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) k-NN for Bayes
    (b) k-NN for regression
3. Properties
    (a) Non-parametric
    (b) Distance
    (c) Standardization
4. Validation (in Lecture 4)

Scribe: Atharv Bhosekar **Edit:** Yuan Qu
Lecture 3 **Date:** 02/01/2017

1 Overview

k-NN, a classification method, 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.

Supposed that we have a data set following:

y	x_1	x_2	...	x_p
y_1	x_{11}	x_{12}	...	x_{1p}
y_2	x_{21}	x_{22}	...	x_{2p}
...	...	...	...	...
y_n	x_{n1}	x_{n2}	...	x_{np}

The problem is to find the output y_{new} for a new input $x_{new} = (x_1, x_2, \cdots, x_p)$. k-NN method suggests that:

$$y_{new} = \frac{\sum y \text{ of } k \text{ nearest points}}{k}$$

$$\Rightarrow P[y_{new} = i] = \frac{\text{number of } y_j = i}{k}, j = 1, 2, \cdots, k$$

In two demension, if there are two classes, $y = 0$ and $y = 1$, so:

$$P[y = 0|x] = \frac{f_{x|y}(t|0)P[0]}{f_x(t)}$$

If x is in IID (Independent and Indentical distribution), we have:

$$f_{x|y}(t|0) \approx \frac{f(t_{[1]}|0) + \cdots + f(t_{[k]}|0)}{k}, t_{[i]} \text{ is the } i\text{th closest number to the } t$$

e.g. To better visualize this, consider an example of a classification problem with two features, x_1 and x_2, and two classes, blue and black, as shown in Figure 1.

The problem is to classify the new point (the red point) 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, 2 blue 1 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.

(some question about it) 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.

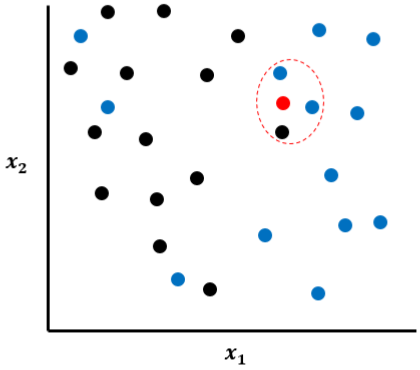

Figure 1: Example of a 2 feature 2 class classification problem. Red point shows the point to be classified. Points in the red dotted circle denote 3 nearest neighbors.

2 Application Methods

2.1 k-NN for Bayes

Considering Bayes' decision rule, with a vector of features $\underset{\sim}{x} = (x_1, x_2, \ldots, x_p)$, where p is the number of features. The problem is to find

$$y_{\text{new}} = \underset{y}{\operatorname{argmax}} P[y | \underset{\sim}{x}_{\text{new}}]$$

To put it in words, given a new point x, the problem is to find the probability that it belongs to the class relevant to the problem. But the probability may be uncomputable because of infinity of points.

k-NN solves this problem by making two assumptions:

1. Replace $\underset{\sim}{x}_{\text{new}}$ with k-neighbors of x_{new}.
2. Approximates $P \approx$ proportion of the neighborhood in each class.

2.2 k-NN for regression

Loss function :$\text{loss}(x, \hat{f} | f) = (f(x) - \hat{f}(x))^2 \longrightarrow$ Square error.

$$\text{risk}(\hat{f} | f) = E_x \cdot \text{loss}(x, \hat{f} | f) = E_x (f(x) - \hat{f}(x))^2$$

Assuming we have the entire population, best $\hat{f}$ is given by $\hat{f} = E_x(y)$, so we have:

$$\hat{f}(x_{\text{new}}) = \text{average of all } \{y \,|\, x = x_{\text{new}}\}$$

which maybe uncomputable because of infinity of points (need to get all $y \mid x = x_{\text{new}}$).

In this case, the error is caculated by square error, which the average leads to the minimum value, while median minimize the absolute error.

k-NN solves this problem by making two assumptions:

1. Replace x_{new} with closest k neighbors x_i of x_{new}.
2. The average of all y at x_{new} is approximated by y_i of the k-nearest neighbors.

If the loss function change to absolute function, the average should change to median.

3 Properties

3.1 Non-parametric

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 $\hat{f}$.

3.2 Distance

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.

