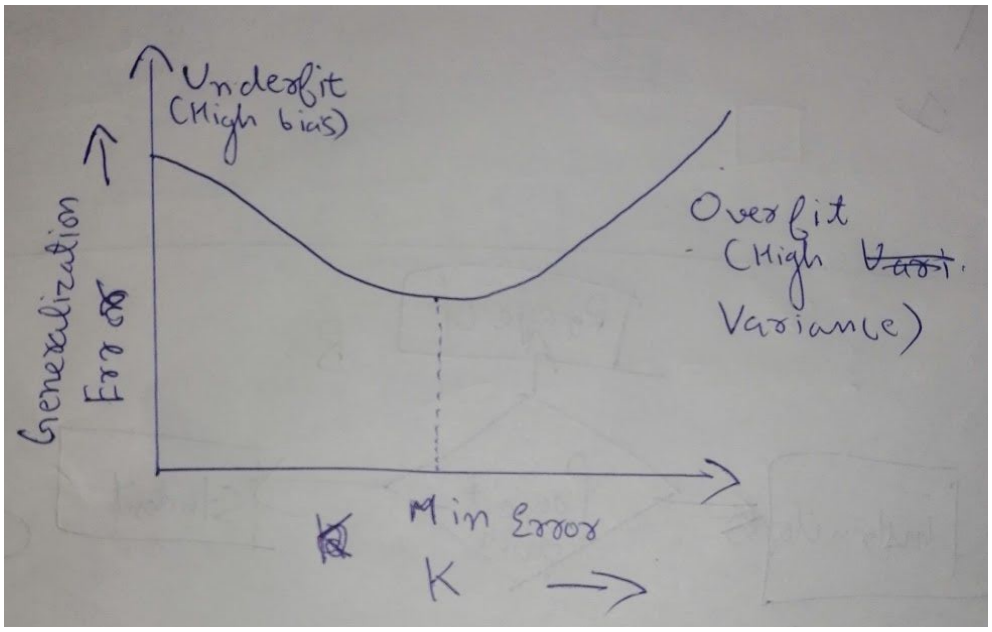
Devansh Agarwal
ES16BTECH11009

A1) a) Training error is the error when the training data is used as the testing data. For k = 1 the training error is equal to zero as each point is compared to itself and it is the nearest neighbour. For k = n, the k - neighbours will have n/2 elements of each class, the error will be maximum. The training error increases as we move from k = 1 to k = n.

A1) b) Generalization error 1st decreases as we move from k = 1 to k = n where it reaches a minima then it starts increasing till we reach to k = n. This is because when k is small the model has low complexity and it is underfit(high bias) when k is small even noise data hinders the prediction. When k is large it has high complexity and it is overfit(high variance) so the model doesn't generalizes well in both the cases. The graph shows the general pattern but in reality it might not be smooth due to the variation in the data sets.(heuristically k = sqrt(n) is good)

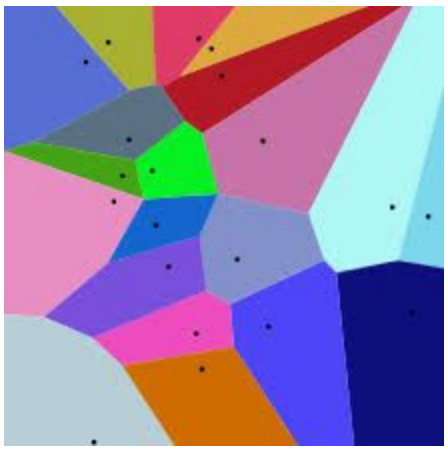


A1)c) 1. In Knn a distance metric needs to be calculated between test data point and every other point which is computationally expensive in high dimensional data set thus making it infeasible. (Although using a Quantum Computer this can be done in constant time complexity but since they are not available easily it does not solve the problem)

2. While dealing with high dimensional data a lot of irrelevant features might be introduced which hinder with the distance that is calculated and make the distance metric not representative of the actual closeness between different classes.

3. In KNN all the data needs to be stored and with high dimension data the size is even more so space complexity is huge.

A1)d) No, it is not possible to build a univariate decision tree which classifies exactly similar to a 1-NN.



