JOURNAL OF SOFTWARE ENGINEERING & INTELLIGENT SYSTEMS
ISSN 2518-8739
30 April 2021, Volume 6, Issue 1, JSEIS, CAOMEI Copyright © 2016-2021
www.jseis.org

IMPROVING THE KNN ALGORITHM BY USING WEIGHTED EUCLIDEAN DISTANCE

SEYEDSOROOSH AZIZI

Purdue University Northwest, Indiana USA
Email: sazizi@purdue.edu

ABSTRACT

The k-nearest neighbors (KNN) algorithm is one of the most popular machine learning algorithms and it performs very well on many machine learning problems. KNN is a nonparametric method used for both classification and regression problems. In this paper, we introduced a modified KNN method that outperforms KNN in both classification and regression settings. In the modified KNN method we use weighted Euclidean distance rather than simple Euclidean distance to find k-nearest neighbors of each observation. Both methods are tested on three datasets, one simulated and two real datasets, and results indicate that the modified KNN outperforms the KNN method in all cases.

Keywords: machine learning; KNN method; regression; classification; Euclidean distance; k-nearest neighbors;

Academic Editor: Abid Mehmood, Saudi Arabia
Published: 30 April, 2021
Funding: Authors have no funds.
Article Type: Research Article

1. INTRODUCTION

Suppose that we observe a response variable Y and p different predictors X1,...,Xp. Machine learning revolves around the problem of prediction: produce prediction of Y from X1,...,Xp. Machine learning manages to fit complex and very flexible functional forms to the data without simply overfitting, it finds functions that work well out-of-sample [1]. In machine learning, the x-variables are usually called predictors or features or predictors and y-variable target or response variable. The focus of machine learning is to find some function that provides a good prediction of Y as a function of X1,...,Xp [2].

Suppose we have a set of training observations (X1,y1),...,(Xn,yn) where each Xi is a vector of length p that we can use to build the model. The goal is to build a model that performs well on new data. In the classification setting, the goal is to classify new observations. In the regression, the goal is to predict the response variable for new observations.

The k-nearest neighbors algorithm (KNN) is a non-parametric method used for both classification and regression. In the case of classification, KNN classifies each new observation by a majority vote of its neighbors. In other words, the new observations will be assigned to the class most common among its k nearest neighbors. In the regression setting, KNN calculates the response variable as the average (or median) of the values of its k nearest neighbors.

The KNN method uses Euclidean distance to find k nearest neighbors to each new data. However, predictors which are very important in determining the response variable and variables which are non-relevant to the response variable (the noise variables) have the same weights in the process Euclidean distance calculation. We suggest that if we assign higher weights to variables that are more important and smaller weights to less important variables, we can improve the performance of the KNN method.

The rest of the paper is organized as follows: Section 2 summarizes the literature review. Section 3 presents the methodology that we use. Section 4 is devoted to describing the datasets that are used and the results that are obtained. Section 5 provides conclusions.

2. LITERATURE REVIEW

Undoubtedly KNN is one of the most frequently used machine techniques [3-6]. Some researchers have tried to improve the KNN method by either combining KNN method with other methods or with adding some features to the KNN method. For example, Parvin et al. (2008) also try to improve the KNN method. Their main idea is classifying the test samples according to their neighbor tags [7]. This method is a kind of weighted KNN so that these weights are determined using a different procedure. The procedure computes the fraction of the same labelled neighbors to the total number of neighbors. They evaluated their method on five different data sets and concluded that their method outperforms the KNN method in terms of accuracy. Xie et al. (2016) propose an

Corresponding Author: SeyedSoroosh Azizi
Email Address: sazizi@purdue.edu

improved Spearman-distance-based KNN. To demonstrate that their modified KNN outperforms KNN they rely on simulation results [8].

Zhang and Zhou (2007) present a multi-label lazy learning approach named ML-KNN. Like KNN, for each unseen instance, its K nearest neighbors in the training set are first identified [9]. After that, based on statistical information gained from the label sets of these neighboring instances, i.e. the number of neighboring instances belonging to each possible class, maximum a posteriori (MAP) principle is utilized to determine the label set for the unseen instance. Li et al. (2001) combine a genetic algorithm and the KNN method for a classification problem [10].

