CS5590 – Foundation of Machine Learning
Assignment - 1

Tushar K Raysad (BM21MTECH14004)

1. k-NN:

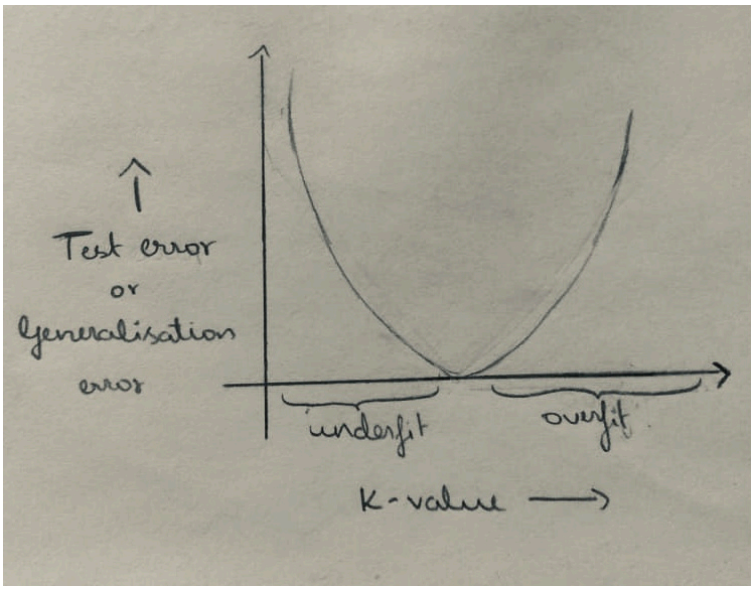
In k-nearest neighbours (k-NN), the classification is achieved by majority vote in the vicinity of data. Given n points, imagine two classes of data each of n/2 points, which are overlapped to some extent in a 2-dimensional space.

(a) Describe what happens to the training error (using all available data) when the neighbour size k varies from n to 1.

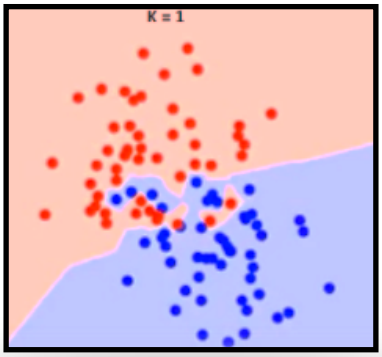
- When k=1, we won't get any training error. Because KNN algorithm will look for one of the nearest point in vicinity and hence there will be more sensitive to specificity of training data. However, if we increase the value of k=n, KNN algorithm look for more points in neighbourhood and hence less sensitive to data and increase the generalisability at the cost of variance.

(b) Predict and explain with a sketch how the generalisation error (e.g. holding out some data for testing) would change when k varies? Explain your reasoning.

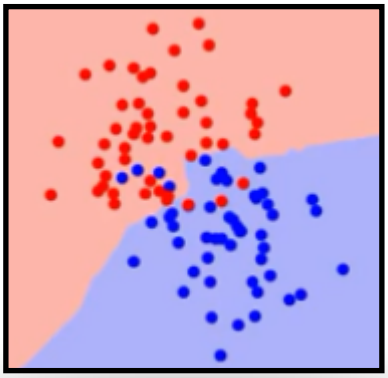
- The following figure shows the curve of Generalisation error rate v/s K value :



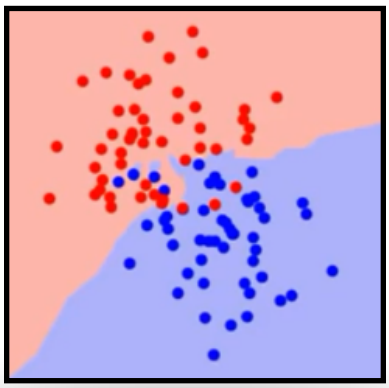
- At K=1, As we are overfitting the data the Generalisation error would be considerably high. Following figure demonstrates the classification of data when k=1,



- As the K value increases the noise in the data reduces which will in turn decrease the generalisation error and we can observe under-fitting of data at this value of k. The below figure shows the under-fitting of data when k=5,



- At a certain value of K the generalisation error will be at its minimal point as shown in the graph above. This point is considered as the optimal value of K for the classification of data. Therefore after this point we can observe the generalisation increasing with the K value. Following figure shows the classification of data when K = n where n is the optimal value of K.

