# Parametric Models Vs Non-Parametric Models

* A parametric model in supervised learning is a model which summarizes what has learnt from a training set through a vector **w** of parameters (aka weights). Therefore, a non-parametric model does not need to keep the training set in memory because it just needs to save the parameters. In parametric models, the size of the vector **w** of parameters does not grow/decrease in relation to the number of data points in the training set as the size of the vector **w** is fixed by the number of attributes that each data point contains. This feature allows parametric models to be fast when actually classifying or regressing a previously unseen data point. Furthermore, when there are a few data points in the training set using only a fixed number of parameters avoids to a certain extent overfitting. However, using a fixed number of parameters when dealing with a huge data set may be limiting. Example of parametric models are linear regression and I would add also Decision trees because unlike non-parametric models they do not need to save the training set in memory but classify an unknow data point thanks to their tree structure.
* A non-parametric model in supervised learning does not use parameters to classify the data or to regress the data but actually it keeps in memory all the data points that have been provided. Thus, the more data points there are in the training set, the more memory will be needed to save all these data points. Such an approach is called instance-based learning or memory-based learning as the data points must be kept in memory. Examples of non-parametric models are table-lookup and K-Nearest Neighbour.

## Table lookup

Table lookup consists of taking all the training examples, put them in a lookup table, and then when asked for h(x), see if x is in the table; if it is, return the corresponding y. The problem with this method is that it does not generalize well: when x is not in the table all it can do is return some default value.

Note that with h(x) we mean the hypothesis function which is the most likely function in the hypothesis space to be equal to the real f function.

## K-nearest neighbour

We can improve on table lookup with a slight variation: given a query **x**q, find the k examples that are nearest to xq. This is called k-nearest neighbours lookup. We’ll use the notation NN (k, xq) to denote the set of k nearest neighbours to **x**q.

K-nearest neighbour can be used for both classification problems and regression problems.

The basic assumption of K-nearest neighbour is that data points that have the same values should be close together.

Given that we are using K-nearest neighbour for classification then we must do the following:

1. Choose K where K indicates the number of closest neighbours to **x**q we want to consider
2. Once we have found NN (k, **x**q), the set of the K closest neighbours to **x**q, we find h(**x**q,) where h(**x**q,) = mode (NN (k, **x**q)). In other words, the value that we attribute to **x**q is the most frequent value that comes up in the set NN (k, **x**q).

It is recommended in the case of Binary Classification problems to pick an odd number as K so that there cannot be ties when computing mode (NN (k, **x**q)). Furthermore, as a rule of thumb given that N is the number of entries in the data set then k = . However, later we are going to see how to choose k using cross-validation.

Given that we are using K-nearest neighbour for regression then we must do the following:

1. Choose K where K indicates the number of closest neighbours to **x**q we want to consider
2. Once we have found NN (k, **x**q), the set of the K closest neighbours to **x**q, we find h(**x**q,) where h(**x**q,) = mean (NN (k, **x**q)) or h(**x**q,) = median (NN (k, **x**q)) . In other words, the value that we attribute to **x**q is the average or the median value of the set (NN (k, **x**q)).
3. Alternatively, we can take the set NN (k, **x**q) and transform it into a regression problem. That is we find a line h(**x**) that best fits the set and then we compute h(**x**q).

Non-parametric models like parametric models can suffer from overfitting and underfitting.

* Overfitting => overfitting occurs when the model is consistent with the training set but does not generalise well with previously unseen data. This occurs when the model has learnt patterns on the training set which are not present in the real problem and this occurs when the training set contains noise or outliers data points.
* Underfitting => underfitting occurs when the model is not consistent with the training set. This occurs when the model has not sufficiently learnt the patterns contained in the training set.

In Figure 18.26, we show the decision boundary of k-nearest-neighbors classification for k = 1 and 5 on the earthquake data set from Figure 18.15. Nonparametric methods are still subject to underfitting and overfitting, just like parametric methods. In this case 1-nearest neighbors is overfitting; it reacts too much to the black outlier in the upper right and the white outlier at (5.4, 3.7). The 5-nearest-neighbors decision boundary is good; higher k would underfit. As usual, cross-validation can be used to select the best value of k.

Thus, in the K-nearest neighbours algorithm a low value of k is likely to produce an over-fit model while a high value of k is likely to produce an under-fit model.

