Enhancing Classification Accuracy of K-Nearest Neighbours Algorithm Using Gain Ratio

Aditya Duneja[1], Thendral Puyalnithi[2]

[1] School Of Computer Science and Engineering , VIT University, Vellore, 632014 , Tamil Nadu, India
[2]Professor, School Of Computer Science and Engineering, VIT University, Vellore, 632014, Tamil Nadu, India

---

Abstract – *K-Nearest Neighbors or KNN classification technique in machine learning is among the simplest of techniques which are used for classification. The approach used in KNN involves finding the k nearest neighbors to a test vector and then assigning a class to it based on majority voting. One of the assumptions in KNN is that data lies in a feature space, that is, measures like Euclidean distance, Manhattan distance and others can be applied on data. It has its applications in computer vision, recommender systems and various other fields. In this paper, a modification to this method has been put forward which aims at improving its accuracy. The modified version combines the concepts of distance metrics, which are applied in K-nearest neighbors, and the entropy or the strength of an attribute. The way in which the distance is calculated between the data points is altered as the strength of each individual attribute or dimension is taken into consideration. The proposed technique was tested on various UCI datasets and the accuracies of both methods, the old and the modified, are compared.*

Key Words: **classification, entropy, Euclidean distance, KNN, dimension**

1. Introduction

In machine learning, classification is a method which uses a training set comprising of various instances and their respective classes to determine the category of a new observation or a set of new observations. It is a supervised learning technique. Classification has a wide range of applications in various fields, namely medical imaging, speech recognition, handwriting recognition, search engines and many more. So, high accuracy classification models are required in order to make correct predictions in the above mentioned fields. Some of the classification models are decision trees, Support Vector Machines and K-nearest neighbor classifier.

K-nearest neighbor or KNN classifier is one of the most frequently used classification techniques in machine learning. K nearest neighbor algorithm involves the computation of the distance of the test observation with every observation in the training set. The distance which is computed is mostly Euclidean distance although other measures of distance can also be applied. Then the k nearest observations are chosen and the class which is assigned to the majority out of these k observations is assigned to the test observation. KNN makes no assumptions regarding the distribution of data, therefore, it is non-parametric. No separate training phase is defined for this algorithm as the whole dataset is used for computation of distance every time.

KNN has undergone many modifications over the years in order to improve its accuracy. Faruk et al.[1] proposed new version of KNN where the neighbors are chosen with respect to the angle between them. Even if two neighbors have the same distance from the test vector they are considered different with respect to the angle they make. Singh et al. [2] devised a way to accelerate the calibration process of KNN using parallel computing. Parvin et al. [3] modified the KNN algorithm by weighting the neighbors of the unknown observation. The contribution of every neighbor is considered to be inversely proportional to its distance from the test vector. Fuzzy K nearest neighbors was first proposed in 1985 by Keller et al.[4] Fuzzy memberships are assigned to the training data in order to obtain more accurate results as every neighbor is given a different weight. A new modification to the K-nearest neighbors algorithm is proposed in this paper taking the strength of the attributes into account. Faziludeen et al.[5] proposed a new modification to KNN known as the Evidential K nearest neighbors (EKNN). The resultant class from all k nearest neighbors of a test sample were used to find the class, not only the majority class. Distance between samples was considered to measure the weight of the evidence attributed to each class. Some research is also done to reduce the size of the dataset required to get effective results from the KNN algorithm. Song et al.[6] proposed an algorithm which removed the outlier instances from the dataset and the remaining were sorted on the basis of difference in output among instances and their neighbors. Another modification to KNN was put forward by Nguyen et al.[8] in the form of a distance metric learning method. The resulting problem was formulated as a quadratic program. Lin et al.[9] proposed a nearest neighbor classifier by combining neighborhood information. While finding the distance between samples, the influence of their neighbors is taken into account. Manocha et al. [10] proposed a probabilistic nearest neighbor classifier which used probabilistic methods to assign class membership, an aspect which KNN doesn't have. Sarkar[11] improved the accuracy of the KNN by introducing fuzzy-rough KNN classifier. It combined the conventional KNN along with fuzzy KNN. The optimal K value was not required for this algorithm to work effectively and fuzzy ownership values were calculated for each class. Timofte et al.[12] proposed Iterative nearest neighbors algorithm which combined the power of sparse representation(SR)[13] and local linear embedding[14] along with K nearest neighbors. A belief based KNN (BKNN) was introduced by Liu et al.[15] to overcome difficulties faced by the conventional KNN in classifying data points which are really close to each other. In BKNN, every point has a specific belief value for each class which helps reduce the error in classification. Zhang et al.[16] proposed a multi-label lazy learning KNN(ML-KNN) which evaluated K nearest neighbors for every instance and the Maximum posteriori principle is utilized to assign the label for the test instance. Sierra et al.[17] proposed K nearest neighbor equality(KNNE) which extended the idea of KNN . Each class was individually searched for k points and the one which had the least average distance between K points and the test instance was chosen as the class for that instance.

