Problem statement:

KNN a supervised learning method is used to classify test data in HADOOP as well as Spark framework.

Bisected K-Means an unsupervised learning method is used to classify given data in HADOOP as well as SPARK.

Time required for computation of algorithms in both the frameworks is compared.

Spark

Introduction:

Apache Spark has its foundations based upon Resilient Distributed Dataset (RDDs for short). RDDs are read only and are distributed over cluster of machines in a fault tolerant way. Read only so that any transformation applied on RDD creates a new RDD and there is no way to modify existing RDD. Fault tolerence is achieved by maintaining multiple copies of existing RDDs over cluster of machines.

In the following report one master and one worker configuration is followed. PySpark software framework is used imported in python environment.

K-NN Algorithm:

Application name is specified as "knnSpark"

Libraries math, operator and random is used.

On localhost master node is created.

Train data is imported in RDD named as RDDtrain

Test data is imported in RDD named as RDDtest

Transformation-1:

MAP transformation is performed on RDDtrain and RDDtest which stores value in RDDtrain2 and RDDtest2. In MAP transform each row of RDDtest/RDDtrain existing as string with space separated datapoints is converted into a list by using split function on string and the list is mapped to RDDtrain2/RDDtest2.

Action-1:

Length of test data is obtained using count() action which counts number of rows in test data. The count function returns string which is converted to int data type by type conversion and output is stored in testlength.

Action-2:

Testlist of datatype list is created by reading RDDtest2 using take() action and passing testlength as argument so that whole RDD is read.

KNN algorithm is performed on testlist by perfoming n(=testlength) iterations where in i^{th} iteration class is assigned to i^{th} test data point.

Transformation-2:

MAP transformation is performed on RDDtrain2 which store value in tempRDD. In the transform euclidian distance of given test point is calculated from train points and the distance and class id of train point is mapped to tempRDD.

Action-3:

Distances and class id of k nearest neighbors are passed to classifier function as arguments. The k nearest neighbors are taken using takeOrdered action which returns k lowest RDD values sorted according to the key passed. Here the key is distance.

The classifier function returns the class which occurs the most amongst the data points passed and hence the class of test data point is assigned according to majority voting.

Percent of test data points whose class is correctly predicted represents the accuracy of algorithm.
Accuracy=99.67%

Time is measured taking the difference between times of start and end of algorithm. Times of start and end of algorithm are measured using timer() function. From timeit library default_timer function which returns system time as output is imported as time() function.
Time taken=5hr 11min

The flowchart for algorithm is given in fig.1.

