ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal
Regular Issue, Vol. 8 N. 3 (2019), 67-77
eISSN: 2255-2863
DOI: http://dx.doi.org/10.14201/ADCAIJ2019836777

ck-NN: A Clustered k-Nearest Neighbours Approach for Large-Scale Classification

Rafi Ullah[a], Ayaz H. Khan[b], and S.M. Emaduddin[a]

[a] College of Computing and Information Sciences, Karachi Institute of Economics and Technology, Pakistan
[b] Computer Science Department, Dhanani School of Science and Engineering, Habib University, Pakistan
rafi@pafkiet.edu.pk, ayaz.hassan@sse.habib.edu.pk, emad@pafkiet.edu.pk

KEYWORD

k-Nearest Neighbor Classification; k-Means Clustering; Machine Learning; Artificial Intelligence; Distributed Computing

ABSTRACT

k-Nearest Neighbor (k-NN) is a non-parametric algorithm widely used for the estimation and classification of data points especially when the dataset is distributed in several classes. It is considered to be a lazy machine learning algorithm as most of the computations are done during the testing phase instead of performing this task during the training of data. Hence it is practically inefficient, infeasible and inapplicable while processing huge datasets i.e. Big Data. On the other hand, clustering techniques (unsupervised learning) greatly affect results if you do normalization or standardization techniques, difficult to determine "k" Value. In this paper, some novel techniques are proposed to be used as pre-state mechanism of state-of-the-art k-NN Classification Algorithm. Our proposed mechanism uses unsupervised clustering algorithm on large dataset before applying k-NN algorithm on different clusters that might running on single machine, multiple machines or different nodes of a cluster in distributed environment. Initially dataset, possibly having multi dimensions, is pass through clustering technique (K-Means) at master node or controller to find the number of clusters equal to the number of nodes in distributed systems or number of cores in system, and then each cluster will be assigned to exactly one node or one core and then applies k-NN locally, each core or node in clusters sends their best result and the selector choose best and nearest possible class from all options. We will be using one of the gold standard distributed framework. We believe that our proposed mechanism could be applied on big data. We also believe that the architecture can also be implemented on multi GPUs or FPGA to take flavor of k-NN on large or huge datasets where traditional k-NN is very slow.

1. Introduction

k-Nearest Neighbor (k-NN) Classification Algorithm (known as Supervised learning technique), widely used algorithm in machine learning having applications in many areas from recommendation systems to image processing, but it is usually known as lazy learning that most of the time it takes during query matching (testing), and it act as a brute-force technique as each record will be matched with query for finding similarity which makes it difficult to apply in huge data that is big data. Although we have GPUs and multi-core systems to do this particular task, still we have memory constraints. We have proposed a technique using many systems in a cluster (distributed system) with each node contains many cores inside. We are taking the advantages of both classification and clustering technique here, we are using k-Means and k-NN together to process big data.

We applied k-means clustering technique on dataset "S" such that the dataset is divided into chunks equal to the number of nodes in a cluster. Number of nodes in a cluster is same as the number of processing machines connected in our distributed environment. Each chunk of data is then transferred to nodes in cluster. Data Transfer can be carried out manually or using some technique to transfer chunks to nodes over the network. Data distribution is the task of Controller or coordinator node. Each of the nodes in cluster executes k-NN classification on their respective chunk of dataset which return their "k" best results after matching query with chunk of data. The selector node selects the best results amongst all received results from each node. For selection of best result, we can also use some weighted scheme. This technique can be used in various areas like real time transactions data, twitter data, any real time streaming, stock exchange data, and even in image processing. The proposed technique is theoretically proven efficient and more accurate in case of large datasets as compared to straight forward k-NN classification algorithm. This technique can also be modified for implementing multi-GPUs or FPGA based systems.

The rest of the paper discusses K-Means clustering, k-NN classification, proposed methodology and result. In section 2 of this paper, current techniques are discussed for using K-NN algorithm as related works. Section 3 and section 4 is more about the description of k-Means clustering technique and k-NN classification algorithm respectively. Section 5 discusses our proposed methodology, while results and conclusion/future work are disi-cussed in section 6 and 7 respectively.

