1 What is Nearest Neighbour classification

A Mexican proverb says "Tell me who your friends are I will tell you who you are." k-NN works the same way! Given an item whose classification is not known, it counts the item's neighbours, and declares that the item is the same class as the majority of its neighbours.

More formally, nearest neighbor classifiers are defined by their characteristic of classifying unlabeled examples by assigning them the class of similar labeled examples.

Despite the simplicity of this idea, nearest neighbor methods are extremely powerful. They have been used successfully for:

- Computer vision applications, including optical character recognition and facial recognition in both still images and video
- Predicting whether a person will enjoy a movie or music recommendation
- Identifying patterns in genetic data, perhaps to use them in detecting specific proteins or diseases

In general, nearest neighbor classifiers are well-suited for classification tasks, where relationships among the features and the target classes are numerous, complicated, or extremely difficult to understand, yet the items of similar class type tend to be fairly homogeneous. Another way of putting it would be to say that if a concept is difficult to define, but you know it when you see it, then nearest neighbors might be appropriate. However, if the data is noisy and thus no clear distinction exists among the groups, the nearest neighbor algorithms may struggle to classify.

2 The k-NN algorithm

The nearest neighbors approach to classification is exemplified by the k-nearest neighbors algorithm (k-NN). This is perhaps the simplest machine learning algorithm. As noted it is still used widely due to its effectiveness.

The k-NN algorithm gets its name from the fact that it uses information about an example's k-nearest neighbors to classify unlabeled examples. The letter k is a variable term implying that any number of nearest neighbors could be used. After choosing k, the algorithm requires a training dataset made up of examples that have been classified into several categories, as labeled by a nominal variable. Then, for each unlabeled record in the test dataset, k-NN identifies k records in the training data that are the "nearest" in similarity. The unlabeled test instance is assigned the class of the majority of the k nearest neighbors.

We have hand waved over two key issues in the description above:

- *how do we determine k?*
- *what do we mean by "nearest"?*

We will add to these later in this note.

2.1 The algorithm

The following is a code-like description of k-NN algorithm. We assume that there are N data items $s_1, s_2, \ldots s_i, \ldots s_N$ in the dataset $\mathcal{S}$. The unknown entity we wish to classify is u.

- Initialise the value of k
- Initialise an empty list
- Iterate over $\mathcal{S}$
  - find the distance of each s_i from u
  - store the distance and the class of s_i in the list
- Sort the list in ascending order of distance
- Pick up the first k rows from the sorted list
- Identify the most frequently occurring class in these k rows
- Return this class as your prediction

2.2 Strengths

- Simple and effective
- Fast training phase

2.3 Weaknesses

- Does not produce a model, limiting the ability to understand how the features are related to the class
- Requires selection of an appropriate k
- Slow classification phase
- Nominal features and missing data require additional processing

3 How do we decide k?

This is one of those simple questions with no real *answer*. As you gain experience in Machine Learning and the domain you learn to figure methods of estimating k. Some simple observations can be made.

- When doing binary classification, k must be odd
- Smaller k tend to make the classification very dependent on local data and possibly noise.
- Larger k tend to provide a smoother fit and is more resilient to outliers.

Technically k is what we refer to as a **hyper parameter** which needs tuning as per the problem.

4 Nearest?

We noted earlier that the concept of distance between the unknown entity u and the entities in the dataset s_i is central to the whole algorithm.

In simple everyday cases (2 and 3 dimensions) we can easily and intuitively associate a distance between two points. In Machine Learning problems we are often dealing with many dimensions– 100's of dimensions is quite usual. So we need to get comfortable with more mathematical notions of distances as well as the notions of alternate definitions of distances. We also need to get comfortable with the standard notations.

4.1 Euclidean Distance

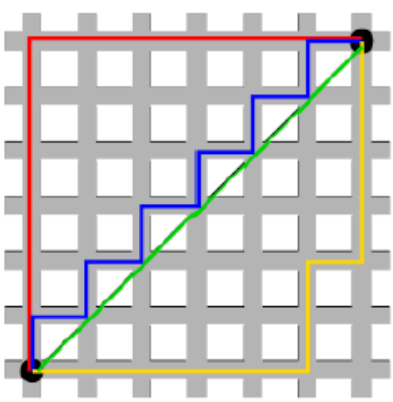
The usual distance between two points on a plane is generalized to the distance metric called the **Euclidean** distance. Since the number of dimensions is large, we cannot use individual symbols such as x, y, z to represent different dimensions. So we switch to calling all points as x and their indiviual components as x_i. That is the point x in N-dimensions has the coordinates: $(x_1, x_2 \ldots x_N)$. In this notation, the euclidean distance between two points x and y is:

$$d(x, y) = \sqrt{\Sigma(x_i - y_i)^2}$$

4.2 Manhattan Distance

Also called the taxicab distance. The reason behind the names is that in a city you can only travel along the roads, and not fly from point x to y. So while the euclidean distance measures the length of the straight line between x and y the manhattan distance measures the length of the actual path between two points if you are restricted to move only along one of the axes.

$$d(x, y) = \Sigma|x_i - y_i|$$



This picture (source: Wikipedia) shows the difference between the two. The diagonal line is the euclidean distance. All the step like paths are the same manhattan distance between those two points. In other words, the euclidean distance measures the distance "as the crow flies", while the manhattan distance measures the distance someone has to drive on the road in an idealized city with parallel grid like roads.

4.3 Other distances

These are applicable for real valued x and y. In Machine Learning problems we often have to define distances between entities such as strings or colors. We mention the names of some other possibilities. Please consult the references for further information.

Chebyshev $d(x, y) = max(|x_i - y_i|)$

Minkowski $d(x, y) = \sqrt[p]{\Sigma(|x_i - y_i|^p)}$

Cosine $d(x, y) = \frac{x \odot y}{|x|.|y|}$

Hamming Measure how many attributes must be changed in order to match x and y.

Jaccard $d(x, y) = 1 - \frac{|x \cap y|}{|x \cup y|}$

References

The blog `https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/` is a good guide for kNN in general.

`http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.DistanceMetric.html` has the complete list of distance metrics implemented in the most popular classical ML library: scikitlearn