Chart, scatter chart

Description automatically generated

Given that the value k we choose influences whether the model will overfit or underfit the data, we need a technique which allows us to evaluate the accuracy of the model with previously unseen data.

The main technique which is used is called **CROSS-VALIDATION** which in turn can be implemented into different ways:

1. Holdout cross validation
2. K-fold cross validation

Before continuing, we need to understand that these methods aim to compare the accuracy of different models at predicting the outcome of previously unseen data after being trained with a training set.

For example, given that we have any data set of entries and we wish to know which learning model is better at being trained with the data and successfully predicting the outcome of a previously unseen data (Decision-Trees, K-nearest neighbours, Support vector machines, etc) we can use any of the two methods of CROSS-VALIDATION.

### Holdout cross validation

Holdout cross validation is the simplest because it just consists of splitting a data set of entries into two partitions:

1. Training set => used to train the model
2. Test set => used to test the accuracy of the model on previously unseen data

The disadvantage of this method is that we cannot use the whole data set to train the model. Bear in mind that the larger the data set is, the less likely the model is to overfit the data as the model learns from more data points.

Another disadvantage is that if we increase the proportion of the training set then the proportion of test set decreases and so by statistical chance we are likely to obtain an accuracy percentage which does not hold in real. Usually, the proportions of the training set is at least from 60% to at most 90%.

Once, holdout-cross validation has been performed on different models, the model with the highest accuracy rate is used and that model will be trained on the whole data set.

Thus, holdout-cross validation just gives us a suggestion which model is the most promising to learn patterns from a data set.

Another disadvantage of holdout cross-validation is that there is a chance that the model which has the most accuracy in the test set is not actually the best at generalising with previously unseen data because by misfortune it generalises the best in that test set but actually it generalises the worse in all the other possible test sets.

### K-fold cross validation

K-fold cross validation tries to remediate to the main disadvantage of holdout cross validation by measuring the average accuracy of the models to be compared in several test sets instead of just one.

The idea is that each example serves double duty—as training data and test data. First we split the data into k equal subsets. We then perform k rounds of learning; on each round 1/k of the data is held out as a test set and the remaining examples are used as training data. The average test set score of the k rounds should then be a better estimate than a single score.

Thus, at the end, the model with the best accuracy is the model which has performed better on average in all the k different test sets.

Naturally, the lower the value of k is, the higher will be the time complexity of the k-fold cross validation algorithm. Common values used for k are in the range [5,10].

The extreme is k = n, also known as leave-one-out cross-validation or LOOCV which is the most expensive k that can be chosen because the algorithm will run for n times where n is the number of entries in the available data set.

A picture containing timeline, bar chart

Description automatically generated

Thus, in the case we want to deal with a problem using the non-parametric model K-nearest neighbour and we would like to know which values achieves the best result at predicting previously unseen data we can use K-fold cross validation.

We will test through K-fold cross-validation different models which differ by the number k of neighbours and we will chose the model which will on average achieves the best accuracy and so we will know what value of K neighbours we need to construct the most promising model.

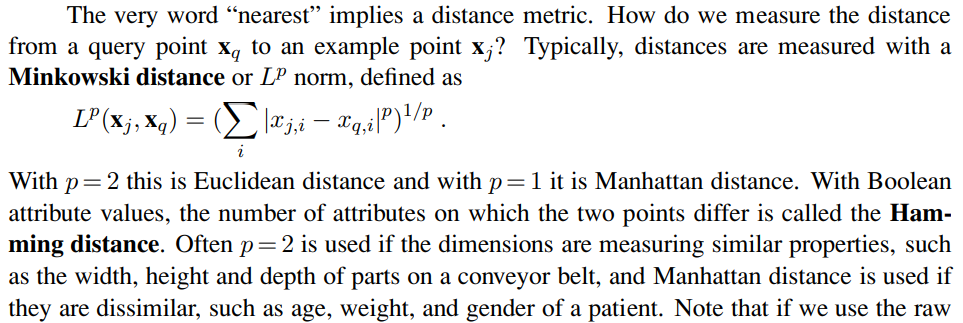
## Peeking

Peeking occurs when information about the test set has leaked into the learning algorithm. This occurs when the test set is used not only to evaluate the accuracy of a model but also to tweak the model so that it achieves high accuracy on the test set. However, this is a very bad practice because the test set is actually used to train the model and so if we use the test set to measure the accuracy of the model we do not actually see how the model performs with previously unseen data.