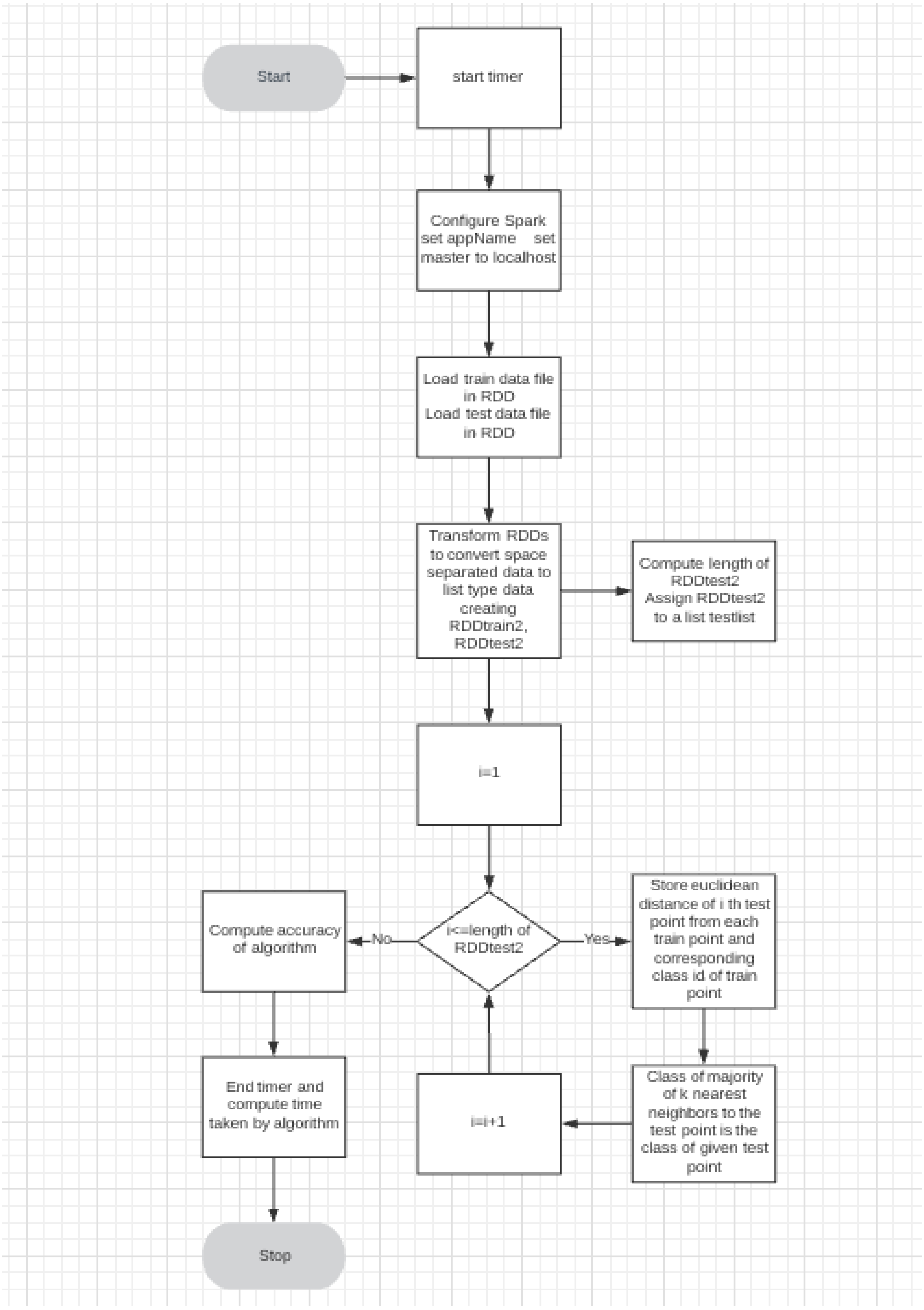

Start
start timer
Configure Spark set appName set master to localhost
Load train data file in RDD Load test data file in RDD
Transform RDDs to convert space separated data to list type data creating RDDtrain2, RDDtest2
Compute length of RDDtest2 Assign RDDtest2 to a list testlist
i=1
i<=length of RDDtest2
Yes
Store euclidean distance of i th test point from each train point and corresponding class id of train point
Class of majority of k nearest neighbors to the test point is the class of given test point
i=i+1
No
Compute accuracy of algorithm
End timer and compute time taken by algorithm
Stop

Fig.1

Bisected K-means Clustering:

Design of The Algorithm

The Pseudo algorithm of Bisected K means is implemented using the following steps:

1. Firstly, one column is attached to the dataset by applying transformation in the RDD by mapping the function addClustrArg() to the RDDfiltered.
2. Using 2 means, the entire dataset is divided into 2 Clusters. Out of these set of clusters, a cluster is selected which has highest SSE and subsequently, 2 means is applied on it. This process is applied recursively till the number of clusters equals to the required K.
3. While performing 2 means, we start with two randomly selected points from the cluster and perform the following till the change in the centroid is less than a specified epsilon:
   a) Assign points to either of the two clusters depending on the least distance with the centroids.
   b) Assign two new centroids by taking mean of the two clusters.

RDD Transformations and Actions

Firstly the entire dataset is taken into RDDList using textFile() function.

Transformation 1**:** The tab separated data in RDDList is transformed into list of records which is then mapped to RDDList2.

Transformation 2**:** As in the first two rows of dataset there are scalar values only records which have 13 fields have been taken using filter transformation.

Transformation 3: In this transformation we transform the RDDfiltered by adding an extra column which stores the cluster to which the point belongs and it is mapped to RDDclustered.

Action 1 – This action is invoked by takesample() function in initialise2Centroids function which returns two random points.

Transformation 4 – This is a map transformation that assigns the 2 initial centroids to the train dataset which is now in RDDclustered2. This transformation transforms RDDclustered to RDDclustered2.

