K-NEAREST-NEIGHBORS

Authors: G. H. Zerwes, *Universidade Federal do Espirito Santo, Vitória, Brazil*

Research Supervisor: L. Homri, *Arts et Métiers ParisTech, Metz, France*

1 Introduction

The K-nearest-neighbors (K-NN) is a versatile algorithm that can be used in unsupervised learning, supervised classification, and regression problems. It works by attributing the class of an unknown data point as the same class of its neighbors. Similarly, in the case of regression, the value of the data point is the mean value of its neighbors. The number of neighbors considered is an arbitrary choice and depends on the data being analyzed.

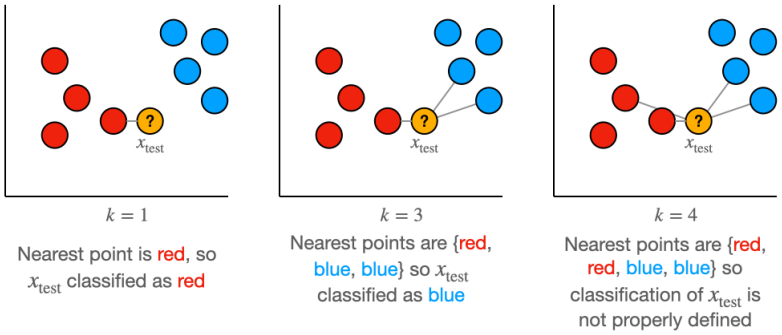

Figure 1: Example of K-NN for different number of neighbors

An example of this algorithm can be seen in figure 1. Here, the goal is to classify the unknown orange data point x_{test}. If only one neighbor is considered, that is, $k = 1$, the class will be that of the closest point. However, if k is increased to 3, the attributed class would be blue.

Another relevant concept to grasp is how the computation is made for evaluating who are the closest neighbors. A common choice is evaluating by 'brute force', which computes the distance to all other points in the dataset. This, however, can take a large amount of time if there are lots of samples with lots of features. An alternative approach is to separate the points into smaller groups and evaluate the distance in the closest groups only. This approach takes more time to construct but is faster at returning results.

2 Pseudo-code

To use this algorithm on your data, the pseudo-code below shows a procedure that should be followed.

```
#Step 1: Gather the data. The X data is the independent variable and should be already
    pre-treated. The Y array is the vector containing the labels for each class.

X = [x_value_1, x_value_2, ...]
Y = [y_value_1, y_value_2, ...]

#Step 2: Fit the model.
model = KNN.fit(X,Y)

#Step 3: Make new predictions.
y_prediction = model.predict(X)
```

3 Other recommendations

The choice of which algorithm (brute force, or tree-based algorithms) to evaluate the distances is important for a faster answer of required queries. The training time of the model should be taken into account as well. However, there is no

technique for deciding the optimal algorithm, nor the optimal number of neighbors to be considered. This should, ideally be done by cross-validation.

Although the algorithm works well with larger amounts of samples and features, performing a dimensionality reduction can improve the results of the model.

The data structure is also very influential on the speed of the algorithm. Sparse data input has higher performance on tree-based algorithms for evaluating the distance but has no influence on the brute force algorithm.

References

Bishop, C. M. (2006). *Pattern Recognition And Machine Learning*. Number 758. Springer.