Algorithmic Learning Theory
Spring 2017
Lecture 3

Instructor: Farid Alizadeh
Scribe: Atharv Bhosekar
Edit: Yuan Qu

02/01/2017

This lecture introduces k-Nearest Neighbor (k-NN) method and discusses the following:

1. Overview

2. Application Methods

    (a) For Bayes

    (b) For Regression

3. Properties

    (a) Non-parametric

    (b) Distance

        i. Eucliden distance

        ii. Mahatten distance

        iii. Minkowski distance

        iv. Hamming distance

        v. Voronoi diagram

        vi. k-d tree
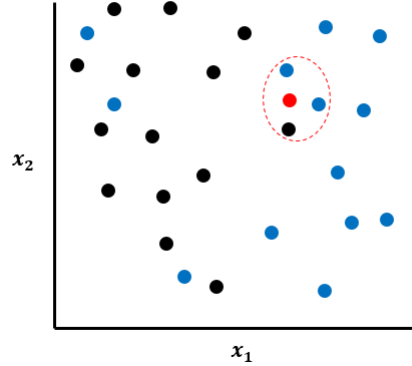
    (c) Standardization

4. Validation (in Lecture 4)

Figure 1: Example of a 2 feature 2 class classification problem. Red point shows the point to be classified. Points in the red dotted circle denote 3 nearest neighbors.

1   Overview

k-NN as the name suggests, takes k nearest points in the available data set to the desired new point, averages the responses for those points and reports that as the response at the new point. To better visualize this, consider an example of a classification problem with two features denoted by x_1 and x_2 and two classes denoted by blue and black color as shown in Figure 1. The problem is to classify the new point (the point denoted in red color) as blue or black. For illustration, consider $k = 3$ i.e. consider three nearest neighbors to the new point. Three points shown in the circle are found to be closest to the new point and two of these neighboring points are blue and one is black. Taking average of three points, the new point is 2/3rd likely to be blue and 1/3rd likely to be black. Therefore,the k-NN algorithm classifies the new point as blue. One small point to notice here is that the class reported based on k-NN is the class that has maximum value and does not take in to consideration the magnitude by which that class is favored over the other class.

Relating this concept to the Bayes' decision rule, with a vector of features $\underline{x} = (x_1, x_2, \ldots, x_p)$ where p is the number of features. The problem is to find $Y_{new} = \underset{y}{\operatorname{argmax}} \Pr[Y|\underline{x}_{new}]$. To put it in words, given a new point x the problem is to find the probability that it belongs to class black or blue or any other class name relevant to the problem. The problem however is that we have no information about the probability. k-NN solves this problem by implicitly making two assumptions. First, it replaces x_{new} with k- neighbors of x_{new}. Second, it approximates probability of the entire population by sample proportion which is said to be approximately equal to the proportion in the

neighborhood of each class.

2   k-NN for regression

Similar to classification, k-NN for regression is analogous to Bayes' decision rule for regression. Looking back at classification, the classification rule is obtained by minimizing loss function that quantifies misclassification. In case of regression, usually the loss function is defined as $\text{loss}(x, \hat{f}|f) = (f(x) - \hat{f}(x))^2$ which is sum of squared errors. Our aim is to minimize the expected loss given by $\text{risk}(\hat{f}|f) = \mathbb{E}_x(f(x) - \hat{f}(x))^2$. Assuming we have the entire population, best $\hat{f}$ is given by $\hat{f} = \mathbb{E}_x(y)$, which is the average of all possible values of y. Two assumptions that k-NN makes implicitly for the case of classification are made even for the case of regression. For example, x_{new} is replaced by k closest data points x_i and average of all y responses at x_{new} are approximated by responses at k nearest neighbors.

3   Key features of k-NN

k-NN is a purely non-parametric method. This can be derived from the observation that the loss function in case of both classification as well as regression does not make any assumption on the nature of surrogate function $\hat{f}$.

3.1   Notion of nearness

Entire k-NN method is dependent on finding and utilizing k-nearest neighbors. However, there are some issues that one needs to take in to account while defining nearness.

It is important to note that the operation on the responses of k nearest neighbors to obtain the response at a new point is dependent on the loss function. For example, average is used only if the loss function is squared error, median will be used if the loss function is sum of absolute values and so on.

3.1.1   Definition of distance

First, notion of nearness is most of the times problem dependent based on the definition of distance. There are number of different ways to define distance between two distinct data points. The most common distance used for numerical problems (classification and regression involving numbers) is Euclidean distance which is defined as $\|\underline{a} - \underline{b}\| = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \ldots + (a_p - b_p)^2}$. There are some other types of distances that one might encounter. For example, l_1 distance $= \|\underline{a} - \underline{b}\|_1 = \|a_1 - b_1\| + \ldots + \|a_p - b_p\|$. In general, l_r distance is given by $[(a_1 - b_1)^r + \ldots + (a_p - b_p)^r]^{1/r}$. Two more special cases of this general form that are used often are $l_\infty = \max|a_i - b_i|$ and l_0 that signifies number of items that are not equal (or classified incorrectly). If individual features are not useful then one can also define distance as $|X - Y| = 0$ if X and Y are same

from class and $|X - Y| = 1$ if X and Y are from different classes. If the problem is not numerical (say words) then an example distance metric can be hanning distance $|"a_1 a_2 \ldots a_n" - "b_1 b_2 \ldots b_n"|$.