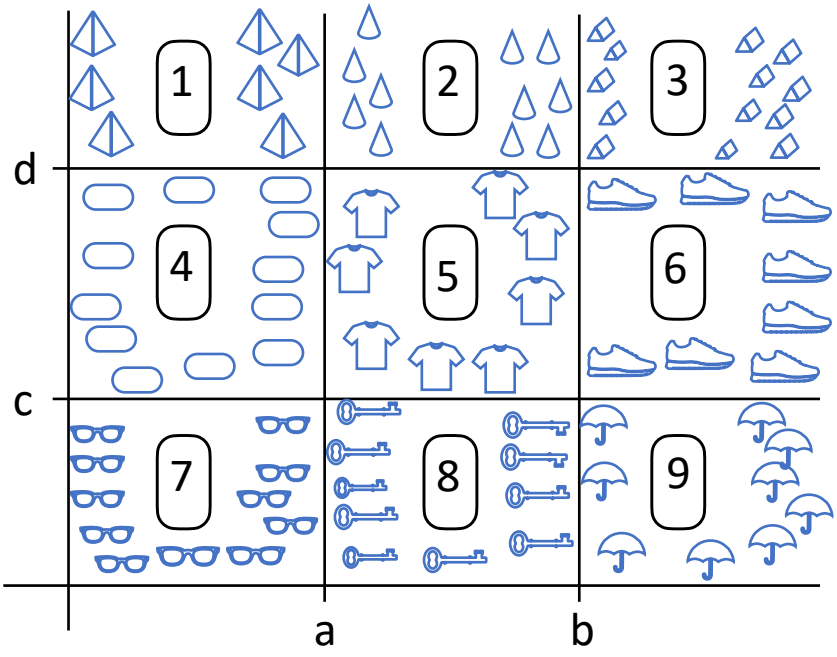


(c) Give two reasons (with sound justification) why k-NN may be undesirable when the input dimension is high.

