CS5063: Foundations of Machine Learning

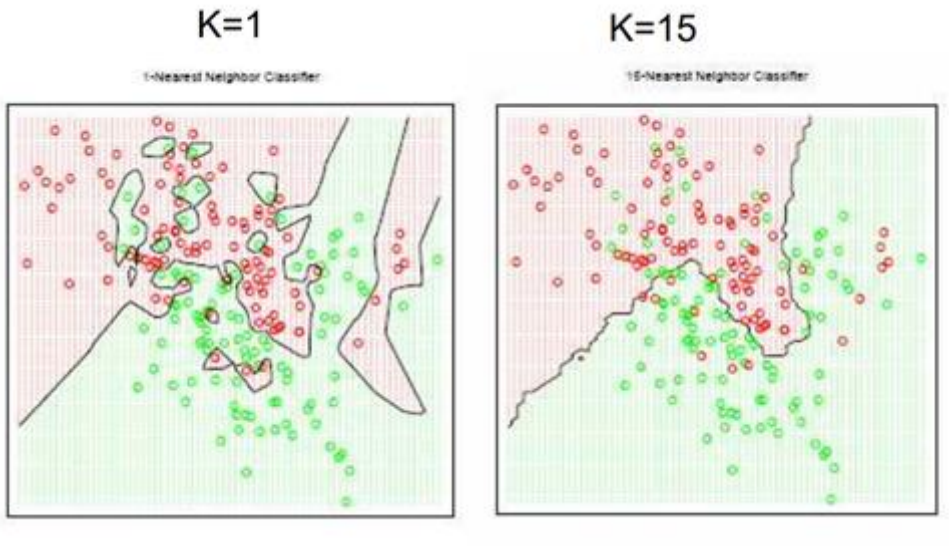
Assignment 1

Submitted By:
Vishal Singh Yadav – CS20MTECH01001

---

1. We have n points of two classes, each containing n/2 points overlapped to some extent in 2D space.

    a. For each new point, we find k nearest neighbours and return the majority votes of their labels, i.e., for K=1, we only compare to 1 nearest neighbour and give the new point the label of nearest one point. As K increases, we average over more and more points. Due to this, higher K leads to smoother predictions as we average over more data whereas. At K=1, we get piecewise constant labelling.



Source: https://www.cs.ubc.ca/~murphyk/Teaching/CS340-Fall07/L4_knn.pdf

Also, at K=1, we get zero training error but badly overfits the data, but as we increase K up to n, our training error initially increases and becomes constant at some value of $1 < K < n$.

b. Testing error at K = 1 is highest as each new point is given the label of the closest point. Since the new point is not generalised over large data and overfitted, it gives poor accuracy. As k increases, the testing error increases as the model is generalising more on the dataset. After a certain point, the test error will increase slowly as K gets too high and the model starts underfitting on the data.

c. When the dimension of input data is high, we get the curse of dimensionality. This affects kNN in two ways:

    i. Distance between points: In the kNN classifier, we assume that similar points share similar labels. In higher dimensions, the points drawn from a probability distribution tend not to be close together. This can be solved by getting more points, but the number of data points grow exponentially with d and hence is not feasible.

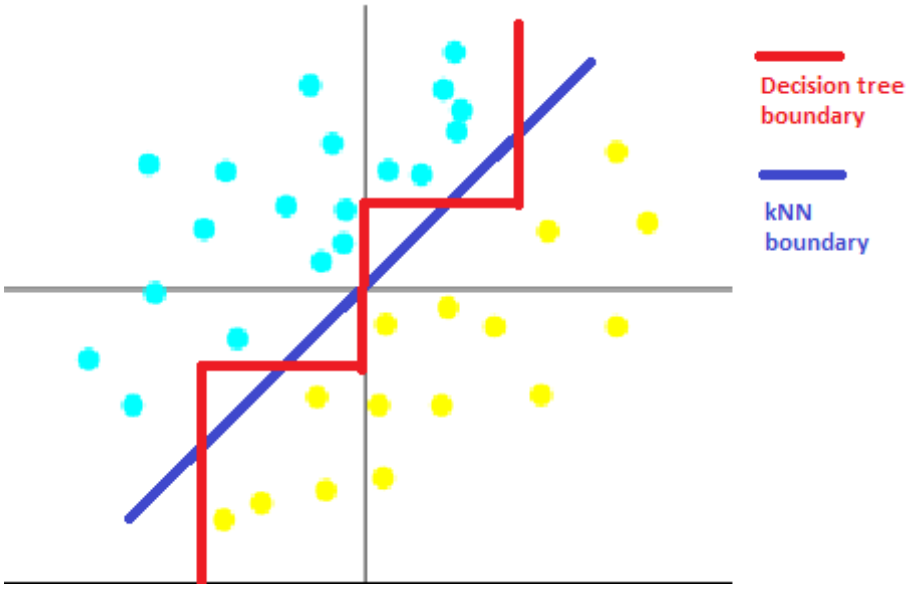
ii. Distances to hyperplanes: In d dimensions, d-1 dimensions will be orthogonal to the normal of any given hyperplane. As the distance between pairwise points large in higher dimensions, distances to hyperplanes become tiny. This results in most points being very close to these hyperplanes, and input can be easily perturbed to change classification outcome (adversarial samples).

d. Assume a trivial dataset with only two points (0,0) and (0,2), each belonging to a different class.