Guo et al. (2003) address two of well-known difficulties with the KNN method: its low efficiency - being a lazy learning method prohibits it in many applications, and (2) its dependency on the selection of a good value for k. In their method, the value of k is automatically determined, is varied for different data, and is optimal in terms of classification accuracy. The construction of the model reduces the dependency on k and makes classification faster. They test their method on some public datasets collected from the UCI machine learning repository [11].

Deng et al. (2016) develop a method to scale the KNN method to the large-scale datasets [12]. They propose to first conduct a k-means clustering to separate the whole dataset into several parts, each of which is then conducted KNN classification. They conduct sets of experiments on big data and medical imaging data. The experimental results show that their proposed KNN classification works well in terms of accuracy and efficiency.

3. METHODOLOGY

The main purpose of the experiments is to measure the attributes of the model. The Experiment consists of five major tasks. Each task has a different set of activities to be performed by users. Each action covers a different range of fields that cover the main features of WERMSs. Table 1 below shows the tasks selected for the experiments.

In this paper, we focus on improving the KNN algorithm. KNN can be used for both classification and regression problem. Given a value for K and a prediction point (test observation) X0, the KNN algorithm first identifies the K training observations that are closest to X0, represented by N0. In the regression setting, KNN estimates f(x0) using the average of all the training responses in N_0 [13]. In other words:

$$\hat{f}(x_0) = \frac{1}{K} \sum_{x_i \in N_0} y_i \quad . \tag{1}$$

In the classification setting, the response variable for each new observation is determined by the majority vote.

When we are comparing different machine learning algorithms, how can we judge which model outperforms the other models? To answer this question we need to assess model accuracy (measure the quality of fit). In the regression setting, the most commonly used measure is the Mean Squared Error (MSE), given by

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{f}(X_i))^2 \tag{2}$$

The MSE will be small if the predicted responses are very close to the true responses, and will be large if, for some of the observations, the predicted and true responses differ substantially [13]. Another frequently used measurement to compare the performance of machine learning algorithms is R-squared. The lower is the MSE of an algorithm, the higher its R-squared would be.

How about assessing model accuracy in the case of classification problem? We define error rate or misclassification rate as the proportion of mistakes that are made if we apply our estimate f^ to the observations:

$$Misclassification\ rate = \frac{1}{n} \sum_{i=1}^{n} I(y_i \neq \hat{y}_i \qquad) \tag{3}$$

The overall performance of a classifier, summarized over all possible thresholds, is given by the area under the receiver operating characteristic (ROC) curve (AUC). ROC curve is a graphical plot that illustrates the ability of a binary classifier as its discrimination threshold is varied. The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR = 1-specificity) at different thresholds. The higher is the area under ROC curve the better is our classifier. Therefore, usually, AUC is used as the ultimate measurement to compare the performance of different classifiers.

No matter what measurement we use to assess the accuracy of our model, in either regression or classification settings, we are interested only on the accuracy of the model assessed on the new observations (test dataset). It is

easy to increase the accuracy of the model on the training dataset by establishing more complicated models. However, the accuracy of the model (measured by MSE, R-squared, misclassification rate, or AUC) on the new observations (which were not used in training the model) is the final determinant of how good the model is.

In the KNN method, K is usually chosen by resampling. Resampling involves repeatedly drawing samples from the training set and refitting a model of interest on each sample in order to obtain additional information about the fitted model. Resampling approaches can be computationally expensive because they involve fitting the same statistical method multiple times using different subsets of training data [13]. The validation set approach involves randomly dividing the available set of observations into two parts: a training set and a validation set. The model is fit on the training set, and the fitted model is used to predict the responses for the observations in the validation set. The resulting validation set error rate (for example, MSE in the case of a quantitative response variable and error rate in the case of a qualitative response variable) provides an estimate of the test error rate.

