KNN ALGORITHM

K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions).

|  |  |  |
| --- | --- | --- |
|  |  |  |
| A case is classified by a majority vote of its neighbors, with the case being assigned to the class most common amongst its K nearest neighbors measured by a distance function. If K = 1, then the case is simply assigned to the class of its nearest neighbor. |  |  |
|  |  |  |
|  |  |  |

It should also be noted that all three distance measures are only valid for continuous variables. In the instance of categorical variables the Hamming distance must be used. It also brings up the issue of standardization of the numerical variables between 0 and 1 when there is a mixture of numerical and categorical variables in the dataset.

The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other.

“Birds of a feather flock together.”



[Image showing how similar data points typically exist close to each other](https://commons.wikimedia.org/wiki/File:Map1NNReducedDataSet.png)

Notice in the image above that most of the time, similar data points are close to each other. The KNN algorithm hinges on this assumption being true enough for the algorithm to be useful. KNN captures the idea of similarity (sometimes called distance, proximity, or closeness) with some mathematics we might have learned in our childhood— calculating the distance between points on a graph.

There are other ways of calculating distance, and one way might be preferable depending on the problem we are solving. However, the straight-line distance (also called the Euclidean distance) is a popular and familiar choice.

The KNN Algorithm

1.Load the data

2.Initialize K to your chosen number of neighbors

3. For each example in the data

3.1 Calculate the distance between the query example and the current example from the data.

3.2 Add the distance and the index of the example to an ordered collection

4. Sort the ordered collection of distances and indices from smallest to largest (in ascending order) by the distances

5. Pick the first K entries from the sorted collection

6. Get the labels of the selected K entries

7. If regression, return the mean of the K labels

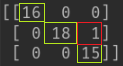
8. If classification, return the mode of the K labels

# EXAMPLE OF KNN ALGORITHM

Data set definition: Includes some features (height, width, etc.) of each flower, as well as 3 iris types containing 50 samples for each species.

In the sample application, a model will be created using the K-NN algorithm from the data file and the iris type will be estimated.

|  |  |
| --- | --- |
|  | import pandas as pd |
|  |  |
|  |  |
|  | data = pd.read\_csv('Iris.csv') |
|  |  |
|  |  |
|  |  |
|  | species = data.iloc[:,-1:].values |
|  |  |
|  |  |
|  | from sklearn.cross\_validation import train\_test\_split |
|  | x\_train, x\_test, y\_train, y\_test = train\_test\_split(data.iloc[:,1:-1],species,test\_size=0.33,random\_state=0) |
|  |  |
|  |  |
|  |  |
|  | from sklearn.neighbors import KNeighborsClassifier |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  | knn = KNeighborsClassifier(n\_neighbors=5,metric='minkowski') |
|  |  |
|  |  |
|  | knn.fit(x\_train,y\_train.ravel()) |
|  |  |
|  |  |
|  |  |
|  | result = knn.predict(x\_test) |
|  |  |
|  |  |
|  | from sklearn.metrics import confusion\_matrix |
|  | cm = confusion\_matrix(y\_test,result) |
|  | print(cm) |
|  |  |
|  |  |
|  | from sklearn.metrics import accuracy\_score |
|  | accuracy = accuracy\_score(y\_test, result) |
|  |  |
|  | print(accuracy) |

 Result of Confusion Matrix

The success rate is 49/50 = 0.98.

## References

<https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761>

<https://www.saedsayad.com/k_nearest_neighbors.htm>

<https://medium.com/@ekrem.hatipoglu/machine-learning-classification-k-nn-k-en-yak%C4%B1n-kom%C5%9Fu-part-9-6f18cd6185d>