Improving the performance of the KNN model

Moudjahed Mohamed
Beggari Mohamed Islem

Abstract

The database we are going to study are waveforms generated by a program written in C. There are 5000 instances in the database and includes 3 classes of waveforms. We will therefore try to predict the shape of a wave as well as possible with KNN

1. Introduction

The KNN (K Nearest Neighbors) algorithm is a classification algorithm that works by comparing an example to be classified to the examples in the training set. It assigns to the example to be classified the class that is predominantly represented among the K closest examples. Our study consists in improving the performance of our model in terms of accuracy and temporal complexity, whether with balanced or unbalanced data. For this we will start to see how to choose a wise k thanks to K-fold cross-validation. Then the benefits of data cleaning. The different algorithms that can accelerate the calculation of the 1-NN. And finally the impact of precision on unbalanced data.

2. Core

2.1. K-fold cross-validation

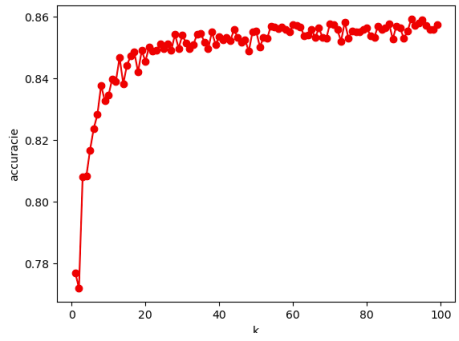

Figure 1. Testing the model for k ranging from 1 to kmax. Measures test accuracie and displays performance as a function of k

K-fold cross-validation consists of dividing the training data into K subsets, and then training the model K times, each time using a different subset as validation data and the remaining K-1 subsets as training data. The performance of the model is evaluated using the average of the scores obtained on each of the K iterations. It is used to evaluate the performance of a machine learning model, in particular to find the best K hyper-parameter in the case of the KNN algorithm. We decided not to exceed 100 for the value of k, because if k is too high the model will be sensitive to noise and less able to generalize efficiently to the test data. After testing our model, we have as best value of K: 92 and its associated score is 0.85725.

2.2. Data cleaning

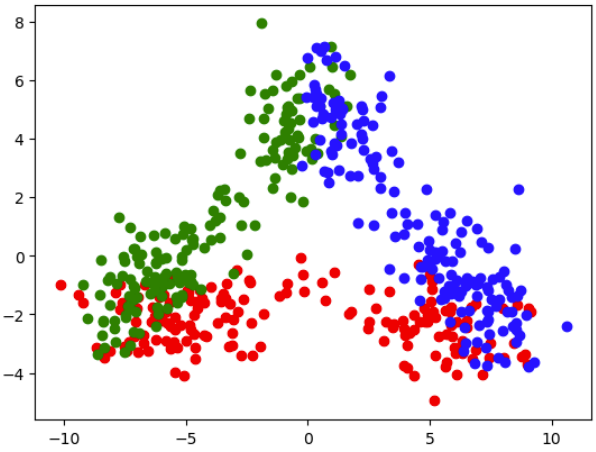

Figure 2. Visualization of data in 2D after reduction with PCA

Before training the model it is important to clean the data, as this can help reduce the complexity of the model and improve its performance. As a first step we removed outliers and examples from the bayes region because if they are included they can lead to biased results and poor generalization to the test data. After reduction we removed 817 examples from the training set and we have a 1NN accuracy of 80%. Then we remove the irrelevant examples by using the condensed nearest neighbor algorithm in order to have even less biased results avoiding over learning and hence the possibility of a lower accuracy. After the second reduction we removed 2685 examples from the reduced set and we have a 1NN accuracy of 76.9%. We finally have only 498

examples and an accuracy of 1NN that has increased by 0.6%.

2.3. Speed up the calculation of the 1NN

We can speed up the computations in terms of time, of 1-NN thanks to algorithms having a lower complexity than a brute force approach. For example by using the KD-Tree or Ball-Tree algorithm. Here in our study, Ball-Tree is used thanks to the sklearn library, it is only used to show the presence of efficient algorithms. We have implemented a simpler variant seen in class, not using a tree. This implementation is not as efficient as Ball tree using a decision tree, because it requires traversing all points on each neighbor search query. With Ball tree we decreased our execution time by 23% and with Kd-Tree by 79% compared to the brute force algorithm. A KD-tree uses a recursive division of the space into dimensions at each node, while a ball tree uses circles of variable radius to separate the space into regions which may explain the better performance of KD-tree.

