KNN explained

**Machine learning basics with the K-Nearest Neighbors Algorithm**

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

K nearest neighbor for machine learning

* It is a model for supervised machine learning
  + A supervised machine learning algorithm relies on labeled input data to learn a function to produce an appropriate output from unlabeled data
* Can be used for both classification regression problems
  + A classification problem has a discrete value as its output (a binary output)
* Assumes that similar things exist in close proximity – similar things are near each other
* The Euclidean distance formula is popularly used to calculate the straight-line distance between two points
* To use the algorithm, you must
  + Load the data
  + Initialize K to your chosen number of neighbors
    - Calculate the distance and the index of the sample to an ordered collection
  + Sort the ordered collection of distances and indices by ascending order by the distances
  + Pick the first K entries from the sorted collection
  + Get the labels of the selected K entries
  + Return the mode of the K labels
* To select the correct K for the data, the KNN algorithm needs to be run several times with different K values.
* Pick the K, which reduces the number encountered errors while maintaining the algorithm’s ability to predict the output when presented previously unseen data
* A too low number of K’s will mean unstable predictions
* With an increasing value of K, the predictions will become more stable
* With too many K’s the error will be increased
* The number of K’s is usually an odd number to have an overrule

**Advantages**

* Simple and easy implementable algorithm
* No need to build a model
* The algorithm is very diverse and can be used for classification, regression and search

**Disadvantages**

* The run time is severely extended as the examples and/or predictors variables increase

**KNN in practice**

* As the volume of data increases the algorithm works significantly slower, which makes it impractical

**Introduction to k-Nearest Neighbors: A powerful Machine Learning Algorithm (with implementation in Python & R)**

<https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/>

Look at 3 important aspects

1. How easy is it to interpret the output
2. Calculation time
3. Predictive power

**How does the algorithm work?**

* In a spread of red circles (RC) and green squares (GS) you want to find the class of the blue star (BS).
  + The BS can either belong to RC or GS and nothing else
  + The K in KNN is the nearest neighbors we wish to take a vote from
  + If K = 3, the three closest points are used to determine whether BS is RC or GS

**How Is the factor K chosen?**

* For a K=1 the training error will always be 0, but the test data will overfit immensely
* We want to find a point where the validation error is minimal

**NEW HEADLINE**

<https://machinelearningmastery.com/k-nearest-neighbors-for-machine-learning/>

* There is no learning required since the algorithm stores the entire dataset
* Good idea to change data since the data is stored, and the results become more reliable
* Predictions are made by searching through the entire training set the K most similar instances and defining the output based on those K instances
* The distance between the new input and the Euclidean distance is calculated between the input and the training set
* EuclideanDistance(x, xi) = sqrt( sum( (xj – xij)^2 ) )
* Afstanden kan også udregnes med:
  + Hamming distance:
  + Manhattan distance
  + Minkowski distance
* The most common used is Euclidean distance, but you can experiment with the different formulas and different K values, to find the best approximation
* The K value can be found by tuning and trying to look for the best outcome
* This is instance based learning: The training instances are used to make predictions.
* For classification KNN the output can be calculated as the class with the highest frequency from the K-most similar instances. The class with the most votes is used to predict.
* Class probabilities are calculated as the normalized frequency of samples, that belong to each class in the K most similar instances for new data: p(class=0) = count(class=0) / (count(class=0)+count(class=1))
* KNN struggles when there are a lot of inputs, as these can each be considered a new dimension. This is due to the fact that even though points are similar they can be very far from each other.

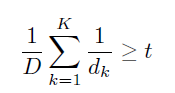
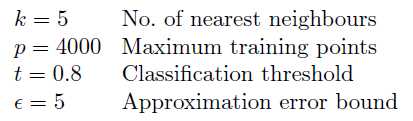
**Preparation of data**

* Rescale data: Functions better if the data has the same scale. Normalizing the data to the range [0,1] is smart. Standardize the data if it has gaussian distribution
* Address missing data: Missing data will mean the distance between the samples cannot be calculated
* Lower dimensionality: KNN is better suited for lower dimensional data.

**Spam filtering using KNN**

* An object is classified by the majority vote of its neighbors, with the object being assigned to the class most common amongst its k nearest neighbors
* In order to identify neighbors, objects are represented by position vectors in multidimensional feature space.
* The training examples are vectors in a multidimensional feature space
* The space is divided into regions by locations and labels of the training samples
* A point in the space is assigned to a specific class if it the most common label among the nearest neighbors
* The test sample is represented as a vector in the feature space
* The distance from this vector to the nearest are calculated and the new vector is labeled as a specific class
* Drawback to classify a new vector to a class is that the classes with the more frequent examples tend to dominate the prediction of the new vector, as they tend to come up in the K nearest neighbors when the neighbors are computed, due to their large representation.
  + A way to overcome this is to take into account the distance of each K nearest neighbors with the new vector and predict the class of that vector based on the distances
* The naïve version of the algorithm is easy to implement by computing the distance from the test sample to all the stored vectors , but this takes a lot for the computer especially if the training set is large

**Spam Classification using nearest neighbor techniques (Parameters – name)**

* Formula for definitive decision if an email is classified as spam 
* K is the number of nearest neighbors, D is the total distance from the point and all the neighbors. Dk is the distance between the point and the kth neighbors. T is a fixed threshold value, that measures the confidence required that a mail is spam in order to classify it as such
* 

**Spam detection Filter using KNN algorithm and resampling**

***Algorithm:***

* Make vector for every document in the test set.
* Make centriod vector for each class.
* Calculate similarity between each document vector and class vector.
* Document belongs to the class for which the similarity is maximum.