Introduction to data classification problems

Wojciech Czaja

November 4, 2015

Norbert Wiener Center
for Harmonic Analysis and Applications

Data classification as a modeling problem

The underlying principle of Machine Learning is that the rules about data must be inferred from the data itself, rather than from some a priori models. This means, for example, that when we have a large and complicated data set (possibly consisting of heterogeneous subsets), it may be impossible to come up with a reasonable set of differential equations (or other models) to describe this data. (You may think of a contrast between problems in, e.g., population dynamics or epidemiology, vs. a need to better understand individual's chances for falling ill to a particular disease.) Instead, we are forced to analyze the structure of the data. This typically takes the form of a *classification* problem.

Data classification

In machine learning, **classification** is the problem of identifying the specific subset of the data to which a new observation belongs, based on the basis of a training set of data containing some preliminary observations whose label/membership is known.

Alternatively, we may think of **clustering**, which is defined as dividing the data set into subsets (categories), based on some measure of inherent similarity. In such case, the attributes of an individual data point are decided based on membership in one of the aforementioned categories.

These two very similar concepts, are actually examples of very different philosophies. Classification is an example of **supervised learning**, whereas clustering is called **unsupervised learning**.

Data classification vs other problems

The problems of classification and clustering belong to a large family of similar mathematical ideas. Some other related concepts include:

- Segmentation;
- Quantization;
- Regression:
- Feature Detection;
- Pattern Analysis.

Nearest Neighbor Search

Nearest neighbor search (NNS), also known as the closest point search, is a problem of finding the closest points in a certain, a priori defined, metric. Formally, the nearest-neighbor (NN) search problem is defined as follows: Given is a set S of points (possibly infinitely many) in a space M and a query point $q \in M$; The problem is to find the closest point $p \in S$ to q by means of a similarity metric. It is often referred to as the post-office problem. In typical situations, the space M is a metric space and the similarity is defined by the distance metric on M, which must be symmetric and must satisfy the triangle inequality.

The simplest solution to the NNS problem is to compute the distance from the query point q to every other point in the data set S, keeping track of the best result. This algorithm, sometimes referred to as the naive approach, has complexity $O(dN)$ where N is the cardinality of S and d is the dimensionality of M. There are no search data structures to maintain, so linear search has no space complexity beyond the storage of the database.

Euclidean vector space

A common example of M is the d-dimensional vector space with a metric. Examples of metric include (but are not limited to) the Euclidean distance, Manhattan distance, or other ℓ_p distance metrics. Recall that

$$dist_p(x,y) = \|x-y\|_p := \left(\sum_{n=1}^{d} |x_i - y_i|^p \right)^{1/p}, \quad p \geq 1.$$

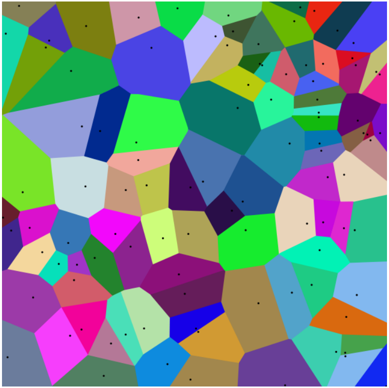
When we fix $p = 2$ in the definition of the p-norm:

$$\|x\|_2 = \sqrt{|x_1|^2 + \ldots + |x_d|^2},$$

which is the standard Euclidean norm for the d-dimensional vector space. This is the same norm that is used in the least squares optimization problems.

Piecewise Constant Interpolation

Piecewise constant interpolation is actually an illustration of the nearest neighbor search. But it is also an example of a nearest neighbor classification.



Source of imagery: Wikipedia

Nearest Neighbor Classification

The preceding picture can be interpreted as follows: the set M is the square; the black dots form the set S, and the metric is the Euclidean distance. If we consider the set S to be the set of labels (represented by different colors), then the classification problem for M is to provide such label to every point in M. This is done by means of nearest neighbor search for each $q \in M \setminus S$, and by assigning the label of the nearest neighbor.

As is evident from the picture, there are instances when a point q may have 2 or more nearest neighbors in S, all separated by the same distance. In such cases some other method must be employed to determine the class membership, e.g., for finite sets S, we can index their members by natural numbers and use the indices of the nearest neighbors to determine the classification.

k-Nearest Neighbor Classification

The **k-Nearest Neighbors algorithm** (k-NN) is an algorithm used for classification of query points $q \in M$. The input of this algorithm are:

- a training set S with its elements labeled with a set of N labels (more than one point may be labeled with the same label);
- a query point $q \in M$;
- a metric on M;
- positive integer k, typically small.

The output is a class membership for q. An object is classified by a majority vote of its k nearest neighbors, with the object being assigned to the class most common among its neighbors. If $k = 1$, then the object is assigned to the class of that single nearest neighbor, as described previously.

kNN Algorithm

Let S denote training set and T denote testing set. The simplest scenario is to consider the training set just as a set of labels, and the testing set to be the set of query points:

```
For all $t \in T$ do
    For all $s \in S$ compute $Dist(t, s)$
    Sort the set $\{Dist(t, s) : s \in S\}$ from smallest to largest
    Select first $k$ elements of $\{Dist(t, s) : s \in S\}$
    Perform a voting procedure to determine the class of $t$
```

The output is a class membership for every $t \in T$.

Optimization in kNN Algorithm

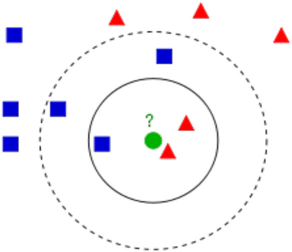
In typical situations the preceding algorithm needs to be optimized. That is the testing set is used to find the optimal selection of the number of nearest neighbors, metric, size of the training set, etc (collectively referred to from now on as *parameters*), to maximize the classification performance. And then classification is performed on the testing set with the choice of optimal parameters.

In such situations, for a large collection of parameter choices we classify each element of the training set using its k nearest neighbors within the training set. When the resulting classification matches the label, we consider this a success, and otherwise we speak of a failure. Computing the percentage of correctly classified points is a measure of success. Associated concepts include **sensitivity, specificity, precision, recall, false positives, false negatives, true positives, true negatives**, etc etc.

Note that the success rate may be computed globally, or for each class individually, or, e.g., we could compute the average of class success rates.

Example

In the following example the query point q is represented by a green dot, and the training set S consists of two classes of blue and red points. The membership of q depends on the choice of k. If $k = 3$, then q is assigned to the red class, and if $k = 5$, then q is assigned to the blue class. (Naturally, the classification depends also on the choice of metric.)



Source of imagery: Wikipedia

Voting

Even in the case where there are only 2 classes, it does not merely suffice to choose an odd k to secure that the simple majority voting procedure is enough. For example, it may happen that several points will be equidistant from our testing point, and a down-selection might be necessary.

For the cases where there are many different classes to choose from the situation is proportionally more complicated.

Thus we may need to come up with more sensitive voting schemes, and the classification results, will depend on the voting procedure in addition to the label set selection, number of nearest neighbors, and choice of metric.

Weighted Voting

Another example of a possible problem with the simple majority voting classification may occur when labels from a more frequent class dominate the prediction of the new query, because there are so many of them among the k nearest neighbors. To overcome this problem we may weight the classification, by taking into account the distance from the query point to each of its k nearest neighbors. The label of each of the k nearest points is multiplied by a weight proportional to the inverse of the distance from that point to the query point.