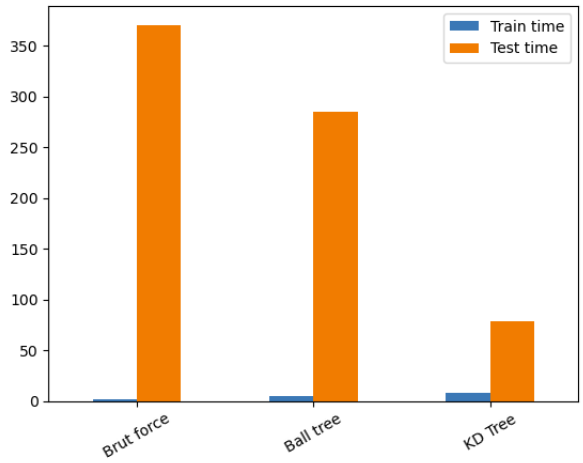

Figure 3. Measurement of the execution times of our algorithms in ms

2.4. Unbalanced data cases

We have unbalanced our data by reducing the number of examples with class 1 by 90%. When we look at the accuracy of our classifier with respect to k we notice that the smaller k is, the higher the accuracy. Accuracy may not be a good indicator of performance in an unbalanced dataset with a minority class, because the KNN algorithm can mainly predict it, which will result in high accuracy but low utility. The F1 score, which considers both precision and recall, is a better performance metric in this case, especially and that it is important to consider performance on minority classes. Therefore, it is better to set k based on the best F1 score rather than on precision alone. In our training data, k=3 is preferable to k=8 because of the best F1 score.

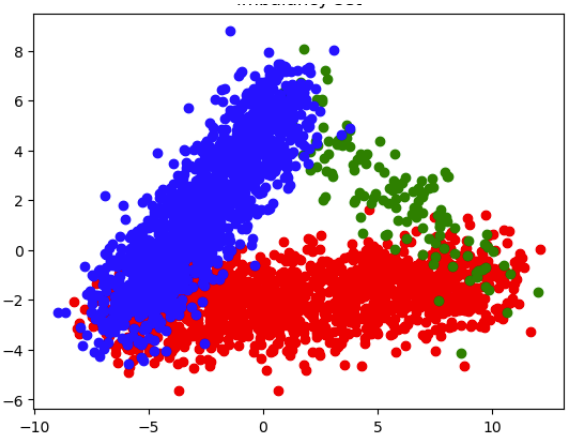

Figure 4. Visualization of unbalanced data in 2D with PCA

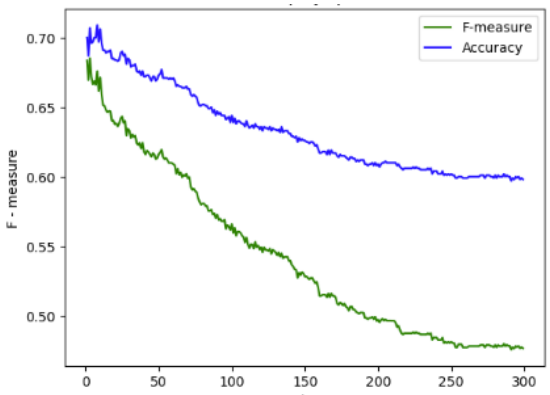

Figure 5. Model test for k ranging from 1 to 300, Measures test f measure, accuracy and displays performance as a function of k. k with the best precision is 3, it is 0.709, k with the best F1-Score is 8, it is 0.69

3. Conclusion

In summary, we examined how to improve the performance of the KNearest Neighbors (KNN) algorithm in terms of accuracy and time complexity. For this, the study used K-fold cross-validation to determine an optimal value of K, cleaned the data by removing outliers and irrelevant samples, and used algorithms such as KD-Tree and Ball-Tree to speed up the calculation of 1-NN. The study also examined the impact of precision on unbalanced data and the interest of the F1-Score. Ultimately, we conclude that these techniques improved the performance of the KNN model. The k-NN algorithm is sensitive to the scales of the variables and normalizing the variables can improve the results of the algorithm if the data is unbalanced. It may also be worthwhile to test other distance metrics and combine the k-NN algorithm with other algorithms to achieve better performance.