In the KNN method, the user determines K, usually with cross-validation, and for any test observation, the machine finds the K nearest neighboring points in the training dataset. The machine uses "Euclidean distance" to find the k nearest neighbors. If we assume each observation has p features (x1,..., xp) then the Euclidean distance between any test observation, like X0 = (x01,x02,...,x0p) and training observation Xi = (xi1,xi2,...,xip) is defined as:

$$d = ((x01 - xi1)2 + (x02 - xi2)2 + ... + (x0p - xip)2)^0.5 \quad (4)$$

One problem with using Euclidean distance to find the k nearest neighbors is that Euclidean distance is affected by the choice of the scale of variables. For example, if variable X1 is a monetary variable, reporting X1 in dollars or a thousand dollars would change the distance entirely. If X1 is the only feature used for predicting Y then the choice of the scale is not going to be a problem. However, if there is more than one feature for predicting Y , then the choice of X1 scale will impact the Euclidean distance, d. In other words, measuring X1 in dollars rather than thousand dollars would make (x01 −xi1)2 one million time bigger and therefore it will cast a shadow on (x02−xi2)2,...,(x0p −xip)2. In order to address this problem, we usually normalize all features (but not the target variable) before using the KNN method. Common methods to normalize data are using min-max transformer which converts all variables to 0 to 1 scale by applying $Z_i = \frac{x_i - min(x_i)}{max(x_i) - min(x_i)}$ transformer or by scaling data in a way that all variables have a 0 mean and unit standard deviation by applying $Z_i = \frac{x_i - \bar{x}}{s}$ where $\bar{x}$ is the mean of variable x and s is the standard deviation of this variable.

Now let's assume that we scaled the data, so all p features have zero mean and unit variance and we picked the optimal k, the k that minimizes test error, by cross-validation. However, some of these p features might be irrelevant to the response variable or have a very small impact on it, while some other features might be strong determinants of the response variable. The KNN method does not differentiate between these variables. All variables have the exact same weight in the calculation of Euclidean distance. Therefore, the machine might get misled when for any test observation, it tries to find the k match (k observations in the training dataset that have the least Euclidean distance from the test observation).

This problem can be clarified with an example. Assume that we want to predict wage for a group of individuals. Available features are education, age, finger size, and shoe size. Although we know that education and age have important impacts on wage but finger size and show size are irrelevant variables, the machine does not know which variables are more important in determining the wage (education, age) and which variables are less important in determining wage (finger size and shoe size). Now consider a test observation with following scaled features X0 = (0,0.5,1,1) (an uneducated middle age person with high finger size and show size) and consider these two training observations X1 = (0,0.5,0,0) (an uneducated, middle age individual with small finger size and show size) and X2 = (1,0,1,1) (a highly educated, young individual

with high finger size and shoe size). Since d0,1 = 2^0.5 and d0,2 = 1.25^0.5 the machine chooses the second person as the nearest observation to the test observation. Therefore, in the case k=1, the machine incorrectly assigns the high income of the highly educated person to the test observation. In other words, the machine overestimates the response variable for the test observation leading to high MSE.

We propose using "weighted Euclidean distance" instead of "simple Euclidean distance" in finding K nearest neighbors. In weighted Euclidean distance, after scaling all features, we assign different weights to different features based on their importance in determining the response variable. Those variables which are more important in determining the response variable will have higher weights and variables with less impact on the response variable will have lower weights. This simply means that after scaling all variables and before starting the KNN procedure we multiply each variable by its weight.

Finding the importance of variables in predicting the response variable is not an easy task. There are many methods suggested by researchers such as ridge regression and lasso regression. Also, many other machine learning techniques will provide the importance of features in predicting the response variable (like the random forest). However, in order to use a computationally inexpensive method, we suggest using the non-linear correlation between each feature and the response variable. The nonlinear correlation between variable 1 and variable 2 of a dataset "df" can be simply calculated with the R code: nlcor(df[,1],df[,2])$cor.estimate (using packages devtools and nlcor). The reason that we chose nonlinear correlation instead of linear correlation is that linear correlation only shows the linear relationship between two variables. However, in many cases, one of the important features might have a nonlinear relationship with the response variable.

4. RESULTS AND DISCUSSION

In this section, we apply both the KNN method and the modified KNN method to three datasets and then we compare the results. In order to test our models on new data, we split the dataset into two subsets: training set (a subset to train a model) and test set (a subset to test the trained model). The goal is to create a model that generalizes well to new data. Our test set serves as a proxy for new data. In this study, we apportion the data into training and test sets, with a 70-30 random split.

For both the KNN method and the modified KNN method, we use the same k which obtains from cross-validation. In both models, we normalize all features (but not the response variables). Therefore, all features would have zero mean and unit standard deviation. After normalizing all features, only in the modified KNN method, we multiply all features by their non-linear correlation with the response variable 1.

