MA2823: Foundations of Machine Learning
Chapter 7: Nearest Neighbors

Lecturer: Chloé-Agathe Azencott Scribes: Hugo Perrin & Nathan Vermeersch

In this chapter we will see:

— how to implement the nearest-neighbor and k-nearestneighbors algorithms;

— how to compute distances between real-valued vectors as well as objects represented by categorical features;

— how to define the decision boundary of the nearestneighbor algorithm;

— why kNN might not work well in high dimension.

Introduction

Looking back at regression methods, we can see some of its flaws as a parametric model. Indeed, as a parametric method, the space induced by the model we feed the algorithm is smaller than the space of all possibilities. Since it can only be a subset of all possibilities, we can conclude that another method can produce better predictions. Hence the need for non parametric models. They won't give us a simple explanation of the underlying laws which could be present to predict the result (like a law of prediction), however since they have less constraints on how the prediction is made from parameters, we can hope they'll give us better predictions. Drawbacks of this approach would be that the prediction complexity depends on the number of data points and not only the number of features. This is usually coupled with the fact that the decision function is expressed from the data points. Because of this, some simplifications can be made to reduce the computational complexity of the algorithm (space partition etc.). The following chapters will deal with a few non parametric algorithms (knn in this chapter, tree based methods in the next one, svm after that etc.).

In the current chapter, we will also investigate a model which is part of instance-based learning methods. This class of methods is based on the principle of analogy that the inductive model primarily uses (it is noteworthy that regression was based on the hypothetico-deductive principle). Here we analyze patterns based on habits, just as a doctor would. Indeed, the doctor examine her patient (the data point to which we want to predict the label) and discovers his symptoms (the features), then if she find the patient has similar symptoms as he one having the flu, she can conclude he has the flu (his assigned label) as well. Therefore during the learning period, we don't want to produce an actual meaningful sense, so the only thing we need is to store the training points with their labels. The training period can be accompanied with the optimization of a few parameters during the cross validation process, but there aren't more overheads involved in the training. The computational overhead is moved to the prediction process. Because of the fact it doesn't need more for the training, we call those methods "'lazy"'. However, the other cost of that training simplicity is that it can be hard to share or store, since the size of information needed depends on the number of data points. Unlike regression which only stores the same number of outputs than there are features, this method stores everything. When shared, it also means we have to give up the entire data set, which can cause issues if there is confidentiality involved.

We will first define mathematically the notion of similarity, and its link to distances in some cases. Then we will present the k nearest neighbors algorithm, how it can be optimized. Finally, we will discuss its strength and weaknesses.

1 Similarities

1.1 Mathematical foreground

Definition A distance $d : E \times E \to \mathbb{R}$ on E is a symmetrical form such as:

$$\forall (x, y) \in E^2, d(x, y) = d(y, x)$$

$$\forall (x, y, z) \in E^3, d(x, y) \leq d(x, z) + d(z, y)$$

$$\forall x \in E, d(x, x) = 0$$

Example We can derive a distance d from a norm N on E:

$$d(x, y) = N(x - y)$$

Application With a distance defined like that, we have an example of how to set similarities between instances of $\mathrm{R}^n(\mathrm{n} \geq 1)$. Similar points will be points which will close to one another. This makes sense a lot for instance in a case where close data points have similar labels.