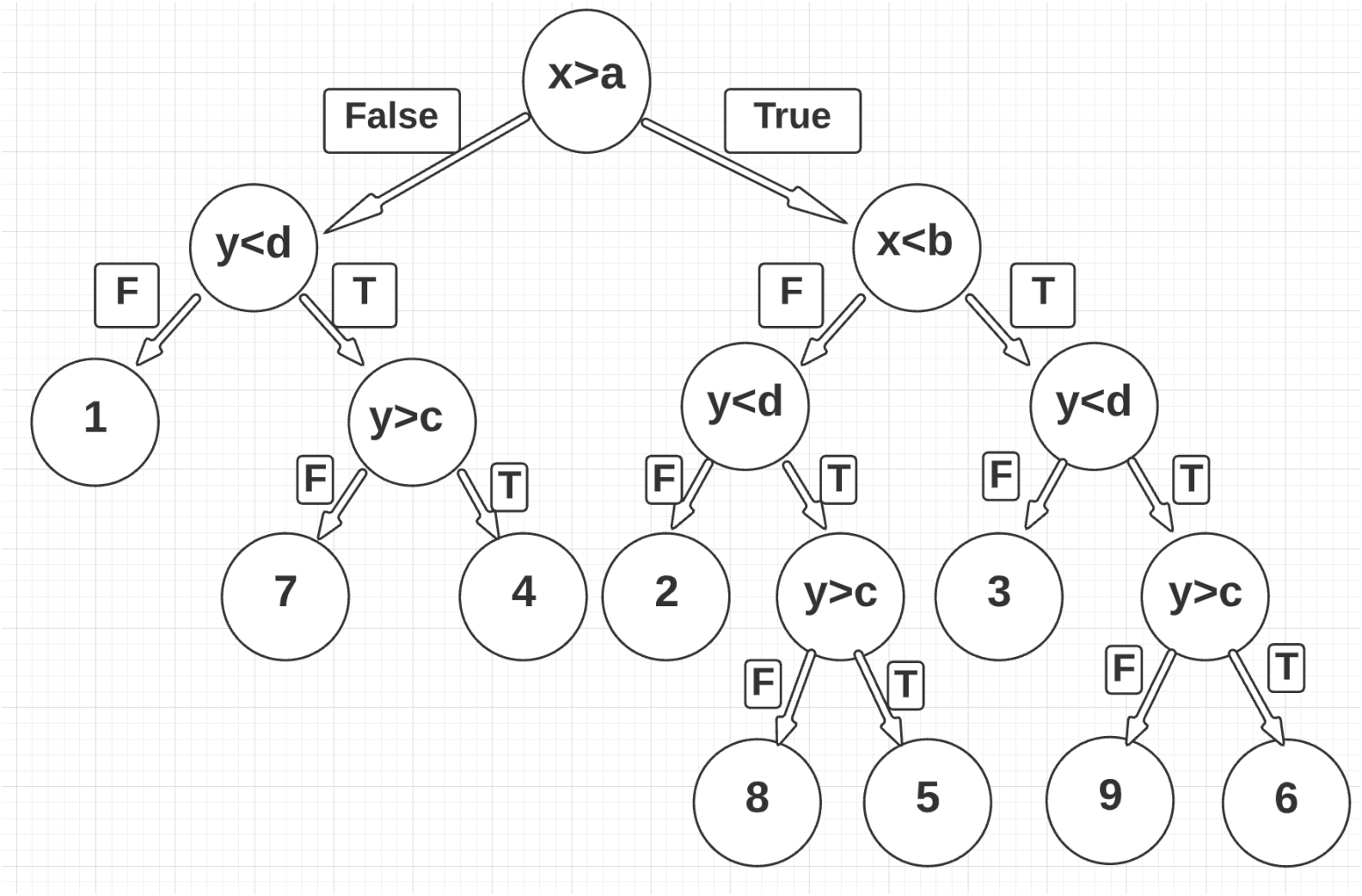
1. *K-NN is a slow algorithm*: K-NN might be very easy to implement but as dataset grows efficiency or speed of algorithm declines very fast.
2. *Curse of Dimensionality:* KNN works well with small number of input variables but as the numbers of variables grow K-NN algorithm struggles to predict the output of new data point.
3. *Optimal number of neighbours:* One of the biggest issues with K-NN is to choose the optimal number of neighbours to be consider while classifying the new data entry.
4. *Imbalance Data problems* : The k-NN doesn't perform well on imbalanced data. If we consider two classes , the majority of data will be labelled to one specified class and thus the preference will be given to that particular class only. This causes imbalance.

(d) Is it possible to build a univariate decision tree (with decisions at each node of the form "is x > a", "is x < b", "is y > c", or "is y < d" for any real constants a, b, c, d) which classifies exactly similar to a 1-NN using the Euclidean distance measure? If so, explain how. If not, explain why not.

- Yes, It is possible to build a univariate decision tree for the given conditions which classifies exactly similar to a 1-NN using the Euclidean distance measure. To justify, let us assume 9 different classes of data sets classified as shown below. And constant points a and b are on x axis and c and d are on y axis.



- Based on above diagram we can build a univariate binary split decision tree. Any point on the co-ordinate axis located on a particular class can be verified in decision tree shown below:



- In this case, x & y are features and a, b, c, d are constants or real values.
- Thus given any point with attributes (x, y) for 1-NN classification or the above decision tree , result will be same.

2. Bayes Classifier:

(a) A training set consists of one-dimensional examples from two classes. The training examples from class 1 are {0.5, 0.1, 0.2, 0.4, 0.3, 0.2, 0.2, 0.1, 0.35, 0.25} and from class 2 are {0.9, 0.8, 0.75, 1.0}. Fit a (one dimensional) Gaussian using Maximum Likelihood to each of these two classes. You can assume that the variance for class 1 is 0.0149, and the variance for class 2 is 0.0092. Also estimate the class probabilities p1 and p2 using Maximum Likelihood. What is the probability that the test point x = 0.6 belongs to class 1?

Given :- $\sigma_1^2 = 0.0149$

$\sigma_2^2 = 0.0092$.

x_i for class 1 = {0.5, 0.1, 0.2, 0.4, 0.3, 0.2, 0.2, 0.1, 0.35, 0.25}

x_i for class 2 = {0.9, 0.8, 0.75, 1.0}

N_j for class 1 = 10 , N_j for class 2 = 4.

