Improve Adaptive k-Nearest Neighbor Algorithm using Multi-threading

[1]Krupali A. Hansaliya, [2]Kamal Sutaria
[1]Research Scholar, [2]Assistant Professor
[1]Department of Computer Engineering,
[1]V.V.P Engineering College, Rajkot, India

***Abstract* - The traditional k-nearest neighbor (*k*NN) algorithm is one of the oldest method for classification. It benefits from distances among examples to classify the data. The traditional *k*NN algorithm usually identifies the same number of nearest neighbors for each test examples. This is one of the limitations of traditional *k*NN algorithm which is overcome by an adaptive k-nearest neighbor (AdaNN) algorithm. The AdaNN algorithm finds out the optimal k, the number of the fewest nearest neighbor to classify each test example correctly. For classifying test examples, AdaNN set k to be the same as the optimal k of its nearest neighbor in the training data. AdaNN algorithm gives better performance than the traditional *k*NN. In this paper we improved the performance of adaptive *k*NN algorithm using multi-threading technique. The performance of the proposed algorithm is tested on several medical datasets. Experimental results indicate that our algorithm performs better than the traditional *k*NN algorithm and adaptive *k*NN algorithm.**

***Index Terms* - k-nearest neighbor algorithm (kNN), Adaptive k-nearest neighbor algorithm (AdaNN), nearest neighbors, Multi-threading**

I. INTRODUCTION

The goal of classification is to accurately predict the target class for each unlabeled example using the labeled examples present in the data. It is an active research area in information retrieval, machine learning and natural language processing [1]. KNN and support vector machine (SVM) are the most used machine learning algorithms to deal with classification. In classification, *k*NN is an easy to understand and easy to implement. It has a high accuracy as compare to the support vector machine and decision tree [2].

The k-Nearest Neighbor Algorithm

KNN is a non-parametric and also a lazy learner. It does not make any assumptions on the underlying data distribution because of its non-parametric feature. This is a major advantage of it because majority of the practical data does not obey theoretical assumptions made and this is where non-parametric algorithms like *k*NN come to the rescue [2]. Being a lazy learning algorithm it does not use the training data points to do any generalization. Lack of generalization means it keeps all the training data [2]. So, the training phase is pretty fast.

Objects are classifying in the *k*NN algorithm based on closest training examples in the feature space.

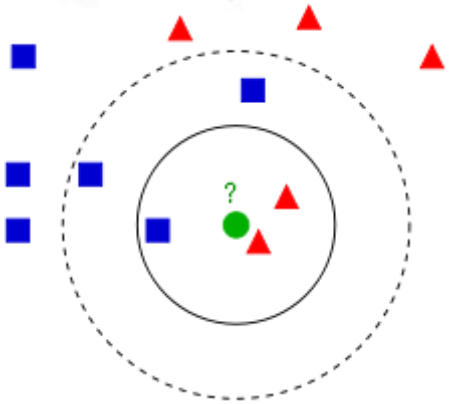

FIG.1 k-Nearest Neighbor [6]

The class of the training sample is same as the closest labeled sample in the pattern space if the value of *k* is 1 in *k*NN algorithm.

Its performance depends on the distance metric used to identify nearest neighbors. In the absence of prior knowledge, most *k*NN classifiers use simple Euclidean metric to find distance between test example and training example [1]. Euclidean distance is defined as the following formula.

$$d_{Euclidean} = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}\ , \qquad (1)$$

In the classification process, first of all *k* nearest neighbors of a test example in the training set are identified. After that according to the class labels of this *k* nearest neighbors the prediction can be made. Generally speaking, the distribution of examples in every class in the training set is uneven. Some classes may have more examples and some classes may have fewer examples than others. Therefore, the classification performance is very sensitive to the choice of the parameter *k*, fewest number of its nearest neighbor. Only odd number of k values is used in *k*NN classification, which is to increase the speed of the system.

The performance of k-nearest neighbour algorithm also depends upon choice of *k*. If value of *k* is small, and noise is present in the pattern space, then noisy samples can win the majority votes, which results into misclassification error. This can be solved with larger value of *k*. If value of *k* is large, and if the portion of the class is small, then instances of other class may win the majority votes, results into misclassification error [2]. So an improved *k*NN algorithm focuses on finding out the suitable *k*, the number of its nearest neighbors, for every test example to get its possible class label. We will describe it on the next section in detail.