3.1.2 Scale dependence

Second issue arises from the scale dependence of distance irrespective of the way we define distance. To illustrate this, a simple example of three points is considered. Given three data points as follows: $a = (45, \$76000)$, $b = (25, \$75000)$, $c = (42, \$78000)$. Distance between points a and b can be written as $d(a, b) = \|a - b\| = \sqrt{20^2 + 1000^2} \approx 1000$. Similarly, $d(b, c) \approx 3000$ and $d(a, c) \approx 2000$. Changing the scale of second co-ordinate by a factor of 1000, one can write the same distance as $d(a, b) = \|a - b\| = \sqrt{20^2 + 1^2} \approx 20$, $d(b, c) \approx 17$ and $d(a, c) \approx 3.6$. As it can be seen clearly, that by changing the scale, now c is closer to a than b unlike the previous case. Thus, scale in which data is reported can affect selection of nearest points and this makes outcome of k-NN scale dependent.

There are solutions to avoid issues with scale dependence such as standardizing the data and it is important to get a brief idea about how it can be helpful. Assuming that the data roughly follows a normal distribution we transform data $\underset{\sim}{x} \to X_1, X_2, \ldots, X_N$ to obtain a standardized data using $X_k = \frac{X_1 - \bar{X}}{s}$ where $\bar{X} = \frac{X_1 + X_2 + \ldots + X_N}{N}$ which is mean and $s = \frac{\sqrt{(X_1 - \bar{X})^2 + \ldots + (X_N - \bar{X})^2}}{N - 1}$ which is standard deviation. One point notice in case of this standardization is that this approach is effective only when features are completely independent of each other. If the features are correlated, one might end up overemphasizing some of the features and under-emphasizing some. To avoid this, use of a covariance matrix is suggested. Covariance matrix $\mathbb{S}$ is defined as $\mathbb{S}_{ij} = \sum_l (x_{il} - \bar{x}_i)(x_{jl} - \bar{x}_j)$. This implies that $X_i \to (X_i - \bar{X}1)^\mathsf{T} \mathbb{S}^{-1} (X_i - \bar{X}1)$. Therefore, $d(a, b) = \sqrt{(a - b)^\mathsf{T} \mathbb{S}^{-1} (a - b)}$. To put it simply, if the correlation is large intuitively, $\mathbb{S}^{-1}$ becomes small and it weighs down the distance. As a result of these computations, evaluating $\hat{f}$ at a new point is expensive.

4 Voronoi diagrams

Voronoi diagrams use the principle of k-NN in the sense that it divides the space based on the nearest sample (or one nearest neighbor) of point. For example, if one has two data points in the space, the space is divided in to two halfspaces from the mid-point of those two data points. Similarly, k-level voronoi diagrams use the principle of k nearest neighbors. A sample Voronoi diagram is shown in Figure 2.
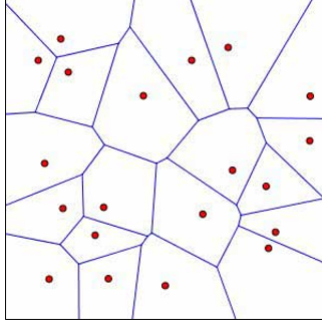
Figure 2: Voronoi diagram example

5 Computational complexity of k-NN

Using efficient data structures such as a binary search tree to find the nearest neighbors, it can be shown that the computational efforts involved are proportional to depth of the tree which is $\approx \log(N)$ where N is the number of data points or nodes stored in the tree. Extending this concept to multiple dimensions, one can use kd-tree. Given a balanced binary tree (starting from the median for example), it can be shown that the computational complexity is of the order $kp \log(N)$ where, k is the number of classes, p is the number of features and N is the number of data points.

   With this analysis, one can comment on the limitations and applicability of k-NN. The method is clearly effective if k and p is small compared to number of data points N. This makes it extremely effective for on-line learning. Additionally, it is clear that k-NN is not effective if number of features is large.

5.1 Issue with high number of features

Having high number of features is a problem in general and it is interesting to get a feel of this problem using a simple numerical experiment. Considering a one dimensional problem and the space is bound between 0 and 1. Let us generously assume that even 0.99 is *near* to 0. In other words, 99% of the space is close to 0. If we scale it up to 2 dimensions, this 'near' region becomes $0.99^2 \approx 0.98$. Extending it to just 1000 dimensions, this region becomes 4.3×10^{-5}. Which means in 1000 dimension one needs approximately 23000 points to get same 'density' of points as one would get with just 1 point in 1 dimension.