowledge from the past and predict the future outcomes.

The k-Nearest Neighbor technique is ranked in the top ten data mining methods. In this paper we implemented the k-Nearest Neighbor technique in Hadoop MapReduce environment. This was done to remove the limitation of computational capability by having a cluster of systems working together. This not only eliminates the limitation of computational demands but also speeds up the processing time by having more than one computer interconnected over a network working on the given task. Also, in the experimental phase we compared MapReduce kNN with sequential kNN and found that MapReduce kNN out performs the sequential kNN with datasets of Larger Size. Hadoop being platform independent gives the user the flexibility to have heterogeneous system to be interconnected. Hence the user can use already existing hardware to setup a Hadoop cluster.

My current research work involves implementing more efficient data mining algorithms on the distributed environment. Also, comparing the performance of the algorithms in all kinds of cluster setup. This includes both homogenous and heterogeneous clusters. Along with the study of performance, the study of behavior of various algorithms with variations in size and computational methods is also an interesting area of research.

ACKNOWLEDGMENT

We would like to thank the continuous guidance and encouragement of all the Professors and support staff at RV College Of Engineering and they’re ever ready to lend a helping hand.

REFERENCES

[8] Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.