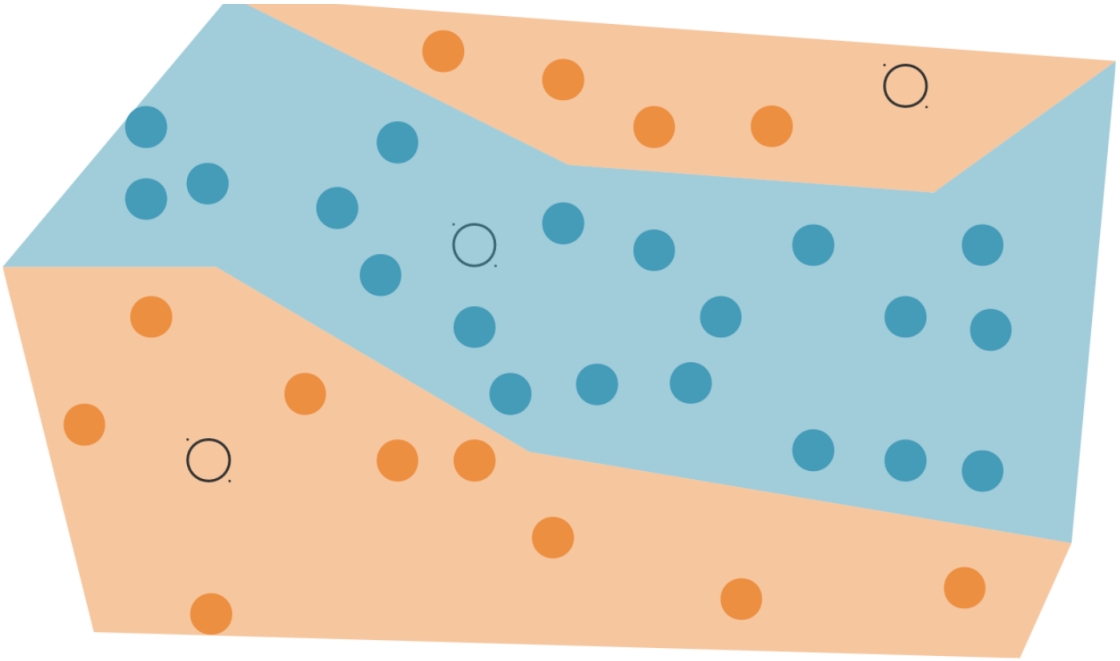

Figure 1: Partition where this method is useful

1.2 Discussion around different distances

Definition We want to present the different norms which are the most used, and how we can modify them slightly to modify our results.

$$L^p \; distance \; noted \; d_p, \; with \; p \geq 1$$

$$d_p(x, y) = \|x - y\|_p = (\sum_{k=1}^{n} |x_k - y_k|^p)^{1/p}$$

Examples We often use the Euclidean distance, which is the L^2 distance, or the Manhattan distance (the L^1 distance), or even the L^∞ distance, which correspond to $max_{1 \leq i}(x_i - y_i)$

Remark The greater p is, the more emphasis the biggest difference will have. If we want to put an emphasis ourselves, we can modify the L^p distance in the following way for instance:

$$We\ define : (\alpha_i)_{i\leq n}\ such\ as : \sum_{k=1}^{n} \alpha_k = 1\ and : \forall i \leq n, 0 < \alpha_i < 1$$

$$We\ have : d(x,y) = (\sum_{k=1}^{n} \alpha_k |x_k - y_k|^p)^{1/p}$$

We can see that there are some other type of geometrical similarities which aren't being taken into account. For instance, with those distance we can only define ball around the data points, but the similarity of having a similar orientation in a polar space. We will discuss this further in the next section.

1.3 Similarities which can't be expressed by distances

Definition $(Pearson's\ correlation)$ This type of similarity can express the similarity of angles in a polar space, but it is morally linked to probabilistic correlation in higher dimensions. We define it as below:

$$\overline{x} = \frac{1}{n}\sum_{k=1}^{n} x_k$$

$$\overline{y} = \frac{1}{n}\sum_{k=1}^{n} y_k$$

$$\rho(x,y) = \frac{\sum_{k=1}^{n}(x_k - \overline{x})(y_k - \overline{y})}{\sqrt{\sum_{k=1}^{n}(x_k - \overline{x})^2}\sqrt{\sum_{k=1}^{n}(y_k - \overline{y})^2}}$$

Geometrical interpretation This correlation is equal to the cosine of the angle between the vector x and the vector y, given the data points belong to R^n

1.4 Categorical features

Construction To represent features, we will consider the list of presence/absence of that particular feature. Most of the time this will lead to a binary representation, but if the feature is the shape, we can code shapes by integers instead of zeros and ones.