2. Conventional and the Proposed Methods:

KNN is a non-parametric lazy learning algorithm. The non-parametric property of KNN is useful as in real world problems too, data does not follow a particular distribution like Gaussian distribution. KNN is a lazy algorithm, meaning that the training phase is minimal and the full dataset is used as the test phase. This is unlike all the other algorithms in machine learning like Support Vector Machines. In the KNN algorithm, the distance between the test vector and the whole training set is computed and the k nearest neighbors are calculated by selecting the first k points(tuples in the training set) after sorting the distances array in increasing order . The distance metric used in K nearest neighbors can vary but Euclidean distance is the popular choice. So, the class which is assigned to the test vector is the one which is in majority out of those k points chosen.

In the ID3 algorithm, a method is defined where we pick the most suitable attribute on which the decision tree will be divided. The concept of entropy is used in the ID3 algorithm. Entropy of a dataset is the number of bits needed to express an attribute. A log function to the base 2 is used, because the information is encoded in bits. Let D be a dataset and C_i be the I th class. Let *C(i,D)* be the set of tuples of class *C(i)* in *D*. Let *D* and C(i,D) denote the number of tuples in *D* and *C(i,D)*, respectively and m denote the number of classes. *Entropy(D)* is just the average amount of information needed to identify the class label of a tuple in *D*.

$$Info(D) = -\sum_{\mathrm{i=1}}^{\mathrm{m}} (\mathrm{p_i} * \log_2 \mathrm{p_i})$$

Where p_i is the probability that an arbitrary tuple in *D* belongs to class *C(i)* and is estimated by *C(i,D)/D.*

Higher the entropy, the tougher it is to choose the class simply based on seeing the input. The dataset is highly unstable as more number of bits are needed to represent it. The formula given above is the initial entropy of the dataset. Now, to make a decision tree, we need to pick the root and at the same time ensure that the height of the tree is least. It should not overfit the data by going till the maximum depth. For that entropy of each attribute is calculated . It is given by,

$$Info_A(D) = \sum_{j=1}^{v} (|D_j| / |D|) * Info(D_j)$$

Where $Info_A(D)$ is the entropy or additional information required to make a classification for an attribute A. $|D_j|/|D|)$ is the weighted sum of the entropies of each of the v partitions of attribute A. Lesser the value of the entropy of the attribute more is the information gained as the initial entropy decreases by that account. That is why Information Gain of an attribute is given by:

$$Gain(A) = Info(D) - Info_A(D)$$

More the information gain, stronger the attribute. So, the partition takes place on the attribute which has the highest information gain. But ID3 algorithm has some drawbacks. C4.5 extends the idea of ID3 and removes its drawbacks. In ID3, there is a bias among attributes which have a large number of partitions, even though they may not be the stronger attributes than others. C4.5 removes this bias by introducing the concept of *splitinfo* of an attribute. This parameter normalizes the value of the information gain and doesn't let it get biased towards such attributes. The normalization is carried out by dividing the information gain of an attribute by its *splitinfo,* a value which is known as the gain ratio of an attribute. It is given by:

$$Gain\ Ratio(A) = Gain(A)/SplitInfo(A)$$

And splitinfo of an attribute A is given by:

$$SplitInfo_A^{(D)} = -\sum_{j=1}^{v} \left(\frac{|D_j|}{|D|}\right) * \log_2 \frac{|D_j|}{|D|}$$

The modification is as follows: While calculating the Euclidean distance of the test vector with each of the instances in the training set, we are dividing the distance between each of the attributes by the Gain Ratio of the attribute..

