Classifying waveforms using kNN mehtod

Sina HOJJATINIA [1] Mohammad Badie ALFATHI [1] Mostafa HAJJ CHEHADE [1]

Abstract

The k-Nearest-Neighbors (kNN) is an extensively used classification algorithm owing to its simplicity, ease of implementation, and effectiveness. In this report, we present the implementation of kNN and its techniques on the waveform dataset which is consisted of 3 classes of waves, and 21 attributes all include noise. We found the best k for kNN, worked on bias-variance trade-off, reduced the complexity of kNN by manipulating data, generated artificial imbalancy in our data, and compared our results with the original data.

1. Introduction

k-nearest neighbor (kNN) is one of the simplest algorithms defined as a supervised learning algorithm that belongs to non-parametric algorithms in machine learning. It is also classified as a lazy learning method. kNN has a few shortcomings affecting its accuracy of classification. It needs large memory requirements as well as high time complexity. This algorithm can be simply implemented and used for both classification and regression tasks. The kNN algorithm performs well across all parameters. However, it is commonly applied because of its simplicity of understanding and quick calculation time. It used in this project to be implemented on waveform dataset with some techniques to study the accuracy and performance in each experiment. The goals of our project presented in problem description section.

2. Problem Description

2.1. Classifying Waveforms

In this project, we aim to classify three different classes of waves using kNN algorithm. In our experiments, our goals are: (1) finding the best k in the algorithm, which has the closest accuracy to the optimal Bayes classification rate. (2) Analyzing the bias-variance trade-off, comparing true and empirical risks over different values of k. (3) applying data reduction techniques to reduce the complexity of kNN. (4) generating artificial imbalancy in our data to examine its effects on the performance of kNN.

2.2. Data Description

We used the waveform dataset that consists of 5000 samples of three different waves. Each sample has 21 attributes.

3. Experiments

3.1. Finding the best k

We used the corss-validation method to find the best value for k in the kNN algorithm. the waveform dataset is split into training, validation, and test sets. We randomly put 1000 samples of waves from the dataset into the test set. Other 4000 samples are used in training and validation sets in the 10-fold cross-validation method. we set values for k from 1 to 800, and train kNN for each k, then save all the obtained test accuracies. This algorithm is a non-deterministic algorithm, and its reuslts varies each time depending on the random state. The best k that we found between all dedicated values was 102. We plot the accuracies over number of neighbors in the figure.1.

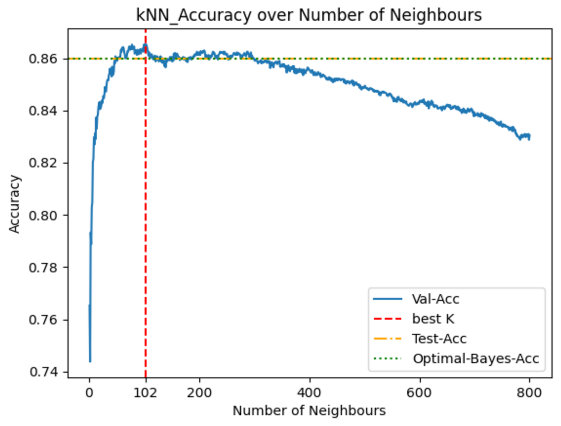

Figure 1. Validation accuracy of kNN over number of neighbors

3.2. Analysis of the bias-variance trade-off

When k increases true risk decreases and empirical risk increases. When k is small, we can see that the empirical risk is lower than the true risk, and as k grows empirical risk starts ascending and true risk descending until we get, what is explained in most data analysis courses, true risk lower than emperical risk. When k increases, we are falling more into under-sampling, which will increase the risk (higher bias, lower variance), this theory is validated in the figure where the risk reached 66.67% meaning when k=4000, the kNN is predicting the point according to the class that has the most points.

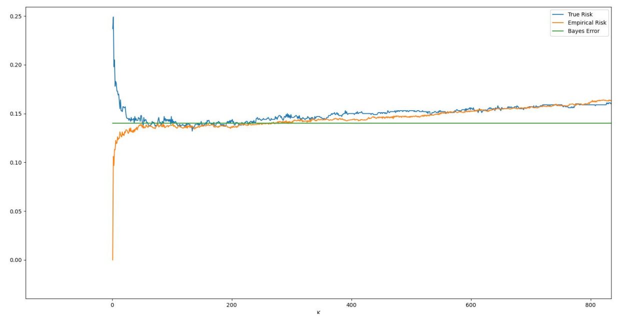

Figure 2. k-Range zoomed.

3.3. Condensed Nearest Neighbor

For data reduction, the first part is to remove the overlapping points, this step decreases the dataset length from 5000 to around 2800. In the second step of data reduction, we have to remove the points that are correctly classified, this will make the process of classifying new data points faster and with the same accuracy if we did not remove these points.

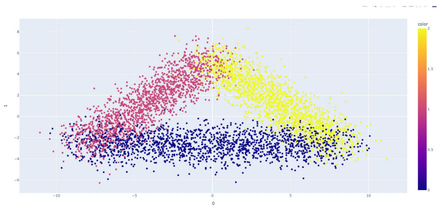

Figure 3. Original

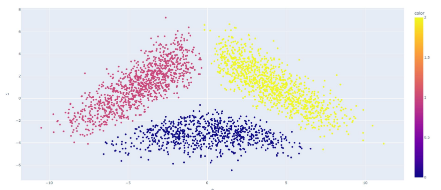

Figure 4. After Step1

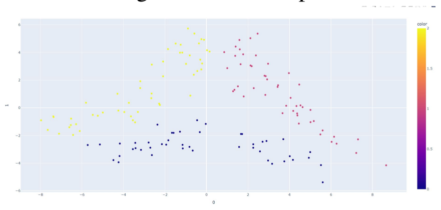

Figure 5. After Step2

3.4. Generating Artificial Imbalanced Data

We generated imbalance in the data by taking only 100 data divided equally between the first two classes, and was completely taken with 1696 data. Then fitting the new model to tune the Hyperparameters and get the best values. We get the Best leaf size as 1, Best p= 2, and the Best n neighbors= 6. By fitting kNN to the Training set, we feed Y test, and Y predict to get the confusion matrix

$$\begin{bmatrix} 6 & 0 & 7 \\ 0 & 8 & 3 \\ 0 & 0 & 336 \end{bmatrix}$$

The table below shows the analysis of the impact on the accuracy by using F-measure that is used generally for the imbalanced data:

Table 1. F-Measure.

	PRECISION	RECALL	F1-SCORE	SUPPORT
0	1.000	0.462	0.632	13
1	1.000	0.727	0.842	11
2	0.971	1.000	0.985	336
ACCURACY			0.972	360
MACRO AVG	0.990	0.730	0.820	360
WEIGHTED AVG	0.973	0.972	0.968	360

4. Results and Conclusions

As you can see in the figure 1, for all values of $k \leq 102$ the algorithm overfits the data. Because of small number of neighbors, the algorithm is too sensetive and fails to generalize well. Also, as the number of neighbors increases, when $k \geq 102$, the algorithm underfits the data, since it cannot learn the local structure of the data distribution. The best k is 102, and the algorithm generalize well by hitting optimal bayes accuracy in test time.

In choosing k, we need to focus on the true risk, and pick when it is the closest to the bayes error. As we can see from the figure 2, we need to take k where the true risk is minimal, which is around 100. As shown in the last part 1 we can find the impact on the accuracy and the compression of the performance with the accuracy.