**INSTANCE BASED LEARNING**

Such methods typically build up a database of example data and compare new data to the database using a similarity measure in order to find the best match and make a prediction. For this reason, instance-based methods are also called winner-take-all methods and memory-based learning. Focus is put on the representation of the stored instances and similarity measures used between instances. There is no such thing called ‘training’ or ‘generalization’ when it goes through the training data; everything is stored in memory.

The most popular instance-based algorithms are:

* k-Nearest Neighbor (kNN)
* Learning Vector Quantization (LVQ)
* Self-Organizing Map (SOM)
* Locally Weighted Learning (LWL)

**k-Nearest Neighbors k-NN**

The model for kNN is the entire training dataset. When a prediction is required for an unseen data instance, the kNN algorithm will search through the training dataset for the k-most similar instances. The prediction attribute of the most similar instances is summarized and returned as the prediction for the unseen instance.

The similarity measure is dependent on the type of data. For real-valued data, the Euclidean distance can be used. For other types of data such as categorical or binary data, Hamming distance can be used.

In the case of regression problems, the average of the predicted attribute may be returned. In the case of classification, the most prevalent class may be returned.

**How does k-Nearest Neighbors Work**

The kNN algorithm is belongs to the family of instance-based, competitive learning and lazy learning algorithms.

Instance-based algorithms are those algorithms that model the problem using data instances (or rows) in order to make predictive decisions. The kNN algorithm is an extreme form of instance-based methods because all training observations are retained as part of the model.

It is a competitive learning algorithm, because it internally uses competition between model elements (data instances) in order to make a predictive decision. The objective similarity measure between data instances causes each data instance to compete to “win” or be most similar to a given unseen data instance and contribute to a prediction.

Lazy learning refers to the fact that the algorithm does not build a model until the time that a prediction is required. It is lazy because it only does work at the last second. This has the benefit of only including data relevant to the unseen data, called a localized model. A disadvantage is that it can be computationally expensive to repeat the same or similar searches over larger training datasets.

Finally, kNN is powerful because it does not assume anything about the data, other than a distance measure can be calculated consistently between any two instances. As such, it is called non-parametric or non-linear as it does not assume a functional form.

**Distance metrics used in k-NN**

In pattern classification, its goal is to allocate an object represented by a number of measurements (i.e. feature vectors) into one of a finite set of classes. The k-nearest neighbor (k-NN) algorithm is one of the most widely used classification algorithms since it is simple and easy to implement. Moreover, it is usually used as the baseline classifier in many domain problems (Jain et al. [2000](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4978658/#CR3)).

The k-NN algorithm is a non-parametric method, which is usually used for classification and regression problems. It is a type of lazy learning algorithms that off-line training is not needed. During the classification stage for a given testing example, the k-NN algorithm directly searches through all the training examples by calculating the distances between the testing example and all of the training data in order to identify its nearest neighbors and produce the classification output (Mitchell [1997](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4978658/#CR6)).

Particularly, the distance between two data points is decided by a similarity measure (or distance function) where the Euclidean distance is the most widely used distance function. In literature, there are several other types of distance functions, such as cosine similarity measure (Manning et al. [2008](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4978658/#CR4)), Minkowsky (Batchelor [1978](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4978658/#CR1)), correlation, and Chi square (Michalski et al. [1981](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4978658/#CR5)). However, there is no a comparative study of examining the distance function effect on the performance of k-NN.

Moreover, since the real world datasets of medical domain problems can contain categorical (i.e. discrete), numerical (i.e. continuous), or both types of data, we believe that different distance functions should perform differently over different types of datasets.

In this paper, we hypothesize that since k-NN classification is based on measuring the distance between the test data and each of the training data, the chosen distance function can affect the classification accuracy. In addition, as different medical domain problem datasets usually contain different types of data, such as the categorical, numerical, and mixed types of data, these three types of data are considered in this paper.

By using four different distance functions, which are Euclidean, cosine, Chi square, and Minkowsky, our experimental results show that k-NN by the Chi square distance function can make the k-NN classifier perform the best over the three different types of datasets. On the other hand, using the Euclidean distance function performs reasonably well over the categorical and numerical datasets, but not for the mixed type of datasets.

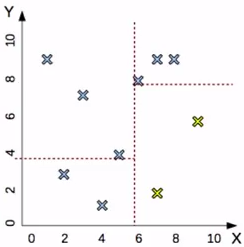
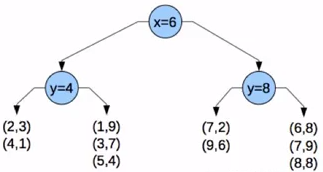
**Algorithms used to form neighbors**

**K-D Tree algorithm:**

Given a collection of data:

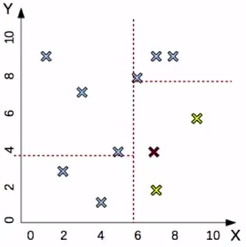
* pick an attribute/dimension
* find the median
* split the data points along that median
* repeat





***Disadvantage:***

Consider we have a new point (7,4). We will plot it as follows:



It falls in fourth quadrant having two neighbors. It is an ***approximate*** split, ***not*** a perfect one as you can see it has a closer neighbor which is in another quadrant altogether.

**LINK**:

[1]. <http://machinelearningmastery.com/tutorial-to-implement-k-nearest-neighbors-in-python-from-scratch/>

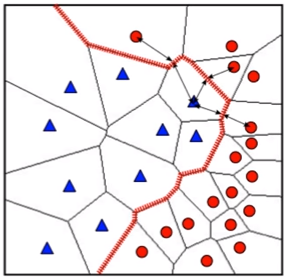
[2]. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4978658/>

[3]. <http://machinelearningmastery.com/k-nearest-neighbors-for-machine-learning/>

[4]. <https://stats.stackexchange.com/questions/99171/why-is-euclidean-distance-not-a-good-metric-in-high-dimensions>

**Another approach**

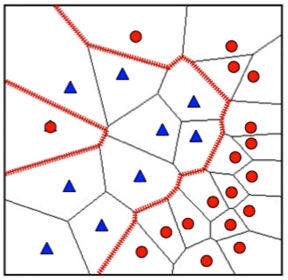
Nearest neighbors tessellate the region into different cells having a boundary that is equidistant from the neighboring training example



Sensitive to outliers:

If one blue point is replaced by a red point, there is significant change in the decision boundary. Moreover, there is no notion of a prior or confidence value for that point.

To overcome this problem, we go with k-nearest neighbors to fit the point in that region to the dominant class.



**kNN regression algorithm:**

The training examples for this case will have data and labels. The labels here will not be discrete classes but continuous values.

* For classification we take the mode (vote) of the labels nearest neighbors.
* For regression we take the mean of the labels of the nearest neighbors.

**Choosing right k value:**

* Small value -> shows high variance and freakish decision boundary
* Large value -> depict the class that is most frequently occurring.
* Cross validation selects the best value of k. this can be done by minimizing the validation error.

**Distance measure:**

Numeric values -> Euclidian distance

Categorical values -> Hamming distance

Minkowski distance

KL divergence