If all x_i are numerical, then,

$$\text{Eucliden distance: } \|\underset{\sim}{a} - \underset{\sim}{b}\| = \sqrt{\sum_{i=1}^{p} (a_i - b_i)^2}$$

$$\text{Manhatten distance: } \|\underset{\sim}{a} - \underset{\sim}{b}\| = \sum_{i=1}^{p} |a_i - b_i|$$

$$\text{Minkowski distance: } l_r \text{ distance} \begin{cases} r = 0 & \text{Number of items} \\ r = 1 & \|\underset{\sim}{a} - \underset{\sim}{b}\| = \sum_{i=1}^{p} |a_i - b_i| \rightarrow \text{Manhatten distance} \\ r = 2 & \|\underset{\sim}{a} - \underset{\sim}{b}\| = \sqrt{\sum_{i=1}^{p} (a_i - b_i)^2} \rightarrow \text{Eucliden distance} \\ r & \|\underset{\sim}{a} - \underset{\sim}{b}\| = \sqrt[r]{\sum_{i=1}^{p} (a_i - b_i)^r} \\ r \rightarrow \infty & \|\underset{\sim}{a} - \underset{\sim}{b}\| = max\{|a_i - b_i|\} \end{cases}$$

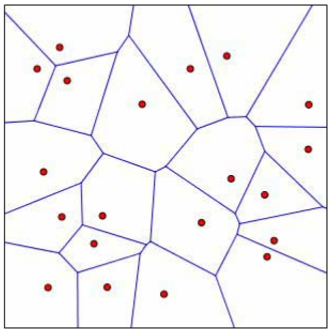

Figure 2: Voronoi diagram example

If all x is categorical, $x \to \{A, B, \cdots, L\}$ then,

$$\|x - y\| = \begin{cases} 0, & \text{if same class} \\ 1, & \text{otherwise} \end{cases}$$

Hamming distance: $|"a_1 a_2 \cdots a_m" - "b_1 b_2 \cdots b_m"|$

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 3.2.

The complexity of k-NN algorithm is same to Voronoi Diagram for k dimensions.

In 2 dimension, the case of Voronoi Diagram, 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 k-d 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.

Scribe: Atharv Bhosekar **Edit:** Yuan Qu
Lecture 3 **Date:** 02/01/2017

High dimension distance 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 4.3x10^{-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. So hard to find near points.

Generally, we keep the rule that: $N \gg 2^p$, $N \rightarrow$ Numbers of points, $P \rightarrow features$

3.3 Standardization

e.g. Scale dependence Given 3 data points, which describe the age and salary, as follows:

$a = (45, \$76000)$, $b = (25, \$75000)$, $c = (42, \$78000)$.

If we choose Eucliden distance as the measure, $d(a, b) = \|a{-}b\| = \sqrt{20^2 + 1000^2} \approx 1000$. Similarly, $d(b, c) \approx 3000$ and $d(a, c) \approx 2000$.

Changing the scale of salary by a factor of 1000, then the distance can be written in $d(a, b) = \|a - b\| = \sqrt{20^2 + 1^2} \approx 20$. Similarly, $d(b, c) \approx 17$ and $d(a, c) \approx 3.6$.

Thus, scale in which data is reported can affect selection of nearest points and this makes outcome of k-NN scale dependent. Assuming that the data roughly follows a normal distribution,

$$\underset{\sim}{x} \rightarrow x_1, x_2, \cdots, x_n$$

Standardize: $x_k = \frac{x_k - \bar{X}}{s}$

where $\bar{X} = \frac{\sum_{i=1}^{n} x_i}{n}$ is the mean, and $s = \frac{\sqrt{\sum_{i=1}^{n}(x_i - \bar{X})^2}}{n-1}$ is the 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

$$\mathbb{S}_{ij} = \sum_{l} (x_{il} - \bar{x_i})(x_{jl} - \bar{x_j})$$

$$x_i \rightarrow (x_i - \bar{X} \cdot \underset{\sim}{i})^T \mathbb{S}^{-1} (x_i - \bar{X} \cdot \underset{\sim}{i})$$

.

In which use $\bar{X} \cdot \underset{\sim}{i}$ to vectorize the $\bar{X}$.

Therefore, $d(a, b) = \sqrt{(a - b)^T \mathbb{S}^{-1} (a - b)}$.

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.