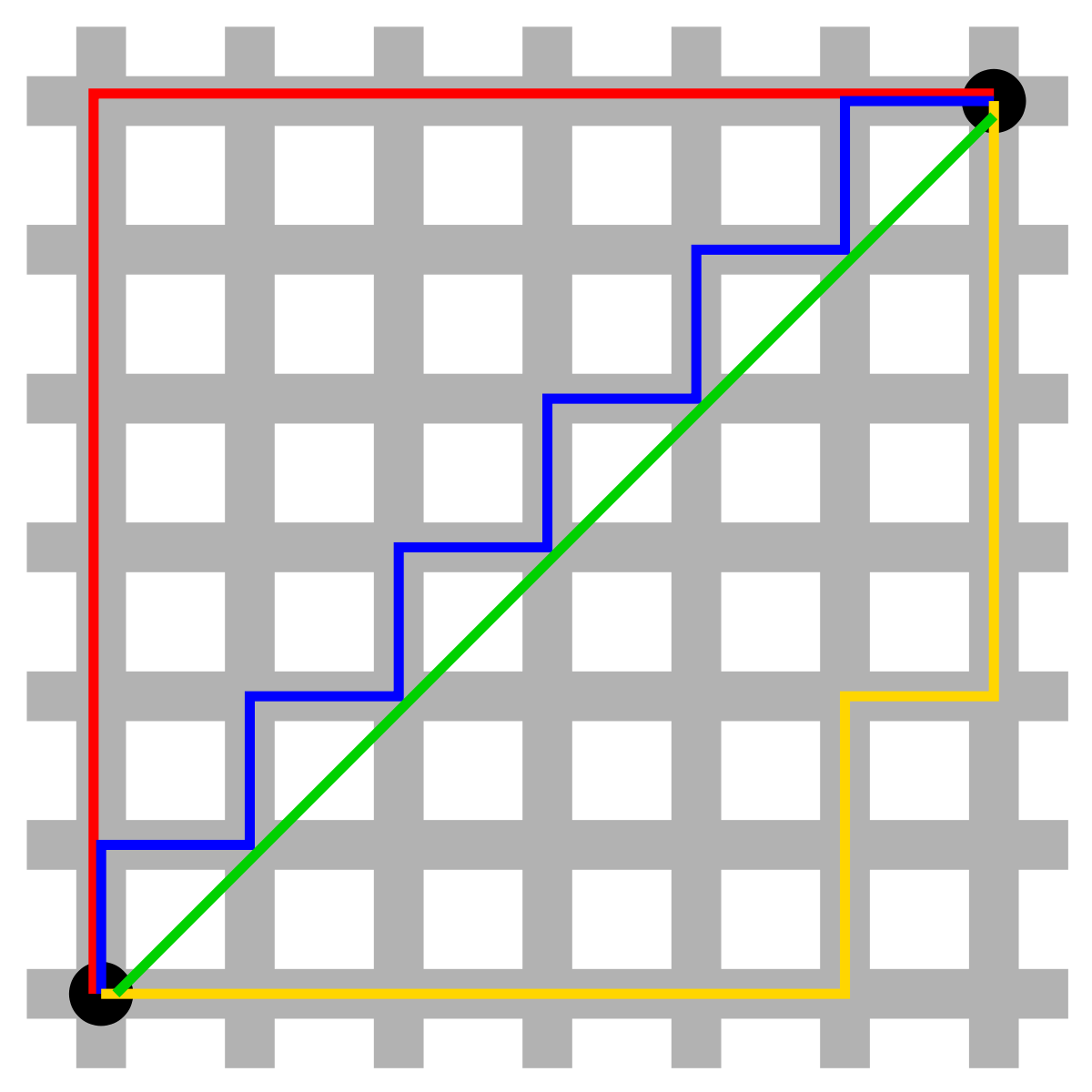
The way to avoid this is to really hold the test set out—lock it away until you are completely done with learning and simply wish to obtain an independent evaluation of the final hypothesis. (And then, if you don’t like the results ... you have to obtain, and lock away, a completely new test set if you want to go back and find a better hypothesis.)

## Distance metrics

Given that K-nearest neighbour consists of finding the k nearest neighbours to a data point, how do we compute the distance between data points?