Transformation 5 – Using this map transformation we assign the data points to corresponding centroids and create the resulting clusters.

Action 2 – This action is called in the newCentroids() function which is used to extract all the records in tempRDD. This is done using collect() transformation function.

Transformation 6 – This map transformation is recursively called in order to assign the points to the new clusters depending on the centroid point which is to their nearest. This is repeated till total k clusters are not formed.

System Function Used to Measure Execution Time

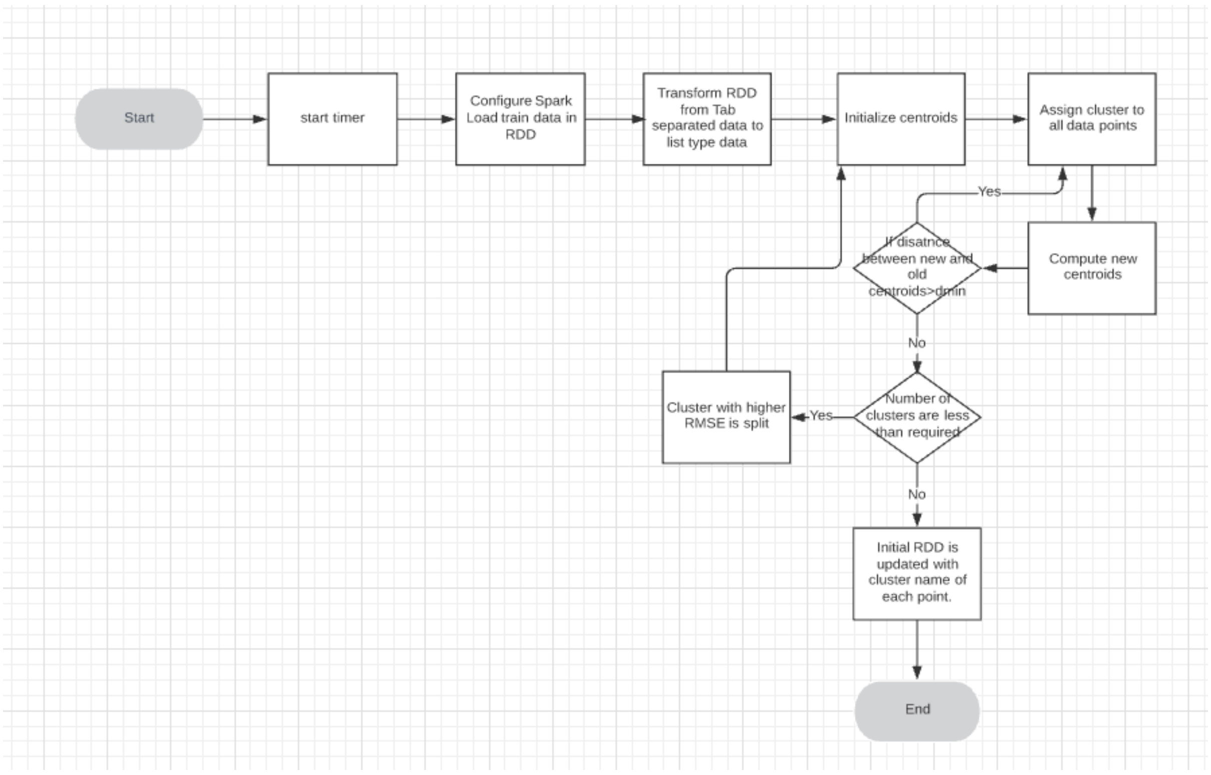
We measure the execution time of the Code using timer() function. The timer() function returns the current time. Just after starting the code the start integer stores the initial timer() value and at the end of code the end integer stores the ending timer() value. The total time taken by the code is equal to end – start.

Time is measured taking the difference between times of start and end of algorithm. Times of start and end of algorithm are measured using timer() function. From timeit library default_timer function which returns system time as output is imported as time() function.

Time taken=8hrs

Flowchart for algorithm is given in fig.2.

The flowchart for algorithm is given in fig.2.