An Adaptive k-Nearest Neighbor Algorithm

AdaNN algorithm is an improved *k*NN algorithm deriving from the traditional *k*NN. AdaNN algorithm works based on the principle that nearest neighbors has similar attributes, we can assume that the test example has the most similar attributes with its nearest neighbor in the training set. The probability is high that a test example adopts the same *k*NN algorithm as its nearest neighbor in the training set to get its correct class label. The optimal *k* is the number of the fewest nearest neighbors a training example has to identify to get its correct class label when assuming it is a test example to the other training examples. Therefore, we just need to get the optimal *k* of its nearest neighbor in the training set, if we want to get a test example's label.

According to the above analysis, the idea of adaptive *k*-nearest neighbor algorithm (AdaNN) shown in below TABLE I [1].

TABLE I
AN ADAPTIVE *k*-NEAREST NEIGHBOR ALGORITHM

1) 9NN algorithm:
Inputs: the whole training examples
Output: the optimal *k* of each training example
The value of *k* may be from 1 to 9. If a training example cannot get its correct class label by using *1*NN algorithm to *9*NN algorithm, we make 9 its optimal *k*.
Procedure:
a) For each training example, use the Euclidean distance metric to compute the Euclidean distances of it and the rest training examples.
b) Sort the Euclidean distances to get the training example's 9 nearest neighbors.
c) Get the training example's optimal *k* by checking from the nearest neighbor to 9 nearest neighbors.

2) AdaNN algorithm:
Inputs: the whole training examples and their optimal *k*, the whole test examples
Output: class label of each test example
Procedure:
a) For each test example, use the Euclidean distance metric to find out its nearest neighbor in the training set.
b) Get the optimal *k* of its nearest neighbor and adopt the corresponding *k*NN algorithm to get its class label for each test example.

II. COMPARATIVE ANALYSIS

Error Rate Analysis

The error rate of the traditional *k*NN algorithm is proved to be between Bayes and double Bayes [1]. Its accurate expression is shown below:

$$P_* \le P \le P_*\left(2 - \frac{c}{c-1}\right)P_*\ , \qquad (2)$$

The next is the error rate analysis of the AdaNN algorithm. Most of the examples in the training set successfully get their correct class labels by identifying their optimal *k* nearest neighbors, according to the meaning of the optimal *k*. However, as to certain training examples, they can't get their correct class for some reason. As a result, the larger the value of k assigned to them as

their optimal *k*, the higher the classification accuracy rate is in the training set. Thus, when both the number of the training examples, M, and the number of the fewest nearest neighbors, *k*, approach infinity in this manner that M→∞, k→∞, the error rate of the *k*NN algorithm for the training set approaches the optimal Bayes error rate. In the best case, the examples of the data set distribute evenly and densely in a small range. Each test example is the same as its nearest neighbor in the training set; the test example uses the same *k*NN algorithm as its nearest neighbor to get its class label. The AdaNN algorithm would perform best in this case and its error rate infinitely approaches the optimal Bayes error rate. In the worst case, all the test examples use the nearest neighbor algorithm to get their class labels, then the AdaNN algorithm degenerates into the nearest neighbor algorithm. In fact, it is impossible that all the test examples use the same *k*NN algorithm to get their class labels, because the values of the optimal *k* of each training example are different from each other. Therefore, we can conclude that the AdaNN algorithm can perform better than the traditional *k*NN algorithm but its error rate is still between the Bayes and double Bayes [1].

III. MULTI-THREADING TECHNIQUE [11]

Ability of a central processing unit (CPU) or a single core in a multi-core processor to execute multiple processes or threads concurrently, appropriately supported by the operating system is called multi-threading. Multi-threading differs from multiprocessing, as with multithreading the processes and threads have to share the resources of a single or multiple cores: the computing units, the CPU caches, and the translation look aside buffer (TLB), where multi-processing systems include multiple complete processing units.

FIG.2 Multi-threading Process [9]

Aim of Multi-threading is to increase utilization of a single core by using thread-level as well as instruction-level parallelism. These two techniques are complementary. So they are sometimes combined in systems with multiple multithreading CPUs and in CPUs with multiple multithreading cores.

