How To Calculate K-Nearest Neighbors

Calculating K-Nearest Neighbors is a part of a Machine Learning algorithm named “K-Nearest Neighbours”. Here, ‘k’ denotes the number of nearest neighbors. k-NN is a supervised learning algorithm used for both classification and regression. People often confuse k-NN with K-means clustering.

- In k-NN classification problem, an unlabelled object is assigned to the class that has majority vote among its k nearest neighbors.
    - If k = 1, then the object is simply assigned to its nearest neighbor.
- In k-NN regression, the output is the value that is the average of the values of its k nearest neighbors. In this article we look into k-NN classification. For K-NN regression visit this link.

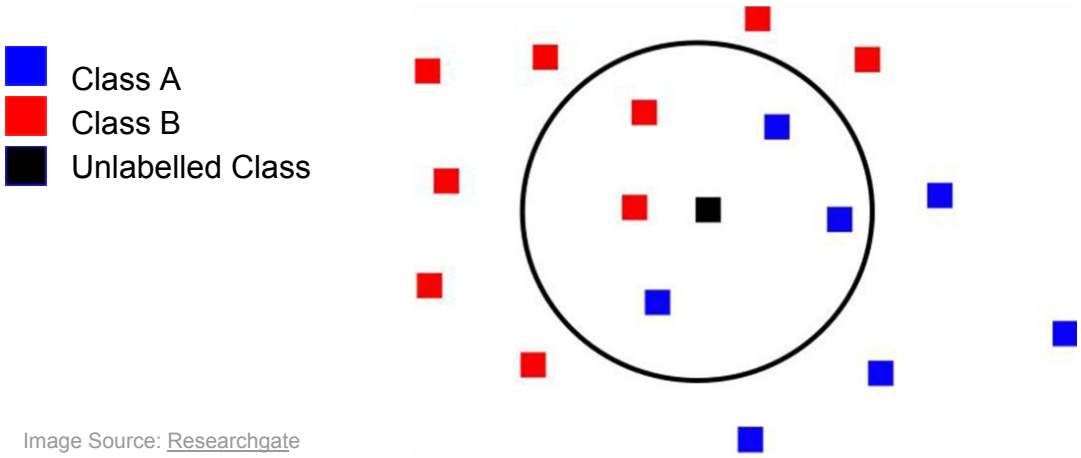


Image Source: Researchgate

The above Image shows distribution of data belonging to two class labels with ‘k’ value equal to five. Black is an unlabelled class whose label is to be determined. Out of five neighbors, three belong to Class A and two belong to Class B. Since, the majority of the votes belong to Class B so, the Black box will be classified as Class A.

Algorithm

K-NN comes under the lazy learning algorithm because it doesn’t learn a discriminative function from the training data but “memorizes” the training dataset instead.

After we have our training data-set with ‘N’ features and ‘M’ training examples, we have a matrix of size MxN. Now we can start classifying the unlabelled data and for each such request we have to:

- Find the distance vector between the item to be classified and each training data-set.
- Find ‘k’ training example with lowest distance value.

- Conduct “Majority Vote” among those 'k' training example, the dominating class will be declared the classified result.

Finding Distance Vector

There are various methods for finding the distance between two points in space. Where to use which method depends on the type of data to be processed. It is always a good idea to check the performance on cross-validation set unless one has some prior insight that clearly leads to using one over the other.

Distance measures for continuous variables

- **Euclidean Distance**: Applicable in problems like pattern recognition.

Euclidean Distance $\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$

- **Manhattan Distance**: Applicable in problems with high dimensional data.

Manhattan Distance $\sum_{i=1}^{k}|x_i - y_i|$

- **Minkowski Distance**

Minkowski Distance $\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$

Special case:

- When q=1, the distance is known as the Manhattan distance.
- When q=2, the distance is known as the Euclidean distance.
- When q --> +infinity, the distance is known as the Chebyshev distance.

Distance measures for categorical variables

- **Hamming Distance**

$$D_H = \sum_{i=1}^{k}|x_i - y_i|$$

$$x = y \Rightarrow D = 0$$

$$x \neq y \Rightarrow D = 1$$

For example,

X	Y	Distance
Male	Male	0
Male	Female	1

Distance measure when magnitude of the distance vector does not matter

- **Cosine similarity**: Applicable in problems like text mining.

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

Finding The Value Of K

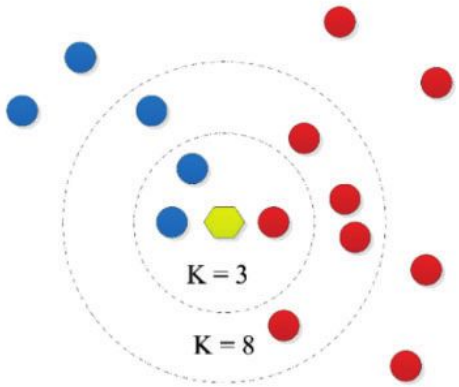
- **remender**(k, no. of classes) != 0 so that there is no tie between choosing a class. Example: Choose k as an odd number if the number of classes is two.
- Another simple approach to select k is k equal to sqrt(n).
- A small value of k means that noise will have a higher influence on the result and a large value make it computationally expensive.

With such a tradeoff, It is always a good idea to check the performance on cross validation set.

Conducting Majority Vote

Let us understand this with an example.

- Blue balls represent Male
- Red balls represent Female
- Yellow is the unlabelled data to be classified.



K	Male	Female	Majority Vote
3	2	1	Male
8	3	5	Female

When K is taken to be three, the yellow geometry is classified as male whereas with k equal to 8 it is classified as female. That is why it is always a good idea to check the performance on cross-validation set.

In this method of K-NN only a set of the training examples are used and the results changes with changing k value. A refinement of the k-NN classification algorithm is to weigh the contribution of each of the k neighbors according to their distance to the test data, giving greater weight to the nearest neighbor. If weighting is used, we can use all training examples, not just ‘k’. Thus, using all training instances, the algorithm then becomes a global one. The only disadvantage is that the algorithm will take more computational time.

Though K-NN is pretty simple and intuitive, the “prediction” step in K-NN is relatively expensive! Each time we want to make a prediction, K-NN is searching for the nearest neighbor(s) in the entire training set. However, there are certain tricks such as BallTrees and KDtrees to speed this up a bit.

Also, k-NN doesn’t perform well on imbalanced data. For example consider the earlier example of two classes, Male and Female. If the majority of the training data say 70–80% of it is classified as Male, then the model will ultimately give a lot of preference to Male no matter how small ‘k’ is. This may result in a lot of test data getting wrongly classified.

Hopefully, the article was able to introduce you getting started with k-NN. It provided the intuition of how to pick the value of ‘k’ and which approach to use for finding out the distance. There is no hardcoded way to apply k-NN for a problem, with tradeoff everywhere we have to be cautious at choosing the ‘k’ value and distance metric. Each approach is data dependent and application specific, so we have to keep verifying the performance on cross-validation set.

References(Not needed to be excluded during Plagiarism Check)

1. https://sebastianraschka.com/faq/docs/lazy-knn.html
2. https://towardsdatascience.com/introduction-to-k-nearest-neighbors-3b534bb11d26
3. https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/
4. https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
5. http://www.saedsayad.com/k_nearest_neighbors.htm
6. https://cmry.github.io/notes/euclidean-v-cosine
7. https://bib.dbvis.de/uploadedFiles/155.pdf
8. https://ieeexplore.ieee.org/document/4235253/