2. Related Works

For finding accurate estimations of measures for high dimensional data in (Callahan and Kosaraju, 1995) (decomposition technique has been proposed) becomes inefficient and difficult, so in (Garcia *et al*., 2008) pre-structuring of data is performed and binary trees are proposed for applying k-Nearest Neighbor Algorithm executed over GPUs for big data.

When there is large amount of data or may be high dimensional data, k-NN in such case fails due to most of time it takes during testing query, to avoid such issue, technique is proposed in (Deng *et al*., 2016), dataset is separated into chunks by performing k-Means clustering algorithm and then the nearest cluster is selected as training sample for k-NN Algorithm. Now complexity of proposed technique is only dependent of cluster size. Authors of (Seidl and Kriegel, 1998) proposed exact and approximate algorithms in MapReduce to perform efficient parallel k-Nearest Neighbor joins in large data such as spatial databases and multimedia databases or simply performing k-NN joins. Parallel and highly optimized kdtree based k-Nearest Neighbor classification algorithms construction and querying mechanism for distributed architecture has been proposed in (Patwary *et al*., 2016). It uses concepts of pruning search space, improvisation of load balancing and partitioning of nodes and threads. Proposed algorithm out-performs in big data analytics platform.

K-NN joins, for finding nearest neighbors from dataset "S", the primitive operation widely used in data mining applications. This becomes costly operation in case of big data; the investigation has been done in (Lu *et al*., 2012), to study how k-NN joins perform using MapReduce. Mappers perform clustering on objects and then reducers perform k-NN joins on each of the clusters of objects. Two approximate algorithms to minimize the number of replicas have been introduced. Parallel algorithm for k-NN graph creation has been proposed in (Connor and Kumar, 2010). Proposed algorithm has been shown efficient over other techniques in term of space usage, better cache efficiency, ease of parallelization and the ability to handle large datasets.

In (Liang *et al*., 2010) (Liang *et al*., 2009) (Wang *et al*., 2018) (Pu *et al*., 2015) (Barrientos *et al*., 2010) show the implementation of k-NN algorithm (lazy learning, most of the time is taken during testing query) on CUDA (Compute Unified Device Architecture), GPU (Graphical Processing Units), FPGA (Field Programmable Gate Array) and Multi-GPUs. And the performance of k-NN on different datasets under different sceneries has been discussed.

Multi-GPU implementation of k-Nearest Neighbor classification algorithm has been proposed in (Masek *et al*., 2015) keeping big data in mind [size of big data and sources have been discussed in (Rehman *et al*., 2015). This technique has been shown efficient and effective for large data. It has been shown in paper that using Open-CL on GPU will increase 750 times than CPU. Authors in (Li *et al*., 2018) proposed an automatic fast double k-Nearest Neighbor Classification algorithm uses the concept of hierarchical clustering.

Similar matching like k-NN technique has been shown used for recommendation system, for closest match in (Dong *et al*., 2011), (Chae *et al*., 2018) and (Sohani *et al*., 2018). Authors in (Sohani *et al*., 2018) have shown record clustering of huge data into clusters or groups and then matching closest records to query. Matching was not straight forward like k-NN but it was somewhat like custom k-NN algorithm.

3. k-Means Clustering

K-means is one of the simplest and most widely used unsupervised learning algorithms that solve well known problems. It classifies data through certain number of clusters that are fixed priori. The main idea behind k-Means is to define “k” centroids, one for each cluster. These centroids are randomly chosen at start and should be place at different locations for better results. Then take each point from dataset and calculate from which cluster it belongs to and classify to that cluster. This process is iterative process until proper clusters are created. The objective function of k-Means is the squared error function; k-Means tries to minimize function value as given eq. 1.

$$j = \sum_{j=1}^{k} \sum_{i-1}^{k} \left\| {x_i}^{(j)} - C_j \right\| \tag{1}$$

Suppose that we have “n” data points or sample feature vectors [x_1, x_2, x_3,……x_n] all from the same class, and we know that they fall into “k” clusters, $k < n$. Let m_i be the mean of the vector in cluster i. Algorithm 1 shows the pseudo code of the standard K-Means clustering algorithm.