The decision boundary of a 1-NN will be a line bisecting the line joining the two points.

In the decision tree, we will find the decision boundary to be a line parallel to the x-axis (0,1).

In the case of 1-dimensional data with sufficiently separated classes, it is certainly possible to create a decision tree that gives a similar classification to 1NN.



If we increase the dimensions of data or the data has overlapped, the same will not hold. In such a case, a decision tree will not classify as 1NN as its decision boundaries are parallel to the x or y-axis. An overlapping dataset will not be divided into proper limits based on the x-axis or y-axis.

2.

a.

Classification probability is given by Bayes theorem.

$$P(y|x) = \frac{p(x|y) \cdot P(y)}{\sum_{y'} p(x|y') P(y')}$$

Gaussian likelihood is given by

$$p(x|c_j) = \frac{1}{\sqrt{2\pi\sigma_j^2}} \cdot e^{-\frac{1}{2}\left(\frac{x-\mu_j}{\sigma_j}\right)^2}$$

$\sigma_1^2 = 0.0149$

$\sigma_2^2 = 0.0092$

$\mu_j = \frac{1}{N_j} \sum_{i=1}^{N} x_i$

$\mu_1 = \frac{1}{10}\left(0.5 + 0.1 + 0.2 + 0.4 + 0.3 + 0.2 + 0.2 + 0.1 + 0.35 + 0.25\right)$

$= 0.26$

$\mu_2 = \frac{1}{4}\left(0.9 + 0.8 + 0.75 + 1.0\right)$

$= 0.8625$

Class probabilities are given by.

$$P(c_j) = \frac{N_j}{\sum_{k=1}^{n} N_k}$$

$$P(c_1) = \frac{10}{10+4} = \frac{10}{14} = 0.714285$$

$$P(c_2) = \frac{4}{10+4} = \frac{4}{14} = 0.285714$$

To find if $x = 0.6$ belongs to class 2, we have to find. $P(c_2 | 0.6)$

$$P(c_1|0.6) = \frac{P(0.6|c_1)P(c_1)}{P(0.6|c_2)P(c_2) + P(0.6|c_2)P(c_2)}$$

$$= \frac{\frac{1}{\sqrt{2\pi \times 0.0149}} e^{-\frac{1}{2}\left(\frac{0.6-0.26}{0.0149}\right)^2} \times 0.714285}{\left(\frac{1}{\sqrt{2\pi \times 0.0149}} e^{-\frac{1}{2}\left(\frac{0.6-0.26}{0.0149}\right)^2} \times 0.714285 + \frac{1}{\sqrt{2\pi \times 0.0092}} e^{-\frac{1}{2}\left(\frac{0.6-0.8625}{0.0092}\right)^2} \times 0.285714\right)}$$

$$= 0.6305$$

b. X = [1, 0, 0, 1, 1, 1, 1, 0]
Let A = attribute
P = Politics
S = Sport

$P(A|P) = 2/6 * 5/6 * 5/6 * 5/6 * 5/6 * 1/6 * 4/6 * 1/6 = 0.0029768709$
$P(A|S) = 4/6 * 2/6 * 5/6 * 4/6 * 1/6 * 1/6 * 0/6 * 5/6 = 0$
$P(A|P)\ P(P) = 0.0029768709 * 6/12 = 0.00148843545$
$P(A|S)\ P(S) = 0 * 6/12 = 0$

The maximum likelihood that X belongs is about politics is 0.00148843545.

3. *Code provided separately*

   a. I have implemented the ID3 algorithm to form the decision tree. I got an accuracy of 0.7689.

   b. In K-fold implementation, I have divided the dataset into 10 almost equal parts. Then I ran the training testing 10 times to simulate K-fold. In each run, the *i*th dataset is not included in the training set and instead is taken as the test set. In the end, the accuracy is averaged over 10 runs. I got an accuracy of 0.8152

      This increase in accuracy is due to more datasets. As we are using all the datasets, no part of the dataset is left. This allows the model to explore all the structures present in the dataset that are left out when K-fold is not used.

   c. Improvement strategies

      i. Using Gini index instead of Entropy – Using Gini index instead of Entropy increased our accuracy to 0.7832, a slight increase of 0.0143. This can be due to implementation error or floating-point calculations error as Entropy requires a lot of calculations. So we can consider the accuracy of Gini similar to that of Entropy. The only difference is that Gini is easier to implement and faster as computations are less than Entropy.

      ii. Decision tree pruning – We set the max depth of the tree to be 3. This increases our accuracy to 0.7853. This can be attributed to the fact that the tree can generalise well over the dataset. When no *max_depth* is set, the tree may be overfitting as we are using a greedy algorithm to construct the tree. Since the amount of data is not large, the tree may overfit the dataset when *max_depth* is higher than 3.