Start
start timer
Configure Spark Load train data in RDD
Transform RDD from Tab separated data to list type data
Initialize centroids
Assign cluster to all data points
Compute new centroids
If disatnce between new and old centroids>dmin
Yes
No
Number of clusters are less than required
Yes
Cluster with higher RMSE is split
No
Initial RDD is updated with cluster name of each point.
End

Fig.2

HADOOP

Introduction

Hadoop MapReduce is a software framework to do distributed data processing. MapReduce divides the input data into several parts and each part is processed at different data nodes. Map and Reduce are the two major tasks of MapReduce. Map basically converts the input data in tuples. The output from Map is given as input to the Reduce task. Reduce combines those data tuples into a smaller set of tuples.

K-nearest neighbor:

- The value of K is taken as input from the file.
- Using K-nearest neighbor, every MapReduce task computes the class of one test record.
- The program can be modified to print the output of each record in different files.

Map:

- In the setup phase of mapper we retrieve the test record.
- The distance between the test and train record is calculated in the map phase and it is stored in a TreeMap. K closest records are kept in the TreeMap at any given moment.
- In the cleanup phase, we emit the following to the reduce:
  <NullWritable, <Distance,Class> Pair >
  A special class is made for the above pair to be Writable.

Reduce:

- Reduce receives the key of type NullWritable.
- The reduce iterates over the values received from the mapper.
- If we had n data nodes, the number of values received by the reduce would be n*k. The reduce filters the closest k records using a TreeMap.
- The most frequent class is assigned to the test record.
- The reduce writes this into the output path as setup for the job.

System.currentTimeMillis(): This system function is used for capturing the time taken by the program. We use this function to measure the start time and the end time, and subtract the two.

Time taken=10hrs

The flowchart is shown in fig.3.

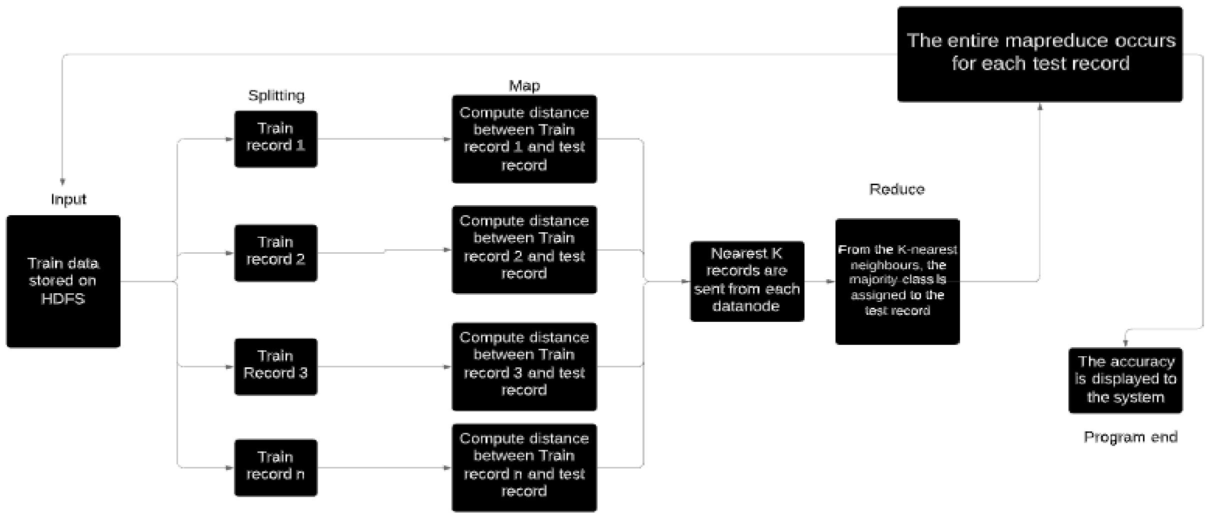

The entire mapreduce occurs for each test record
Input
Train data stored on HDFS
Splitting
Train record 1
Train record 2
Train Record 3
Train record n
Map
Compute distance between Train record 1 and test record
Compute distance between Train record 2 and test record
Compute distance between Train record 3 and test record
Compute distance between Train record n and test record
Nearest K records are sent from each datanode
Reduce
From the K-nearest neighbours, the majority class is assigned to the test record
The accuracy is displayed to the system
Program end

Fig.3

Bisected K-means

Please note the following points:-

1. We have removed the number of attributes and the number of records, which are present in the first two line.
2. In twelve iterations of MapReduce we bisect a given cluster into two using two means.
3. For two means, two random points (centroids) are generated in the beginning using a randomizer function.
4. The first iteration of two-means breaks the entire dataset into two clusters.
5. Each subsequent iteration bisects the cluster with greatest SSE into 2.