Algorithm 1 K-Means Clustering

```
K-means Clustering (Data, Clusters)

Parameters:
Data: input data points
Clusters: No. of clusters

Algorithm:
1: m = [$m_1$, $m_2$, $m_3$,……, $m_k$ ]
2: x = initial_guess($m_1$, $m_2$, $m_3$, …, $m_k$)
3: x_new = 0
4: repeat
5:    x = x_new
6:    for i = 1 to k do
7:          Replace (mi, x_new)
8:    end for
9: until x == x_new
```

In Algorithm 1, line 1 shows the complete data or all the data examples. In line 2 we have initial guesses that are about choosing the number of clusters by setting centroids. Initially centroid can be randomly calculated. Each of the created centroid defines one of the clusters. In this step, each data point is assigned to its nearest centroid, based on the squared Euclidean distance or may use other distance measure methods. Once all data points are classified in their respective clusters, line 4- 9 is the centroid update step, here centroids are recomeputed. Centroid update is to simply replace old centroid by the mean of all data points in that centroid clusters. Here iteration of the clustering algorithm is shown until some stopping a criterion as given in line 9 occurs. Stopping criteria shows that stage where no data points change clusters, the sum of distance from all data points to centroid is minimized or may be the limit of iteration reached. More iteration may give better results and less iteration may give local optimum. But running algorithm more than once with different random centroid may give better results.

4. k-NN Classification

In classification algorithms, the k-Nearest Neighbor also know as lazy learning technique essentially boils down to forming a majority vote between the "k" most similar examples to given unseen examples. Similarity is defined according to distance metrics between two examples. Distance can be found using popular Euclidean Distance formula as given in eq. 2,3,

$$d = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2 + ... + (Z_2 - Z_1)^2} \tag{2}$$

or Manhattan distance formula

$$d = |(X_2 - X_1)| + |(Y_2 - Y_1)| + ... + |(Z_2 - Z_1)| \tag{3}$$

and others likes jaccard similarity, cosine similarity, Chebyshev and Hamming distance. Each has pros and cons and can be used in specific problem, depend upon the nature of problem.

Formally given integer "k" and unseen examples "x", and the distance "d", K-Nearest Classifier performs two steps.

Algorithm 2 K-Nearest Neighbor Algorithm

```
K-Nearest Neighbor Algorithm(TrainnigData[], Initial_Query, k)

Parameters:
TrainningData[]: input data points Initial_Query: Test data
k: No. of samples for deciding class label

Algorithm:
1: observation[p1,p2] = Read(TrainningData)
2: x = initial_query // testing data that need to be clustered
3: for i = 1 to n in observation do
4: distancei = ComputeDistance(x, observationi)
5: end for
6: compute result[] // where result is an array containing frequency of smallest distance
7: return k label with highest frequency
```

In Algorithm 2, line 1 explains the working of reading data from source i.e. file. Here, 'p1' is data point and 'p2' is class label. Observation contains all the data attributes and class label. Line 3 and 4 discussed the main working of k-NN algorithm, computing differences or distances among the query and all observations. Line 6 explains the selection of best results (best result is the observation having minimum distance with query). Line 7 is returning the class label of observation to query, hence query is classified as label same as observation class label.

5. Proposed Methodology

Fig 1. shown the big picture of the system. There is a cluster of connected nodes, nodes could be GPUs or CPUs, but this paper focused on CPUs. Main components of the system are:

1. Coordinator Node or Controller Node
2. Intermediate Nodes
3. Selector Node

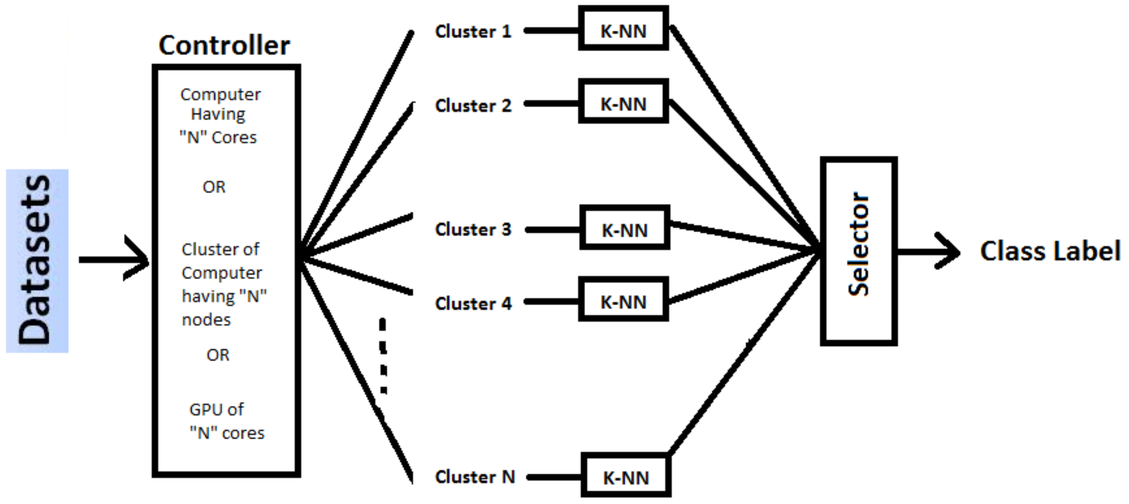

Figure 1: Big Picture of the System Architecture.

Algorithm 3 shows the pseudo code of the coordinator node algorithm. Line 1 of the algorithm given or executed on coordinator node, data from file or some different source is read. Line 2 explains the pre-processing of data. Pre-processing involve all basic data pre-processing steps like missing values, label encoding, scaling, standardization and normalization of data. This step may differ mainly dependent on the nature of data such as numeric, textual data or images data. In Line 3 k-Means Algorithm is applied on data by specifying number of clusters (here number of clusters equals to the number of nodes in clustered environment), and then each cluster is saved separately in the form of file on each node as given in Line 4. As data saved on each node belongs to different cluster, hence we can infer that any noise in data can be only in one or some clusters, most of the clusters will be noise-free, which makes it more efficient to be applied in case of noisy data. Line 5 shows working of data distribution on each cluster. This can be done manually by transferring all the files to nodes through USB or something else. But we have a direct transferring mechanism in our framework, the coordinator will distribute data files such that the related data file is accessible by each node in the distributed system.

Algorithm 3 Coordinator Node Algorithm

```
Coordinator Node Algorithm(Data)

Parameters:
Data: input data points

Algorithm:
1: data = CoordinatorReadData()
2: cleanedData = CoordinatorPreProcess(data)
3: clusters [] = kMean(cleanedData, nodes)
4: files [] = saveFile(clusters)
5: DistributeFiles(files, nodeID)
```

Algorithm 4 Intermediate Node Algorithm

```
Intermediate Node Algorithm()

Algorithm:
1: data = readFile(location,nodeID)
2: results [] = applykNN(data, cores)
3: bestRecord = selectBest(results)
4: return bestRecord
```

Algorithm 4 shows the pseudo code of the intermediate node algorithm. In line 1, readFile method simply reads their own data using the data path and the node id (as we have different data on different nodes). As the coordinator node apply k-Means clustering and distribute data on different nodes in cluster by giving nodeID as a name. Line 2 describe successful reading of data, each of the node apply k-NN algorithm on their data and calculate local best results. Local best results are the best classification label at each cluster data. Line 3 explains selection function (simple voting method, weighted voting method etc.). In line 4 the local best result (classification label) is returned to selector node.

Algorithm 5 Selector Node Algorithm

```
Selector Node Algorithm(NodeID[])

Parameters:
NodeID[]: list of all node IDs

Algorithm:
1: results [] = retriveResult(nodeID)
2: results [] = selectionFuction(results[])
3: Return bestResult
```

In our framework the selector node is the node where no supervised or unsupervised algorithm executed, this node only serves for retrieving best results from each of the nodes in clusters by using nodeID. Algorithm 5 shows the pseudo code of the selector node algorithm. Line 2 shows the working of selection function. Selection function is used to find the final classification label of query, as we may have different data on different nodes, the local result may differ by accuracy or confidence value, selector node finds best among those results. Selection methods are briefly described in the paper section 5.

Global best result is selected on a function, we called this selection function. Three different types of selection functions are introduced in proposed technique:

1. Voting selection function: Select class variable on the basis of votes by different clusters of data. Let suppose the final results is [(C1, class1),(C2, class2),…..(CN, class N)], of which three of the clusters results says that query belongs to class 1, then final global result is selected on this basis. If all of the clusters results are unique, then select class having minimum distance.

2. Accuracy selection function: Take the Average of all Accuracies of different clusters and then select maximum.

3. Weighted clusters selection function: Weight is allocated to the node according to the amount of data they possess. Higher value of weight is given to the node with larger dataset. Cluster is created according to the value of k and weight of nodes.