Definition $(Hamming\ distance)$ It is the number of bits that are different in the representation:

$$d_{Hamming}(x,y) = \sum_{k=1}^{n} (x_k\ XOR\ y_k)$$

If the representation is binary this is equivalent to both the Manhattan and Euclidean distances.

Definition $(Tanimoto\ similarity)$ It is the number of shared features with normalization, and is expressed as following:

$$s_{Tanimoto}(x,y) = \frac{\sum_{k=1}^{n}(x_k\ AND\ y_k)}{\sum_{k=1}^{n}(x_k\ OR\ y_k)}$$

Definition $(MinMax\ similarity)$ It is the number of shared features with normalization in the case where the representations are not necessarily binary representations, and is expressed as following:

$$s_{MinMax}(x,y) = \frac{\sum_{k=1}^{n} min(x_k, y_k)}{\sum_{k=1}^{n} max(x_k, y_k)}$$

Remark In the case of binary representations the MinMax and Tanimoto similarities are equivalent.

2 The kNN algorithm

Now that we have learned how to evaluate the similarity of two items, we are able to find the k nearest neighbors of an item. In this section, we will see how to define the decision boundaries through Voronoi tesselation, and then some practical matters of the kNN algorithm.

2.1 Voronoi tesselation

When $k = 1$, predicted classes are quite easy to visualize. We use the concept of *Voronoi tesselation* to define them.

Definition. The *Voronoi cell* of $x \in \mathbb{R}^p$ is the set which contains all the points of $\mathbb{R}^p$ which are closer to x than to any other point of the training set. Formally, the Voronoi cell of x equals: $\{u \in \mathbb{R}^p : \forall x_i \in \mathcal{D}, d(x, u) \leq d(x_i, u)\}$.

For instance, in $\mathbb{R}^2$, the Voronoi cells are approximately drawn on the following figure.

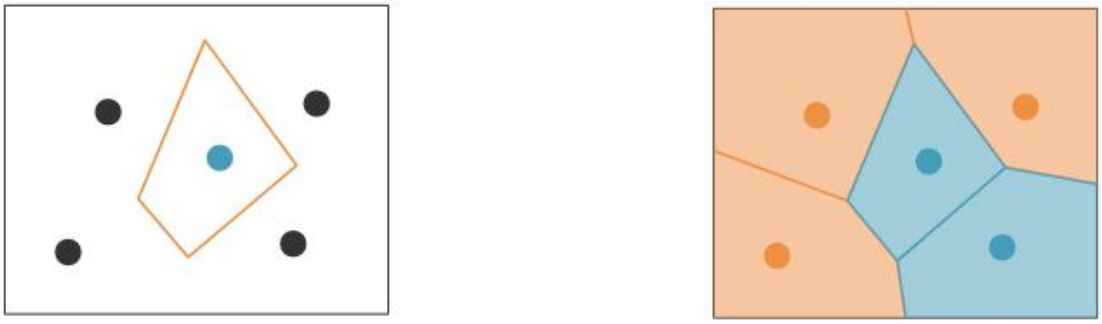

Figure 2: Left: a single Voronoi cell. Right: Illustration of the partition of space thanks to Voronoi cells, in case of binary classification (yellow and blue).

One can guess that these Voronoi cells have the bisector hyperplanes as boundaries.

For the 1-NN algorithm, the predicted classes are simply the union of the Voronoi cells surrounding the training points which belong to them.

As concerns the kNN method with $k > 1$, the predicted classes are less intuitive. Nonetheless, the following property of the Voronoi cells is shared: they also partition the space (in a more complex way though).