The first dataset, which is created by simulation, includes 10,000 observations, one response variable, and 9 predictors. Therefore, the training dataset has 7,000 observations and the test dataset has 3,000 observations. The data is created artificially by simulations. All variables have a normal distribution with mean 0 and variance 1. The correlation between the response variable and 9 predictors are 0.5, 0.5, 0.5, 0.2, 0.2, 0.2, 0.1, 0.1, and 0.01 respectively. The predictors are uncorrelated with each other.

Both the KNN procedure and the modified KNN procedure are implemented 10 times. The MSE for both methods and their difference are reported in Table 1.

Table. 1 KNN performance versus modified KNN performance for simulated data

Try	1	2	3	4	5	6	7	8	9	10
MSE of KNN	0.220	0.214	0.203	0.196	0.204	0.213	0.210	0.209	0.214	0.213
MSE of Modified KNN	0.169	0.165	0.163	0.157	0.161	0.169	0.171	0.165	0.157	0.169
Difference	0.051	0.049	0.041	0.039	0.043	0.044	0.039	0.044	0.058	0.045

Table 1 shows how the modified KNN procedure constantly has a lower MSE than the KNN procedure. In other words, Table 1 shows us how the modified KNN outperforms KNN when they both applied to the simulated data. In order to formally compare the results of the modified KNN with the results of the KNN method, we use t-Test for paired samples. The p-value for the test is less than 0.0001, meaning that the difference between two means is significant.

The second dataset that is one of R Built-in datasets: dataset "diamonds" which is in library ggplot2. The dataset has 53,940 observations and 10 variables about diamonds. The goal is to predict the price of diamond based on the features in the dataset. Again, both the KNN algorithm and the modified KNN algorithm are implemented 10 times. The MSE for both methods and their difference are reported in Table 2.

[1] For test observations, since we cannot use test response variable, we multiply each feature by the non-linear correlation between the training response variable and corresponding training feature. The implementing this is very easy in R. Suppose we have two datasets" "train" and "test". The goal is to predict the response variable for the test observations. Suppose the response variable is the first column of the train and test datasets. The following R code does the job:

```
for (i in 2:ncol(train)) {
train[,i] = train[,i] * nlcor(train[,1] , train[,i])$cor.estimate
test[,i] = test[,i] * nlcor(train[,1] , train[,i])$cor.estimate }
```

Table. 2 KNN performance versus modified KNN performance for diamonds dataset

Try	1	2	3	4	5	6	7	8	9	10
MSE for KNN	521	551	509	532	545	560	560	584	519	570
MSE for Modified KNN	436	458	443	450	457	465	465	496	433	466
Difference	85	93	66	82	88	95	95	88	86	104

Both MSEs and their difference are divided by 1000.

Table 2 shows how the modified KNN algorithm has a lower MSE than the KNN algorithm in all 10 trials. Based on MSE standard the modified KNN algorithm outperforms the KNN method. In order to formally compare the results of the modified KNN with the results of the KNN method, we use t-Test for paired samples and p-value for the test is less than 0.0001, meaning that the difference between two means is significant.

The third dataset is dataset "401ksubs" provided in [14] which is publicly available. The dataset has 9,275 observations and 11 variables. The goal is to predict the eligibility of individuals for 401k plan based on the features in the dataset (such as their income, age, gender, marital status, etc). The response variable is a binary variable that takes value 1 if the individual is eligible for 401k and 0 otherwise. Again, both the KNN algorithm and the modified KNN algorithm are implemented 10 times. The AUC for both methods and their difference are reported in Table 3.

Table. 3 KNN performance versus modified KNN performance for 401k dataset

Try	1	2	3	4	5	6	7	8	9	10
AUC of KNN	0.614	0.612	0.618	0.606	0.618	0.606	0.603	0.598	0.617	0.600
AUC of Modified KNN	0.633	0.630	0.634	0.621	0.620	0.624	0.622	0.607	0.624	0.611
Difference	-0.019	-0.018	-0.017	-0.015	-0.001	-0.018	-0.018	-0.009	-0.007	-0.011

Table 3 shows how the modified KNN algorithm has a higher AUC than the KNN algorithm in all 10 trials. Based on AUC measurement the modified KNN algorithm outperforms the KNN method in all 10 trials. In order to formally compare the results of the modified KNN with the results of the KNN method, we use t-Test for paired samples and p-value for the test is less than 0.0001, meaning that the difference between two means is significant. The method provided here can be used by any research that uses KNN method, for example, look at [15].