6. Results

6.1. Evaluation Metrics

We have tested our proposed technique on a different datasets having different number of records and types of attributes. Basic data preprocessing is required if data in not numeric. Two of the parameters under consideration were time of execution (training time in machine learning terminology) and the Accuracy. Time of Execution was observed decreasing effectively and the Accuracy varies with number of clusters. Table 1 shows the properties of each dataset. Table 2 shows statistics of k-NN Algorithm when applied directly on data. Clustering technique K-means algorithm is then applied to make up to 15 clusters and for different clusters Execution time and Accuracies are recorded as shown in Table 3.

Table 1: Dataset Description

Data Set	Iris	Sonar	Pen Based	Ring	Soil Cover Type	Hearts
Origin	Real World	Real World	Real World	Laboratory	Real World	Real World
Instances	729	1040	103285	48839	581012	1350
Features	04	60	16	28	56	13
Classes	03	02	10	02	05	02
Missing Values	No	No	No	No	Yes	No
Noise	20%	20%	20%	20%	0%	20%

Table 2: kNN Algorithm statistics on different datasets

Data Set	Iris	Sonar	Pen Based	Ring	Soil Cover Type	Hearts
Execution Time	0.1206	0.1326	0.8486	22.16	12.39	0.1232
Accuracy	0,890	0.9192	0.9946	0.9628	0.9672	0.8609

All the datasets described in this paper contains 20% noisy data. Fig 2(A) is the visualization of Iris dataset. Features are only 4, and there are 3 different classes with no missing data. Here it has been observed that as the number of clusters increases the time to compare decreases while the accuracy increases with increasing number of clusters except for the clusters equals to 8. Highest Accuracies were observed when clusters are 6,7,9 and 12. Ration of Instance to Features is almost 182:1 and Features to classes is 1.3:1 and instance to class is 243:1.

Fig 2(B) is the visualization of Sonar dataset having 60 features, and there are 2 different classes with no missing data. Here it has been observed that there is no specific direct or indirect relation between number of clusters and the accuracy. Highest Accuracies were observed when clusters are 2,7,8 and 12. Ration of Instance to Features is almost 18:1 and Features to classes is 30:1 and instance to class is 520:1.

Table 3: Detail Description of proposed technique (clusters, accuracy, execution time)

	Iris		Sonar		Pen Based	
Clusters	Accuracy	Time	Accuracy	Time	Accuracy	Time
2	0.9016	0.00713	0.8871	0.0341	0.9928	0.2022
3	0.902	0.0077	0.9389	0.0189	0.994	0.101
4	0.9051	0.0075	0.8636	0.0168	0.9924	0.0937
5	0.9096	0.0049	0.915	0.0159	0.994	0.0606
6	0.9428	0.0059	0.8534	0.0157	0.9945	0.0483
7	0.9437	0.0055	0.83047	0.0141	0.9914	0.0557
8	0.8703	0.0056	0.95	0.0126	0.9891	0.0401
9	0.9646	0.0054	0.9474	0.0143	0.9893	0.0286
10	0.9532	0.0057	0.9205	0.0125	0.9898	0.0422
11	0.918	0.0058	0.9059	0.01361	0.991	0.0273
12	0.954	0.0042	0.896	0.0131	0.9938	0.0374
13	0.9173	0.0057	0.9442	0.0127	0.9938	0.0299
14	0.9286	0.0061	N/A	N/A	0.9928	0.0312

	Ring		Soil Type		Hearts	
Clusters	Accuracy	Time	Accuracy	Time	Accuracy	Time
2	0.9722	0.2169	0.9698	3.6086	0.9398	0.0091
3	0.9666	0.201	0.9671	1.8675	0.8905	0.0076
4	0.9687	0.0829	0.9721	1.2134	0.8905	0.0087
5	0.9562	0.053	0.9645	0.937	0.8596	0.0081
6	0.9648	0.045	0.972	0.7357	0.8738	0.007
7	0.9632	0.0416	0.9678	0.638	0.8702	0.0072
8	0.9724	0.0532	0.9718	0.6431	0.8534	0.007
9	0.9683	0.0331	0.9712	0.4881	0.9082	0.0073
10	0.9676	0.0358	0.9708	0.4206	0.8917	0.0066
11	0.9676	0.0339	0.966	0.3896	0.9086	0.0058
12	0.966	0.0313	0.9695	0.3456	0.9097	0.0077
13	0.9684	0.0315	0.9686	0.3309	0.9315	0.007
14	0.9734	0.0293	0.9454	0.3046	0.8689	0.0069