The Manhattan distance is also called snake distance as it is possible to see in the following picture (blue line).



The green line refers to the Euclidean distance between the two points which is the shortest possible path. The red, blue and yellow lines instead are the Manhattan distance between the two points which is equal to 12 because the first data point is (0,0) and the second one is (6,6).

Given that the data points **x**q and **x**n are two vectors of Boolean values then the Manhattan distance between them computes the number of attributes where they differ. Such a measure of distance is called the Hamming distance.

As an example, the Hamming distance may be used in a small data set of movies to suggest people movies in the following way:

1. A user likes a movie X
2. Compute the hamming distance between movie X and all other movies in the data set
3. Suggest to the user the K closest movies to the movie X

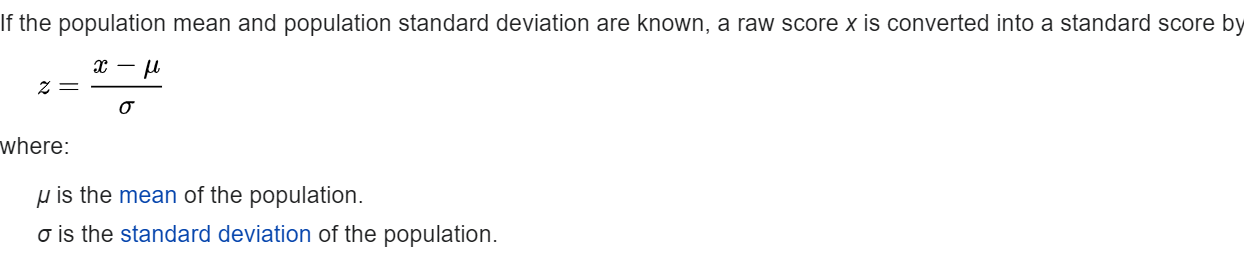
Because of the hamming distance, the k closest movies should be movies that are of the more or less the same kind of genre.

Usually, the Manhattan distance is used when the different attributes of the vector data points have different scale. However, let’s suppose that we have data points of this kind (age, sold items) where the range of age is between 1 and 100 and the range of sold items is between 0 and 1 million. As we can see the distance between two data points is more heavily influenced by the number of sold items as it has a wider range. Thus, the categorization is more heavily influenced by the number of sold items and the age of a person is less incident.

Given that cases where the different attributes have different scales are very likely to occur then in most cases we need to **NORMALISE** the data.

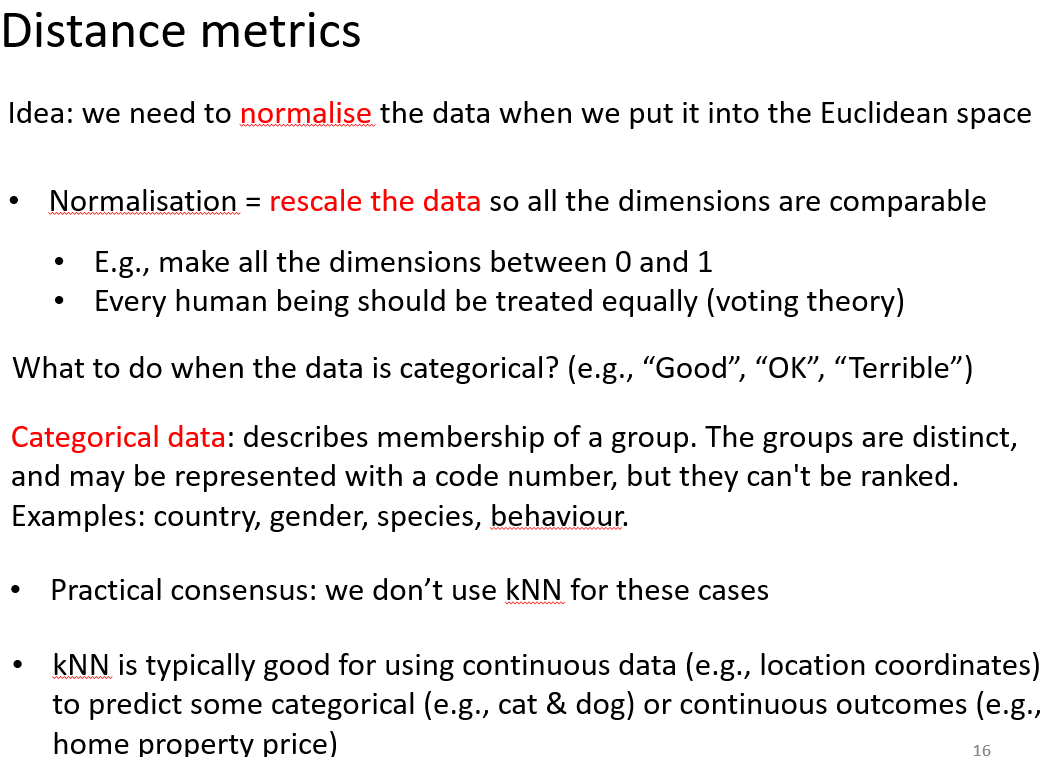
Usually, the most common technique to normalise the data in the same scale consists of computing the z-scores.