The advantage of multi-threading technique is that if a thread gets a lot of cache misses, the other threads can continue taking advantage of the unused computing resources, which may lead to faster overall execution as these resources would have been idle if only a single thread were executed. Also, if a thread cannot use all the computing resources of the CPU (because instructions depend on each other's result), running another thread may prevent those resources from becoming idle.

IV. THE PROPOSED ALGORITHM

In proposed algorithm every threads do its own work and give the output to another thread and so on. Here every thread works in parallelism. For example when thread2 doing its work at a same time thread1 continue with its own work. Schematic representation explain in below figure.

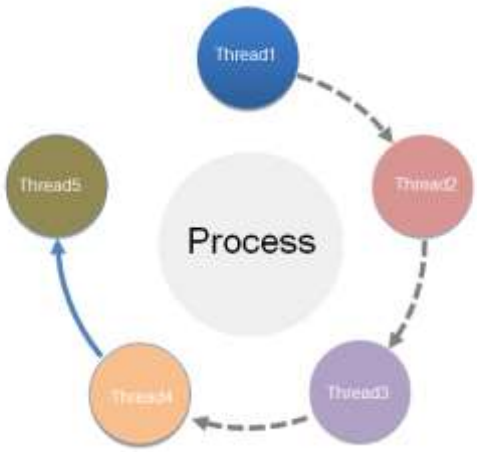

FIG.3 Proposed Method

Thread1: Take one test example and find out its nearest neighbor in the training set using distance metrics
Thread2: Find distances using distance metrics for test example's nearest neighbor
Thread3: sort the distances and take 9 nearest neighbors of that training set
Thread4: Get the training example's optimal *k* by checking from the nearest neighbor to 9 nearest neighbors and take this optimal k as test example's optimal *k*
Thread5: Get correct class label of test example using majority classes of its nearest neighbor

V. THE EXPERIMENTAL RESULTS AND ANALYSIS

Data Set Used from UCI:

1) Parkinson's disease
2) Indian Liver Patient Dataset
3) Haberman's Survival

TABLE II
DATASET INFORMATION USED IN EXPERIMENT

DATSET	CLASSES	NO. OF INSTANCES	NO. OF ATTRIBUTES	MISSING VALUES
Parkinson's disease	2	197	23	N/A
Indian Liver Patient	2	593	10	N/A
Haberman's Survival	2	306	3	N/A

As shown in fig 4 and fig 5, when using different dataset and get the accuracy and execution time for different value of k. The observation made from the experimental results was that the Multi-threading Adaptive kNN classification took high accuracy and much lesser time compared to the traditional kNN and Adaptive kNN classification.