Fig 2(C) is the visualization of Pen Based dataset having 16 features, and there are 10 different classes with no missing data. Here it has been observed that there is no specific significant effect when the numbers of clusters increases in term of accuracy but the execution time will definitely decreases. Ration of Instance to Features is almost 6456:1 and Features to classes is 1.6:1 and instance to class is 10329:1.

Fig 2(D) is the visualization of Ring dataset having 28 features, and there are 2 different classes with no missing data. Here it has been observed that Accuracy Increases when clusters increases from value 4 and decrease Accuracy when clusters increase from 1 to 4. Highest Accuracies were observed when clusters are 1,7,12 and 13. Ration of Instance to Features is almost 1744:1 and Features to classes is 14:1 and instance to class is 14420:1.

Fig 2(E) is the visualization of Soil Cover Type dataset having 56 features, and there are 5 different classes with no missing data. Here it has been observed that there is Accuracy Decreases while increasing number of clusters. The only advantage we get here is the decrease in execution time as number of clusters increases. Ration of Instance to Features is almost 10376:1 and Features to classes is 11:1 and instance to class is 116203:1.

Fig 2(F) is the visualization of Hearts dataset having 13 features, and there are 2 different classes with no missing data. Here it has been observed that there is no specific direct or indirect relation between number of clusters and the accuracy. Highest Accuracies were observed when clusters are 1,8 and 12. Ration of Instance to Features is almost 104:1 and Features to classes is 6.5:1 and instance to class is 675:1.

