# K Nearest Neighbours

**Instance Based Learning**

In instance based learning, examples are just stored, and generalisation is postponed until a new instance must be classified. Instance-based techniques can construct different approximations to the target functions for each distinct query instance that must be classified. These approaches usually just construct locally optimal target functions, which work when the globally optimal target function is very complex.

Instance based learning can also use more complex, symbolic representation for instances. In case-based learning, instances are represented in this fashion and the process for identifying neighbouring instances is elaborated accordingly.

One disadvantage of instance based approaches is the cost of classifying new instances, which is very high. This is due to the fact that nearly all computation takes place at the time of classification, rather than when the training examples are first encountered.

The second disadvantage is that these approaches consider all attributes of the instances while retrieving similar examples.

**The K Nearest Neighbours Learning**

The KNN algorithm assumes all instances correspond to points in an n dimensional space R. The nearest neighbours of an instance are defined by their Euclidean distances.

For discrete valued classifications, the KNN algorithm just outputs the mode of the outputs occurred in the K nearest examples to the new example.

For continuous valued functions, the mean value of the K nearest neighbours is taken.

**Algorithm**

1. Training
    a. For each training example (x, f(x)), add the example to training_examples
2. Classification
    a. Given a query instance x to be classified:
        i. Let x1,x2...,xK denote the k instances nearest to x.
        ii. Return

$$f(x) = \underset{v \in V}{argmax} \sum_{i=1}^{k} \partial\left(v, f\left(x_i\right)\right)$$

Here, d(a,b) is 1 if a = b and 0 otherwise. (For discrete valued targets).

**Distance-Weighted KNN**

An optimisation on the KNN algorithm is to weight examples based on their distance to the query point, where nearer examples have higher weightage.

This is done by modifying the distance rule as

$$f(x) = \underset{v \in V}{argmax} \sum_{i=1}^{k} w_i \partial\left(v, f\left(x_i\right)\right)$$

*where*

$$w_i = \frac{1}{\partial\left(x, x_i\right)^2}$$

When this method is applied to all examples, this is called Shepard's method.

**Remarks on the KNN Algorithm**

1. Highly effective, inductive inference method
2. Robust to noisy training data
3. Effective when large amount of data is provided
4. Weighted KNN smoothes out the impact of isolated noise.

The bias for classification corresponds to the assumption that the classification of an instance x will be most similar to the classification of examples closest to x by Euclidean distance.

The main issue with the algorithm is that it uses all the attributes, whereas other approaches only consider subsets while forming the hypothesis. This causes the curse of dimensionality in KNN.

To counter this problem, weight the attributes differently while calculating the distance. This corresponds to stretching the axes in Euclidean space, shortening it for irrelevant attributes and widening it for relevant ones. This value is chosen to minimize the true classification error, and this true error can be estimated using cross-validation.

Another way is to drop the irrelevant columns.

Weighting the attributes by a value varying over the instance space is not done because we increase the degrees of freedom, risking overfitting. So local stretching is usually avoided.

Another issue with KNN is efficient memory indexing. Because all the processing is delayed, significant computation is required to process a new query. Indexing methods are used to reduce this computation, such as kd-trees, by storing additional variables in memory.

**Regression using KNNs**

For regression problems,

$$f(x) = \frac{\sum_{i=1}^{k} f(x_i)}{k}$$

One obvious improvement is to weight the distances of every query point by it's distance.

$$f(x_q) = \frac{\sum_{i=1}^{K} w_i f(x_i)}{\sum_{i=1}^{K} w_i}$$

*where*

$$w_i = \frac{1}{d(x_q, x_i)^2}$$