In statistics, the z-score (aka standard score) is the number of standard deviations by which the value of an observed value is above or below the mean value of what is being observed or measured.

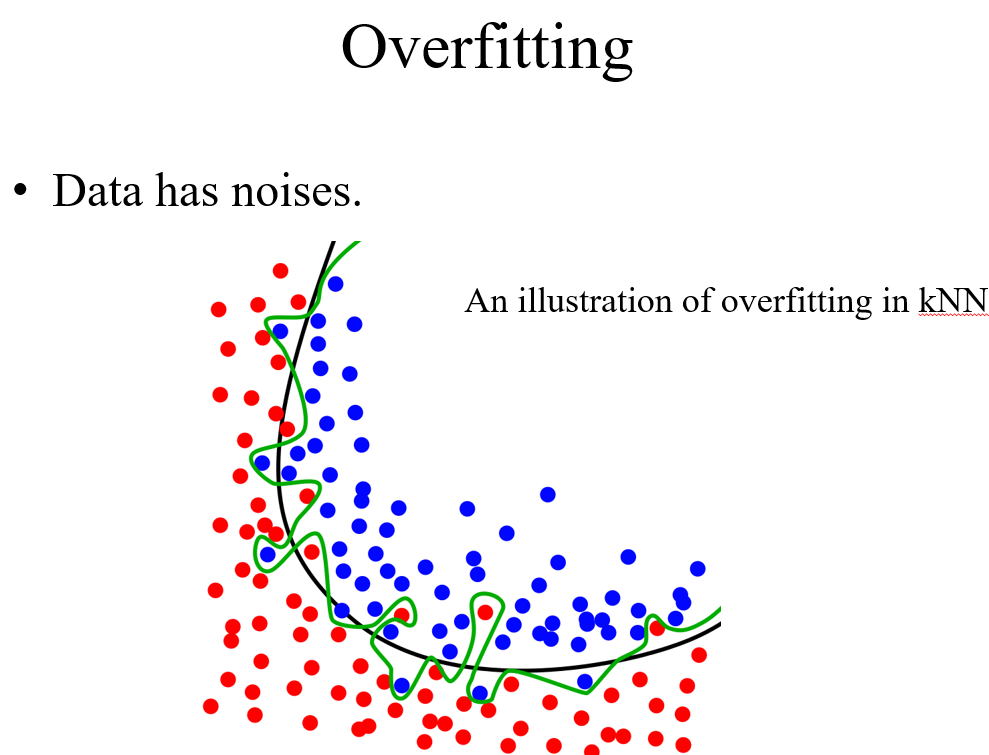
Thus, rescaling all data points to the same scale consists of :

* For each attribute X contained in the data points:
  + Compute the mean value of the attribute X population
  + Compute the standard deviation of the attribute X population
  + Compute the z-score of the attribute X for all data points

The Euclidean distance is best to be used when the data is scaled in the same way for all attributes.

Even though the slide says that k-nearest neighbours are not good for categorical data as Decision Trees can perform better in that case, what if the vector data points contain both continuous attribute and categorical attributes?

In that case, I would personally normalise the data by computing the z-score of the continuous attributes. The I would compute the distance between data points using the Euclidean distance between the continuous attributes + the hamming distance between the categorical attributes.



To continue…

1. Complexity k nearest neighbour algorithm in peter Norvig book
2. Weighted majority voting
3. Decision tree overfitting