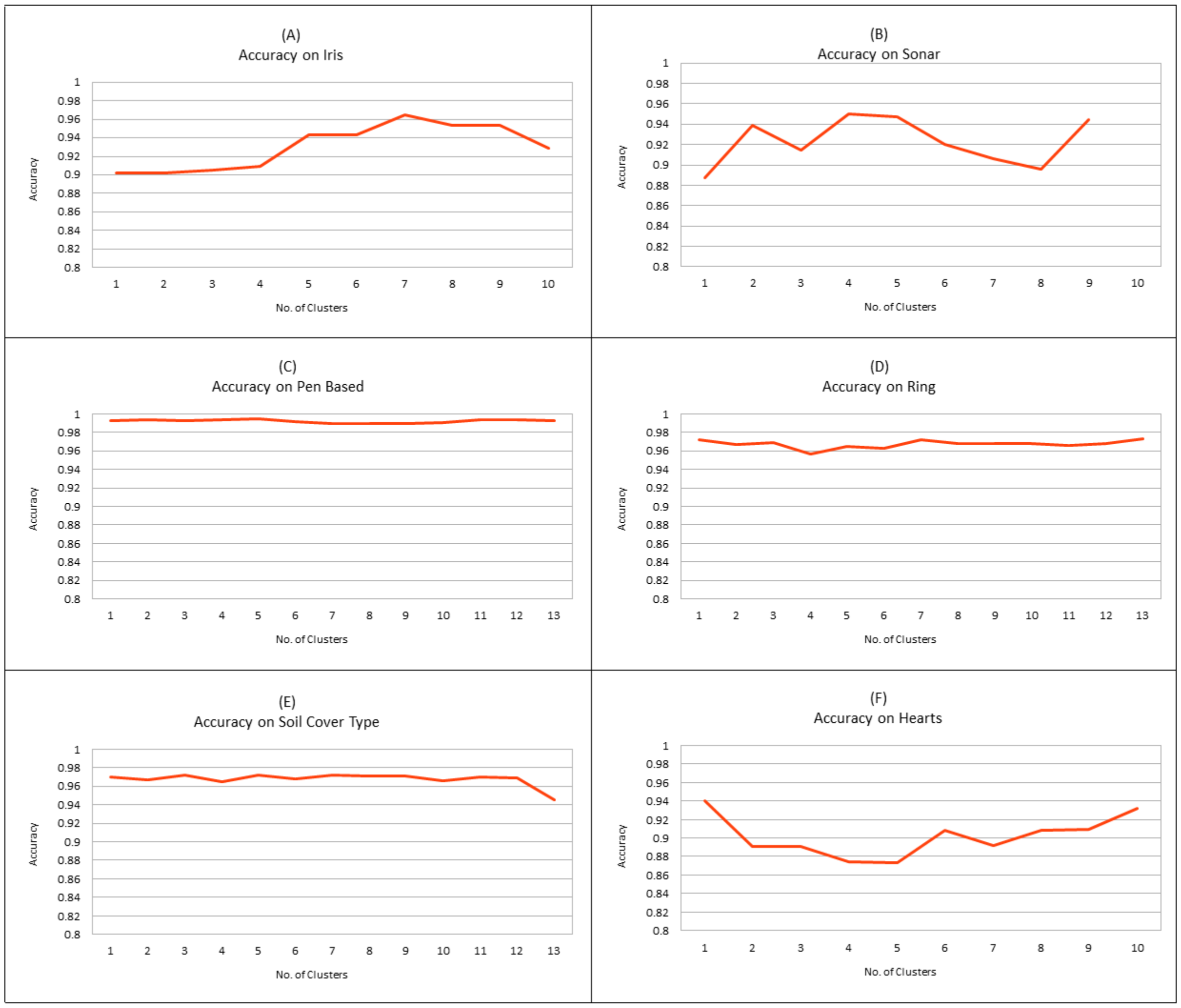

Figure 2: Comparison of Accuracy on Data Sets.

The general intuitions that can be concluded from above experiments are as follows:

1. Proposed technique can be very useful to process huge amount of data in terms of execution time, because it cannot degrade accuracy. So, we can significantly reduce execution time while maintaining same accuracy

2. In most of the cases it has been observed that clusters equal to 2 times of features or 6 times of features gives significant results.

3. For large value of instance to feature ration clustered k-NN has no significant results, but for small value of instance to features ration, proposed technique has significant results, so it can be concluded that proposed technique works best for high dimensional data where other techniques performed poorly. Hence

no need of dimensionality reduction. We can make better decisions through many dimensional data having lesser number of records.

4. Proposed technique out performed for small datasets. It solves the issue of large datasets requirements for training better model to avoid over-fitting and under-fitting.

5. In case of noisy data, it always out-performed than simply applying k-NN.

7. Conclusion and Future Work

The proposed technique works best and efficient in case of high dimensional big data and especially out performs when datasets are noisy in terms of accuracy and execution time. Time required for creating clusters has been considered in statistics shown in paper, because clustering is one-time process. This technique can be compared with straight forward k-Nearest Neighbor classification algorithm applied on large and noisy data. This proposed technique works best in case of outliers in data as discussed that all datasets contains 20% noisy data i.e. outliers. The accuracy of proposed technique overshoots in most of the cases in the presence of Noise in data. Clear distinction between traditional k-NN and clustered k-NN has been observed on mentioned datasets except pen based dataset, we have not got any advantage of accuracy as k-NN has greater than our proposed technique. But we have advantage in our proposed technique in term of execution time. We can also try parallel k-Nearest Neighbor for application of proposed technique in a large-scale data processing. This technique can be modified to execute over GPUs. Or we can also try other clustering technique instead of k-Means to observe accuracy and execution time. This technique has been shown effective and efficient for large and noisy datasets i.e. having datasets with high tendency of outliers. Once system has been deployed somewhere, arrival of new records does not need re-clustering or retraining, we just calculate distance between new record(s) to all clusters and then keep in the most similar cluster. In case, when cluster size increases, splitting of cluster may take place. In future, the proposed technique will be evaluated on a multi-node compute cluster (distributed system) to evaluate the scalability of the proposed methodology. Proposed system may be very successful in case of other datasets such as textual and images data. Potential work can be performed on relation between clusters and accuracy, as in some cases accuracy increases when a cluster increases and vice-versa.

8. References

Barrientos, R., Gómez, J., Tenllado, C., and Prieto, M., 2010. Heap based k-nearest neighbor search on GPUs. In *Congreso Espanol de Informática (CEDI)*, pp. 559-566.