Say the dataset has 3 attributes . The test vector is (test1, test2, test3) and training instance is (training1, training2, training3). So the modified distance is given by:

$Dist(test, training) = (\sum_{i=1}^{k}(\frac{test(i)-training(i)}{Gain\ Ratio(i)}$ ^2)) ^1/2

where i goes from 1 to the number of attributes (or dimensions), x is the test vector and is the GainRatio(i) is the gain ratio of the ith attribute.

The intuition behind this method is that when the distance is calculated, the point with the least distance is selected as the neighbor(k times). As the distance calculated is the sum of the squares of distances between individual attributes, minimizing the effective distance of an attribute contributes in that point being selected as a neighbor. So, we are taking the strength of the attribute into consideration while we calculate the distance. That is why if the distance of an attribute is more but it is a stronger attribute than others(that is, if we choose that value of an attribute, we can tell with maximum certainty (relative to other attributes) that a particular class is going to be assigned to the test example).So, when we divide a larger number(distance) with a larger number(as a stronger attribute will have a larger value of Gain ratio), the effective distance is reduced and the stronger attributes have contributed more in the classification process. This concept is not covered in the original KNN algorithm as every attribute is given equal weightage.

3. Analysis and Result

The proposed algorithm was successfully implemented in MATLAB and was tested on 16 datasets. (table given below).

Table1: Information about datasets used for evaluation.

SNo.	Dataset name	No. of rows	No. of features	Type
1	Bal. Scale Weight & Distance	625	4	Categorical
2	Car evaluation	1728	6	Categorical
3	Contraceptive method Choice	1473	8	Categorical
4	Tic-Tac-Toe Endgame	958	9	Categorical
5	Congressional voting records	435	16	Categorical
6	Australian Credit Approval	680	14	Mixed
7	Thyroid disease	1225	21	Mixed
8	German Credit	1770	24	Mixed
9	Teaching Assistant Evaluation	151	5	Mixed
10	Heart Disease	256	13	Mixed
11	Poker Hand	1240	10	Mixed
12	Sleep	1600	13	Mixed
13	Mushroom	2500	21	Mixed
14	Credit Card	500	15	Mixed
15	Shuttle	3600	9	Mixed
16	Breast Cancer Wisconsin	730	10	Mixed

A rule of thumb in machine learning is to pick k near the square root of the size of the training set. In practice this does a good job of telling signal from noise. The algorithm was tested on categorical and mixed (categorical + numerical) datasets.

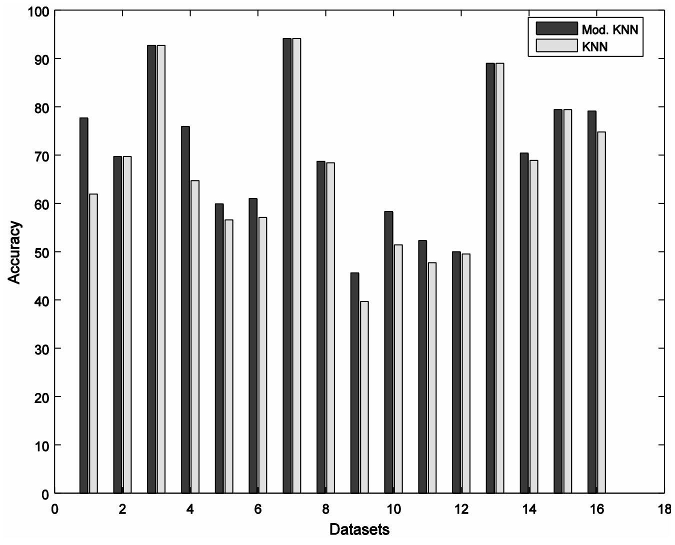
Optimal value of K, by convention, is taken as the square root of the number of instances of the dataset. The accuracy mentioned in the table given below is calculated after taking an average of 5 trials. Accuracies (in %) of modified KNN(Mod. KNN) and KNN are mentioned for these datasets.

Table 2: Accuracies of datasets for conventional KNN and the modified version of KNN.