(i) Therefore class probabilities,

$$P_1 = \frac{N_j \text{ for class 1}}{\sum N_j} = \frac{10}{10+4} = \frac{10}{14}$$

$$\boxed{P_1 = 0.714}$$

$$P_2 = \frac{N_j \text{ for class 2}}{\sum N_j} = \frac{4}{10+4} = \frac{4}{14}$$

$$\boxed{P_2 = 0.2857}.$$

(ii) To find :- $P(C_1/x=0.6) = ?$

We have, $$P(C_j/x) = \frac{P(x/C_j)\,P(C_j)}{P(x)}.$$

$$P(C_j/x) = \frac{P(x/C_j)\,P(C_j)}{\sum P(x/C_j)\,P(C_j)}$$

But, to get $P(C_1/x)$ we should find

$P(x/C_1)$ & $P(x/C_2)$.

To find $P(x/C_j)$:-

Using the Guassian Maximum Likelihood,

$$P(x/C_j) = \frac{1}{\sqrt{2\pi\sigma_j^2}} e^{-\frac{1}{2}\left(\frac{x-\mu_j}{\sigma_j}\right)^2}$$

we have to find μ_1, To find μ_1 we have

$$\mu_1 = \frac{1}{N_j}\sum x_i = \frac{1}{10}[0.5+0.1+0.2+0.4+0.3+0.2+0.2+0.1+0.35+0.25]$$

we get $\mu_1 = 0.26$

Similarly, $\mu_2 = \frac{1}{4}[0.9+0.8+0.75+1.0]$

$$\mu_2 = 0.8625.$$

Therefore,

$$P(x/C_1) = \frac{1}{\sqrt{2.\pi.0.0149}} e^{-\frac{1}{2}\left[\frac{0.6-0.26}{\sqrt{0.0149}}\right]^2}$$

$$P(x/C_1) = 0.0675 \cdot e^{-\frac{1}{2}\left[\frac{0.6-0.8625}{\sqrt{0.0092}}\right]^2}$$

$$P(x/C_2) = \frac{1}{\sqrt{2.\pi.0.0092}}$$

$$P(x/C_2) = 0.09831$$

Therefore,

$$P(C_1/x=0.6) = \frac{0.0675 \cdot 0.714}{0.0675.(0.714)+0.09831.(0.2857)}$$

$$P[C_1/x=0.6) = 0.6306$$

(b) You are now going to make a text classifier. To begin with, you attempt to classify documents as either sport or politics. You decide to represent each document as a (row) vector of attributes describing the presence or absence of the following words. x = (goal, football, golf, defence, offence, wicket, office, strategy) Training data from sport documents and from politics documents is represented below in a matrix in which each row represents the 8 attributes.

x politics =

$$\begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

x sport =

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Using a maximum likelihood naive Bayes classifier, what is the probability that the document x = (1, 0, 0, 1, 1, 1, 1, 0) is about politics?

To find :- $P((1,0,0,1,1,1,1,0)/\text{Politics}) = ?$

We have,

$$P((1,0,0,1,1,1,1,0)/\text{Politics}) = \frac{P(\text{Politics}/(1,0,0,1,1,1,1,0)) \cdot P(\text{Politics})}{P(1).P(0).P(0).P(1).P(1).P(1).P(1).P(0).}$$

We know that, $P(\text{Politics}) = \frac{1}{2}$, $P(0) = \frac{1}{2}$, $P(1) = \frac{1}{2}$

We Should find $P(\text{Politics}/(1,0,0,1,1,1,1,0))$.

Therefore,

$P(\text{Politics}/(1,0,0,1,1,1,1,0))$

$= P(\text{goal}/\text{Politics}).P(\text{football}/\text{Politics}).P(\text{golf}/\text{Politics}).P(\text{defence}/\text{Politics}).$
$P(\text{offecce}/\text{Politics}).P(\text{wicket}/\text{Politics}).P(\text{office}/\text{Politics}).P(\text{strategy}/\text{Politics})$

To find all these unknow probabilities, we should look for how many times 1 or 0 has occurred in 6 sets of training data for 8 attributes in the given matrix Xpolitics.

Therefore,

$P(\text{goal}/\text{Politics}) = \frac{2}{6}$ (1 has occurred 2 times out of 6 sets of training data).

$P(\text{football}/\text{Politics}) = \frac{5}{6}$ (0 has occurred 5 times out of 6).