Callahan, P. B. and Kosaraju, S. R., 1995. A Decomposition of Multidimensional Point Sets with Applications to K-nearest-neighbors and N-body Potential Fields. *J. ACM*, 42(1):67-90. ISSN 0004-5411. doi: 10.1145/200836.200853.

Chae, D.-K., Lee, S.-C., Lee, S.-Y., and Kim, S.-W., 2018. On identifying k-nearest neighbors in neighborhood models for efficient and effective collaborative filtering. *Neurocomputing*, 278:134-143.

Connor, M. and Kumar, P., 2010. Fast construction of k-nearest neighbor graphs for point clouds. *IEEE transactions on visualization and computer graphics*, 16(4):599-608.

Deng, Z., Zhu, X., Cheng, D., Zong, M., and Zhang, S., 2016. Efficient kNN classification algorithm for big data. *Neurocomputing*, 195:143-148.

Dong, W., Moses, C., and Li, K., 2011. Efficient k-nearest neighbor graph construction for generic similarity measures. In *Proceedings of the 20th international conference on World wide web*, pp. 577-586. ACM. Garcia, V., Debreuve, E., and Barlaud, M., 2008. Fast k Nearest Neighbor Search using GPU. *CoRR*, abs/0804.1448.

Li, H., Li, H., and Wei, K., 2018. Automatic fast double KNN classification algorithm based on ACC and hierarchical clustering for big data. *International Journal of Communication Systems*, 31(16):e3488. doi:10.1002/dac.3488. E3488 IJCS-17-0750.R1.

Liang, S., Liu, Y., Wang, C., and Jian, L., 2009. A CUDA-based parallel implementation of K-nearest neighbor algorithm. In *Cyber-Enabled Distributed Computing and Knowledge Discovery, 2009. CyberC'09. International Conference on*, pp. 291-296. IEEE.

Liang, S., Liu, Y., Wang, C., and Jian, L., 2010. Design and evaluation of a parallel k-nearest neighbor algorithm on CUDA-enabled GPU. In *Web Society (SWS), 2010 IEEE 2nd Symposium on*, pp. 53-60. IEEE.

Lu, W., Shen, Y., Chen, S., and Ooi, B. C., 2012. Efficient processing of k nearest neighbor joins using mapreduce. *Proceedings of the VLDB Endowment*, 5(10):1016-1027.

Masek, J., Burget, R., Karasek, J., Uher, V., and Dutta, M. K., 2015. Multi-GPU implementation of k- nearest neighbor algorithm. In *Telecommunications and Signal Processing (TSP), 2015 38th International Conference on*, pp. 764-767. IEEE.

Patwary, M. M. A., Satish, N. R., Sundaram, N., Liu, J., Sadowski, P., Racah, E., Byna, S., Tull, C., Bhimji, W., Dubey, P. et al., 2016. PANDA: Extreme Scale Parallel K-Nearest Neighbor on Distributed Architectures. In *Parallel and Distributed Processing Symposium, 2016 IEEE International*, pp. 494-503. IEEE.

Pu, Y., Peng, J., Huang, L., and Chen, J., 2015. An efficient knn algorithm implemented on fpga based heterogeneous computing system using opencl. In *Field-Programmable Custom Computing Machines (FCCM), 2015 IEEE 23rd Annual International Symposium on*, pp. 167-170. IEEE.

Rehman, A.-u., Ullah, R., and Abdullah, F., 2015. Big Data Analysis in IoT. In *Handbook of Research on Trends and Future Directions in Big Data and Web Intelligence*, pp. 313-327. IGI Global.

Seidl, T. and Kriegel, H.-P., 1998. Optimal Multi-step K-nearest Neighbor Search. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, SIGMOD '98, pp. 154-165. ACM, New York, NY, USA. ISBN 0-89791-995-5. doi:10.1145/276304.276319.

Sohani, A., Ullah, R., Rai, A., and Karni, O., 2018. Closest Match Based Information Retrieval and Recommendation Engine using Signature-Trees and Fuzzy Relevance Sorting. *American Scientific Research Journal for Engineering, Technology, and Sciences (ASRJETS)*, 45(1):120-134.

Wang, L., Zhu, X., Yang, B., Guo, J., Liu, S., Li, M., Zhu, J., and Abraham, A., 2018. Accelerating nearest neighbor partitioning neural network classifier based on CUDA. *Engineering Applications of Artificial Intelligence*, 68:53-62.