DATASET	K	Mod. KNN(%)	KNN (%)
Bal. Scale Weight and distance	25	77.68	61.92
Car evaluation	42	69.74	69.74
Contraceptive method Choice	39	92.77	92.77
Tic-Tac-Toe Endgame	31	75.94	64.72
Congressional voting records	20	59.9	56.6
Australian Credit Approval	26	61	57.1
Thyroid disease	35	94.1	94.1
German Credit	42	68.7	68.4
Teaching Assistant Evaluation	12	45.6	39.7
Heart Disease	16	58.3	51.4
Poker Hand	35	52.3	47.7
Sleep	40	50	49.5
Mushroom	50	89	89
Credit Card	22	70.4	68.9
Shuttle	60	79.4	79.4
Breast Cancer Wisconsin	27	79.1	74.8

Given below is an illustration of the accuracy of the datasets mentioned above.

Figure 1: Accuracies of 16 datasets for KNN and modified version of KNN



4. Conclusion:

A lot of modifications made to KNN which were previously discussed are fairly complex and account for slight improvements in accuracy of its classification. But the modification made in this paper is quite intuitive and easy to comprehend. It also improves the results of the conventional KNN, slightly for some datasets but drastically for others. In the 16 datasets evaluated, 8 of the datasets showed an increase of more than 5% in their classification accuracy and 4 datasets showed an average increase of around 3%. The worst case accuracy of any dataset evaluated by the proposed algorithm is equal to that of conventional KNN. So, from the above analysis we conclude that the accuracy in the case of modified version of the KNN is equal or more than the accuracy of the original K-nearest neighbors algorithm.

5. References:

[1] Ertugrul, Ö. F., & Tagluk, M. E. (2017). A novel version of k nearest neighbor: Dependent nearest neighbor. Applied Soft Computing, 55, 480-490.

[2] Singh, A., Deep, K., & Grover, P. (2017). A novel approach to accelerate calibration process of a k-nearest neighbor classifier using GPU. Journal of Parallel and Distributed Computing, 104, 114-129.

[3] Parvin, H., Alizadeh, H., & Minati, B. (2010). A modification on k-nearest neighbor classifier. Global Journal of Computer Science and Technology.

[4] Keller, J. M., Gray, M. R., & Givens, J. A. (1985). A fuzzy k-nearest neighbor algorithm. IEEE transactions on systems, man, and cybernetics, (4), 580-585.

[5] Faziludeen, S., & Sankaran, P. (2016). ECG Beat Classification Using Evidential K-Nearest Neighbours. Procedia Computer Science, 89, 499-505.

[6] Song, Y., Liang, J., Lu, J., & Zhao, X. (2017). An efficient instance selection algorithm for k nearest neighbor regression. Neurocomputing, 251, 26-34.

[7] H. Bian,W. Shen, Fuzzy-rough nearest-neighbor classification method: an integrated framework, in: Proc. IASTED Internat. Conf. on Applied Informatics, Austria, 2002, pp. 160–164.

[8] Nguyen, B., Morell, C., & De Baets, B. (2016). Large-scale distance metric learning for k-nearest neighbors regression. Neurocomputing, 214, 805-814.

[9] Lin, Y., Li, J., Lin, M., & Chen, J. (2014). A new nearest neighbor classifier via fusing neighborhood information. Neurocomputing, 143, 164-169.

[10] Manocha, S., & Girolami, M. A. (2007). An empirical analysis of the probabilistic k-nearest neighbour classifier. Pattern Recognition Letters, 28(13), 1818-1824.

[11] Sarkar, M. (2007). Fuzzy-rough nearest neighbor algorithms in classification. Fuzzy Sets and Systems, 158(19), 2134-2152.

[12] Timofte, R., & Van Gool, L. (2015). Iterative nearest neighbors. Pattern Recognition, 48(1), 60-72.

[13] Aharon, M., Elad, M., & Bruckstein, A. (2006). $ rm k $-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. IEEE Transactions on signal processing, 54(11), 4311-4322.

[14] Roweis, S. T., & Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. science, 290(5500), 2323-2326.

[15] Liu, Z. G., Pan, Q., & Dezert, J. (2013). A new belief-based K-nearest neighbor classification method. Pattern Recognition, 46(3), 834-844.

[16] Zhang, M. L., & Zhou, Z. H. (2007). ML-KNN: A lazy learning approach to multi-label learning. Pattern recognition, 40(7), 2038-2048.

[17] Sierra, B., Lazkano, E., Irigoien, I., Jauregi, E., & Mendialdua, I. (2011). K Nearest Neighbor Equality: Giving equal chance to all existing classes. Information Sciences, 181(23), 5158-5168.

[18] Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science