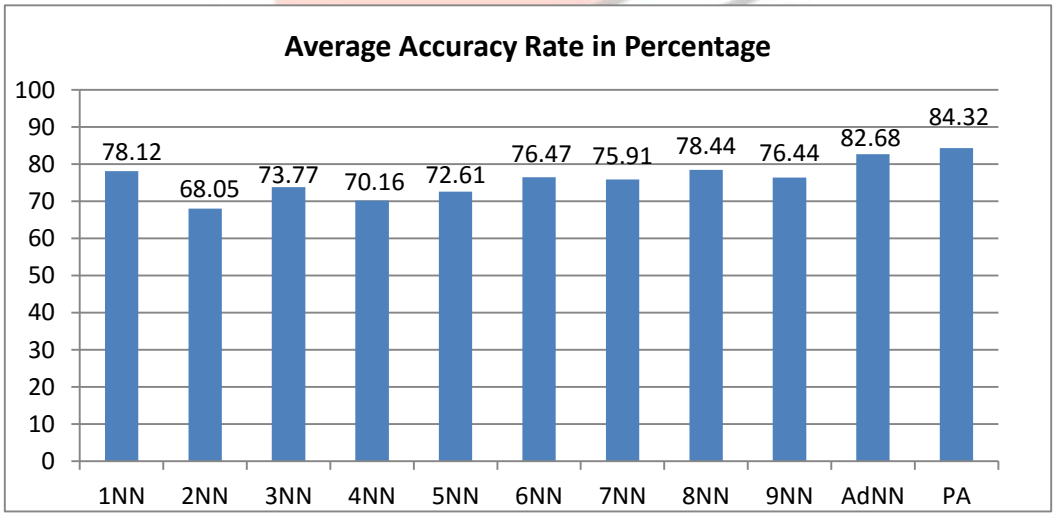

FIG.4 Output of Proposed Algorithm with Accuracy

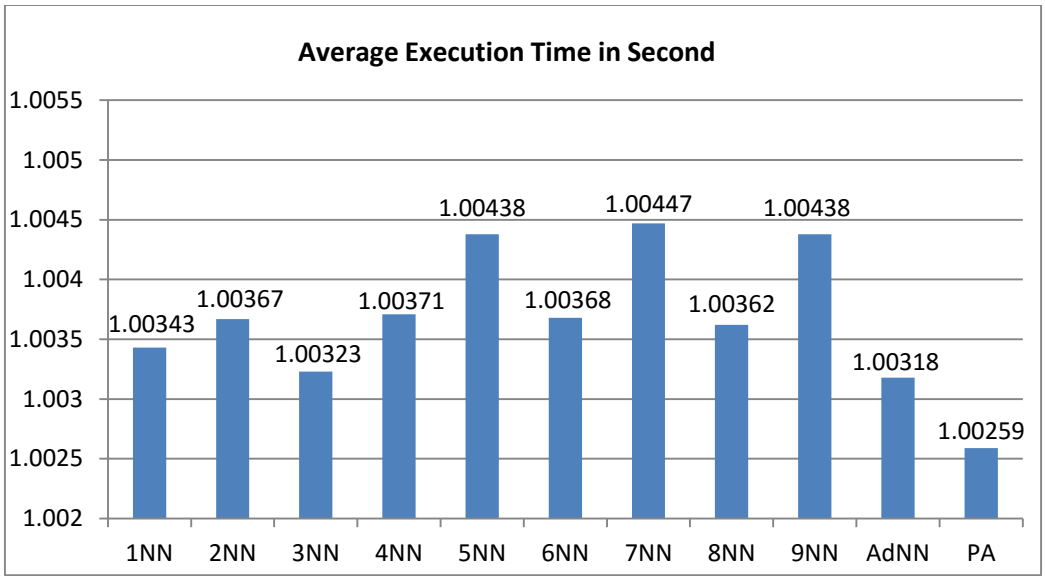

FIG.5 Output of Proposed Algorithm with Execution Time

In Figure6 average accuracy comparison of the multi-threading kNN with traditional kNN and adaptive kNN is illustrated by the graph using three different medical dataset. Proposed multi-threading algorithm has the higher accuracy as compare with the traditional and adaptive kNN algorithm.