Map:

1. In the map phase, every record to be clustered is retrieved and the closest centroid is found. It is stored globally in a HashMap as well for fast retrieval and further usage.
2. In cleanup phase, <centroid, record> pair is sent to the Reduce.

Reduce:

1. The key that we get from the reduce contains the centroid.
2. The values contain all the records in consideration that are from the “key” cluster.
3. The calculation of the new centroids and the calculation of the Sum of Squared Errors happens in this phase.
4. The reduce writes this into the output path as setup for the job.

System.currentTimeMillis(): This system function is used for capturing the time taken by the program. We use this function to measure the start time and the end time, and subtract the two.

Time taken=27hrs

The flowchart is shown in fig.4.

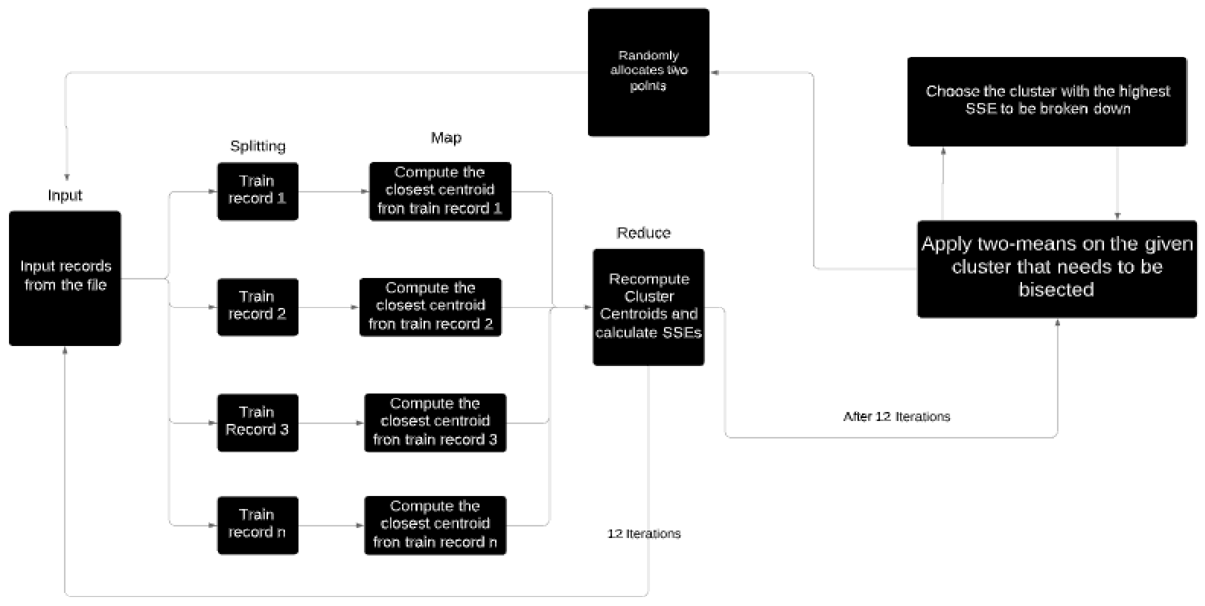

Randomly allocates two points
Choose the cluster with the highest SSE to be broken down
Splitting
Map
Input
Input records from the file
Train record 1
Compute the closest centroid fron train record 1
Train record 2
Compute the closest centroid fron train record 2
Train Record 3
Compute the closest centroid fron train record 3
Train record n
Compute the closest centroid fron train record n
Reduce
Recompute Cluster Centroids and calculate SSEs
Apply two-means on the given cluster that needs to be bisected
After 12 Iterations
12 Iterations

Fig.4

Spark vs. HADOOP

It has been observed that the time taken to compute the algorithms in Spark framework is much lesser than that required in Hadoop framework. This is due to the fact that HADOOP takes a subsequent time in performing I/O operations as map reduce is performed for each transformation. Spark does in memory computations hence has comparatively lesser I/O operations. Thus, computation time is reduced significantly.