5. CONCLUSION

In this paper, we focused on improving the performance of the KNN method. KNN is one of the most popular machine learning algorithms used by many data scientists and economists. KNN is a non-parametric algorithm meaning that it makes no assumption about the relationship between the response variable and the features. KNN is used for both classification and regression problems.

In the regression setting, the goal is to predict the response variable for new observations. For any test observation that we want to predict its response variable and for a given number k KNN first finds the K nearest neighbors of that observation in the training dataset. Then KNN uses the median or mean of the response variable of those k neighbors as the prediction of the response variable for the test observation. In the classification setting, the goal is to classify new observations and KNN classifies each new observation by a majority vote of its neighbors.

The KNN method uses Euclidean distance to find the K neighbors, putting equal weights for all variables. We suggest using a modified KNN method in which variables get weighted by their importance in determining the response variable. By using one simulated data and two real datasets we show that the modified KNN method outperforms KNN.

However, there are some limitations in applying the modified KNN. The modified KNN especially outperforms KNN when some of the features are very poor predictors of the response variable or they are irrelevant to the response variable. In the absence of noise variables, neither of the modified KNN algorithm and the KNN algorithm might outperform each other.

DATA VALIDITY STATEMENT

The first dataset is created by simulation. The second dataset that is one of R Built-in datasets: dataset "diamonds" which is in library ggplot2 and publicly is available. The third dataset is dataset "401ksubs" provided by [14] which is publicly available.

REFERENCES

1. Mullainathan, S. and J. Spiess, *Machine learning: an applied econometric approach.* Journal of Economic Perspectives, 2017. 31(2): p. 87-106.
2. Varian, H.R., *Big data: New tricks for econometrics.* Journal of Economic Perspectives, 2014. 28(2): p. 3-28.
3. Antônio, W.H., et al., *A proposal of an animal detection system using machine learning.* Applied Artificial Intelligence, 2019. 33(13): p. 1093-1106.
4. Bi, Y., et al., *Combining multiple classifiers using Dempster's rule for text categorization.* Applied Artificial Intelligence, 2007. 21(3): p. 211-239.
5. Kebriaei, H., M.N. Ahmadabadi, and A. Rahimi-Kian, *Simultaneous state estimation and learning in repeated Cournot games.* Applied Artificial Intelligence, 2014. 28(1): p. 66-89.
6. Jadhav, A., D. Pramod, and K. Ramanathan, *Comparison of performance of data imputation methods for numeric dataset.* Applied Artificial Intelligence, 2019. 33(10): p. 913-933.
7. Parvin, H., H. Alizadeh, and B. Minaei-Bidgoli. *MKNN: Modified k-nearest neighbor*. in *Proceedings of the world congress on engineering and computer science*. 2008: Citeseer.
8. Xie, Y., et al., *An improved K-nearest-neighbor indoor localization method based on spearman distance.* IEEE signal processing letters, 2016. 23(3): p. 351-355.
9. Zhang, M.-L. and Z.-H. Zhou, *ML-KNN: A lazy learning approach to multi-label learning.* Pattern recognition, 2007. 40(7): p. 2038-2048.
10. Li, L., et al., *Gene selection for sample classification based on gene expression data: study of sensitivity to choice of parameters of the GA/KNN method.* Bioinformatics, 2001. 17(12): p. 1131-1142.
11. Guo, G., et al. *KNN model-based approach in classification*. in *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*. 2003: Springer.
12. Deng, Z., et al., *Efficient kNN classification algorithm for big data.* Neurocomputing, 2016. 195: p. 143-148.
13. James, G., et al., *An introduction to statistical learning*. Vol. 112. 2013: Springer.
14. Wooldridge, J.M., *Econometric analysis of cross section and panel data*. 2010: MIT press.
15. Azizi, S. and K. Yektansani, *Artificial Intelligence and Predicting Illegal Immigration to the USA.* International Migration, 2020. 58(5): p. 183-193.

AUTHORS PROFILE