2.2 Advantages and drawbacks of the kNN method

2.2.1 Advantages

First, training is very fast. Actually, there is virtually nothing to do, since one only has to store the training set. However, if one wants to speed up the testing process, a relevant indexing procedure could be applied to get the nearest neighbors more rapidly than by computing all the distances each time.

The kNN method is also hardly disturbed by noisy data, since if k is large enough, the prediction is based upon a mean. Furthermore, kNN can learn complex functions.

Another practical aspect is the fact that the kNN method keeps the training data, which can be useful for collective projects and collaborative work.

2.2.2 Drawbacks

kNN may require a lot of memory, since it has to store many distances to find the nearest neighbors.

At variance with training, prediction can be very slow, since the complexity of labeling one new data point is $O(np + n \log k)$.

Finally, albeit resistant to noise, kNN is not efficient when irrelevant features are involved. Indeed, it computes distances using these useless features and is fooled by them.

2.3 The curse of dimensionality

One major issue of the kNN method is the problems which occur when the dimension p of the space gets high. Indeed, points are all the more distant as p grows, which is very counter-intuitive. Let us explain this by an example.

With $p = 2$, the fraction of the points of a square that are outside of the circle inscribed in it is $1 - \frac{\pi}{4}$, which seems reasonable... Yet, for $p = 3$, this fraction increases, and generally speaking, the proportion of volume outside of a p-sphere inscribed in a hypercube gets closer to 1 as p grows!

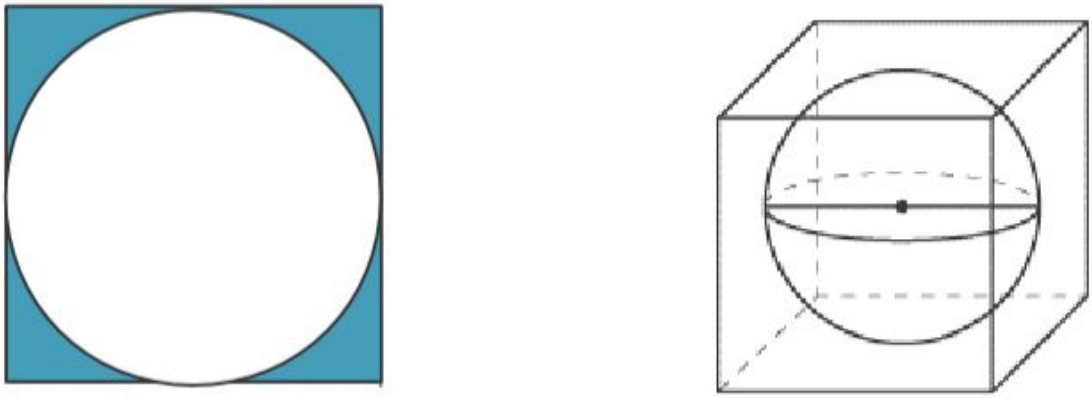

Figure 3: Illustration of the curse of dimensionality: hypersphere inscribed in a hypercube for dimensions $p = 2$ and $p = 3$.

What it all boils down to is that hyperspace is very big and stretches distances, so that dimension reduction is needed (c.f. chapter 11).

2.4 Variants of kNN algorithm

2.4.1 ϵ-ball neighbors

Instead of fixing the number of neighbors, one can fix the distance of the neighbors that will be compared to the to-be-predicted item. The main flaw of this method is that ϵ has to be big enough so that no item has no neighbor.

2.4.2 Weighted kNN

An intuitive alternative to classic kNN can be to add coefficients to the vote of the neighbors as a decreasing function of their distance to the test point. One may use an exponential function for instance. A theory to find optimal weights is developed in Swamidass, Azencott et al. 2009, *Influence Relevance Vote*.

Summary

— The kNN method has an extremely simple training, but prediction may be very slow.
— It relies on a good choice of similarity function.
— The decision boundaries are shaped by Voronoi tesselation for 1-NN method.
— Notwithstanding, the algorithm is less efficient as dimensionality grows.