$P(\text{golf}/\text{Politics}) = \frac{5}{6}$ (0 has occurred 5 times out of 6).

$P(\text{defence}/\text{Politics}) = \frac{5}{6}$ (1 has occurred 5 times out of 6).

$P(\text{offence}/\text{Politics}) = \frac{5}{6}$ (1 has occurred 5 times out of 6).

$P(\text{wicket}/\text{Politics}) = 1/6$ (1 has occurred 1 time out of 6).

$P(\text{Office}/\text{Politics}) = \frac{4}{6}$ (1 has occurred 4 times out of 6)

$P(\text{Strategy}/\text{Politics}) = \frac{1}{6}$ (0 has occurred one time out of 6).

$\therefore P(\text{Politics}/(1,0,0,1,1,1,1,0)) = \frac{2}{6} \times \frac{5}{6} \times \frac{5}{6} \times \frac{5}{6} \times \frac{5}{6} \times \frac{1}{6} \times \frac{4}{6} \times \frac{1}{6}$

$= \frac{5000}{(6)^8}$

So,

$P((1,0,0,1,1,1,1,0)/\text{Politics}) = \frac{\frac{5000}{(6)^8} \times \frac{1}{2}}{(1/2)^8}$

$\therefore P((1,0,0,1,1,1,1,0)/\text{Politics}) = 0.381056$

3. Decision Trees:

In this question, you will use the Wine dataset1 , a popular dataset to evaluate classification algorithms. The classification task is to determine, based on various parameters, whether a wine quality is over 7. The dataset has already been preprocessed to convert this into a binary classification problem (scores less than 7 belong to the "zero" class, and scores greater than or equal to 7 belong to the "one" class). Each line describes a wine, using 12 columns: the first 11 describe the wine's characteristics (details), and the last column is a ground truth label for the quality of the wine (0/1). You must not use the last column as an input feature when you classify the data.

(a) Decision Tree Implementation: Implement your own version of the decision tree using binary univariate split, entropy and information gain.

(b) Cross-Validation: Evaluate your decision tree using 10-fold cross validation. Please see lecture slides for details. In a nutshell, you will first make a split of the provided data into 10 parts. Then hold out 1 part as the test set and use the remaining 9 parts for training. Train your decision tree using the training set and use the trained decision tree to classify entries in the test set. Repeat this process for all 10 parts, so that each

entry will be used as the test set exactly once. To get the final accuracy value, take the average of the 10 folds' accuracies. With correct implementation of both parts (decision tree and cross validation), your classification accuracy should be around 0.78 or higher.

(c) *Improvement Strategies: Now, try and improve your decision tree algorithm. Some things you could do include (not exhaustive):*

- *Use Gini index instead of entropy*
- *Use multi-way split (instead of binary split)*
- *Use multivariate split (instead of univariate)*
- *Prune the tree after splitting for better generalisation*

Report your performance as an outcome of ANY TWO improved strategies.

Deliverables:

- *Code*
- *Brief report (PDF) with:*

(i) Accuracy of your initial implementation;

(ii) Accuracy of your improved implementation, along with your explanation of why the accuracy improved with this change.

(i) ***A)*** **Accuracy of the decision tree using binary univariate split, entropy and information gain is 83.4356**

B) Accuracy of the decision tree and cross validation is 84.3200.

(ii) Accuracy of the improved implementation are as follows:

1. Improved accuracy of the decision tree after using Gini index is 85.6851.

Accuracy is improved because of the following reasons :

- The Gini Index allows the larger distributions that are easy to implement but the Information Gain favours smaller distributions having small count with various specific values.
- The Gini Index is used by CART algorithms, while Information Gain is used in ID3 , C4.5 and other various other algorithms.
- The Gini index handles the categorical variables in terms of "success" or "failure" and gives only binary split, in contrast to that Information Gain computes the difference between entropy before and after the split and indicates the impurity in classes of elements.

2. Improved accuracy of the decision tree after using Multivariate split instead of Univariate split is 85.4806.

Accuracy of the decision tree improved because of the following reasons :

- In a multivariate decision tree, each test can be based on one or more of the input features. This makes the tree less complex. Thus the evaluation becomes easy and also it avoids replication problems that occur in univariate trees. Hence multivariate trees can have improved accuracy than univariate trees.

Note : The Python code is attached with this pdf as ".ipynb" file.