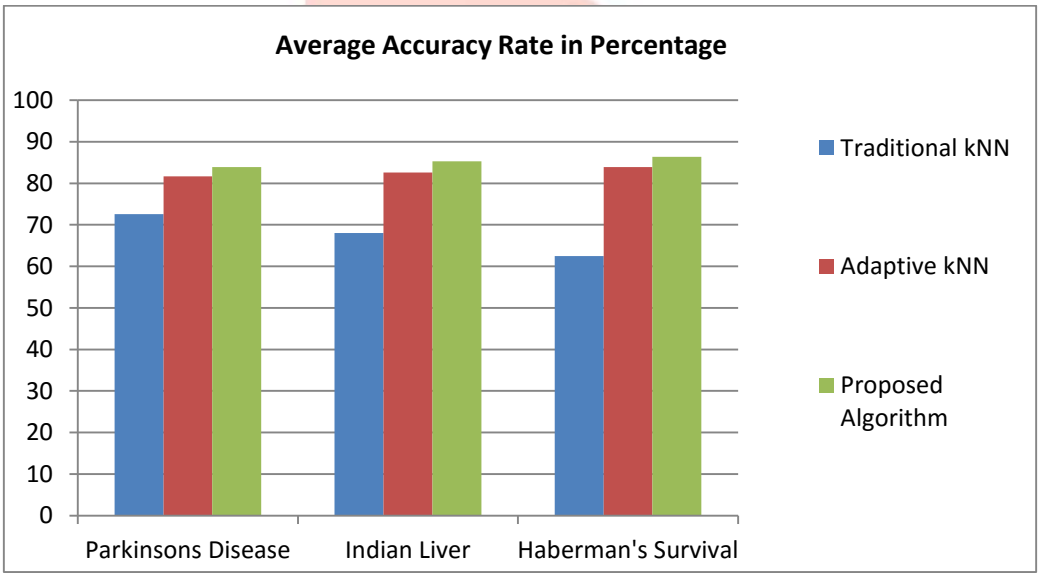

FIG.6 Comparison of Average Accuracy

VI. CONCLUSION

In this paper, adaptive k nearest neighbor algorithm is improved using multi-threading technique. Experimental result shows that the proposed algorithm is superior than the traditional kNN algorithm and adaptive kNN algorithm. Proposed algorithm took less time and higher accuracy than both existing algorithms. In this paper we are using three different medical datasets for experiment. In the future work, we will further implement other classification algorithms and conduct the experiments and consummate the parallel algorithms to improve usage efficiency, accuracy and reduce the execution time.

REFERENCES

[1] Shiliang Sun, Rongqing Huang, "An Adaptive k-Nearest Neighbor Algorithm", IEEE circuits and systems society, Seventh International Conference on Fuzzy Systems and Knowledge Discovery 2010, pg. no. 91 - 94.

[2] Xindong Wu, Vipin Kumar, J. Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J. McLachlan, Angus Ng, Bing Liu, Philip S. Yu, Zhi-Hua Zhou, Michael Steinbach, David J. Hand, Dan Steinberg, "Top 10 algorithms in data mining," Springer, 2008.
[3] Amanda K. Ziemann, David W. Messinger, Paul S. Wenger , "An Adaptive k-Nearest Neighbor Graph Building Technique With Applications to Hyperspectral Imagery", Image and Signal Processing Workshop, 2014 IEEE Western New York, pg. no. 32 - 36.
[4] Mir Yasir Umair, Kopparapu Venkata Ramana, Yang Dongkai, "An Enhanced K-Nearest Neighbor Algorithm for Indoor Positioning Systems in a WLAN", Computing, Communication and IT Application Conference, IEEE 2014, pg. no. 19 - 23.
[5] Mahmut KAYA, Hasan Şakir BİLGE, "Classification of Pancreas Tumor Dataset Using Adaptive Weighted k Nearest Neighbor Algorithm", Innovations in Intelligent Systems and Applications (INISTA) Proceedings, 2014 IEEE International Symposium, pg. no. 253 - 257.
[6]http://www.google.co.in/imgres?imgurl=https://relentlessdevelopment.files.wordpress.com/2014/05/220px-knnclassification-svg.png&imgrefurl=https://relentlessdevelopment.wordpress.com/&h=199&w=220&tbnid=aWQXFY4VVeKkhM:&docid=lkgWEauxv6aVTM&ei=yI1JVry9I4K2mwWV8Ia4Dw&tbm=isch
[7] Faruk Bulut, Mehmet Fatih Amasyali, "Locally adaptive k parameter selection for nearest neighbor classifier: one nearest cluster", Springer-Verlag London 2015.
[8] Apexa B. Kamdar, Ishan K. Rajani, "Improved Adaptive K Nearest Neighbor Algorithm using MapReduce," International Journal of Science, Engineering and Technology Research (IJSETR), Volume 4, Issue 6, June 2015.
[9] https://www.google.co.in/imgres?imgurl=http%3A%2F%2F2we26u4fam7n16rz3a44uhbe1bq2.wpengine.netdna-cdn.com%2Fwp-content%2Fuploads%2F061813_1239_Multithread1.png&imgrefurl=http%3A%2F%2Fresources.infosecinstitute.com%2Fmultithreading%2F&docid=F8yrXhdByW5jvM&tbnid=uNGPCL9eokArxM%3A&w=360&h=290&bih=657&biw=1366&ved=0ahUKEwiGicj2zcTMAhVOvo4KHYITCFUQMwhkKD4wPg&iact=mrc&uact=8
[10] Jigang Wang, Predrag Neskovic, Leon N. Cooper, "Improving nearest neighbor rule with a simple adaptive distance measure," 2006, Elsevier.
[11] https://en.wikipedia.org/wiki/Multithreading_%28computer_architecture%29