The decision boundaries correspond to the cell boundaries of each point in 1-NN(voronoi diagram). So the boundaries are not always parallel to the coordinate axis. In a univariate decision tree the boundaries are parallel to the coordinate axis depending on the variable. To approximate the gradient using a univariate decision tree it will take an arbitrary number of decisions, hence it is not possible.

A2)

A2 a) $\mu_1 = \frac{0.5+0.1+0.2+0.4+0.3+0.2+0.2+0.1+0.35+0.25}{10}$

$= 0.26$

$\mu_2 = \frac{0.9+0.8+0.75+1.0}{4} = 0.8625$

$\sigma_1^2 = 0.149 \quad \sigma_2^2 = 0.0092$

Class 1 Probability $P_1 = \frac{10}{14} = 0.7143$

Class 2 Probability $P_2 = \frac{4}{14} = 0.2857$

$x = 0.6$

$$P(x/C_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{-\frac{1}{2}\frac{(x-\mu_k)^2}{\sigma^2}}$$

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

$$= 0.06756$$

$$P(x/C_2) = \frac{1}{\sqrt{2\pi 0.0092}} \exp\left[-\frac{1}{2}\frac{(0.6-0.8625)^2}{0.0092}\right]$$

$$= 0.09834$$

$$P(C_1/x) = \frac{p(x|C_1)\,p(C_1)}{p(x|C_1)\,p(C_1) + p(x|C_2)\,p(C_2)}$$

$$= 0.6305$$

A2 b) In x_{sport}; attribute for above all the values are 0.

$P(sport) = \frac{1}{2}$; $P(Politics) = \frac{1}{2}$

To solve 'zero-frequency-problem' for Bayesian setting is to add 1 to the count of every attribute value class combination.

$$P(Politics/x) = \frac{P(x/Politics).\,P(Politics)}{P(x/Politics)\,P(Politics) + P(x/sports)\,P(sports)}$$

$$= \frac{\frac{3}{8} \times \frac{3}{8} \times \frac{6}{8} \times \frac{6}{8} \times \frac{6}{8} \times \frac{6}{8} \times \frac{5}{8} \times \frac{2}{8} \times \frac{2}{8} \times \frac{1}{2}}{\frac{3}{8} \times \frac{6}{8} \times \frac{6}{8} \times \frac{6}{8} \times \frac{5}{8} \times \frac{2}{8} \times \frac{2}{8} \times \frac{1}{2} + \frac{5}{8} \times \frac{3}{8} \times \frac{6}{8} \times \frac{5}{8} \times \frac{1}{8} \times \frac{6}{8} \times \frac{2}{8} \times \frac{1}{2}}$$

$$= 0.878$$

A3 a) The decision tree is implemented using a dictionary tree based binary tree and the split value for a particular node is calculated by using the mean of each attribute.
A3 b) Implemented Cross Validation
A3 c) Accuracy before improvement : 0.8346

Accuracy after gini Index usage : 0.8346 The accuracy did not change
Laura Elena Raileanu and Kilian Stoffel compared both Gini Index and Entropy in "Theoretical comparison between the gini index and information gain criteria". They mentioned that the swap only mattered in 2% of the cases they tested so it is not surprising that it didn't change.

Accuracy after pruning:0.8361 The accuracy increased slightly here because the overfitting is reduced. The pruned tree had approximately 150 nodes lesser than unpruned one. I implemented pruning by increasing the entropy for the leaf nodes, earlier the leaf nodes had entropy 0, but here they have entropy < 0.18.

A4) I have used sklearn to implement this part. Top to accuracy scores that I got are 1)0.71148(Bernoulli Naive Bayes(BNB)) 2)0.61866(Decision Tree).
For processing data: I counted all the unique ingredients and for each point i made a column vector such that it has a 1 at a particular index if the corresponding ingredient is present otherwise 0. I also made a similar mapping between the class cuisine and an integer between 0 and 19.

From sklearn documentation: BNB implements the naive Bayes training and classification algorithms for data that is distributed according to multivariate Bernoulli distributions.Decision rule based on:

$$P(x_i \mid y) = P(i \mid y)x_i + (1 - P(i \mid y))(1 - x_i)$$

Each attribute is either 0 or 1.
From sklearn documentation:
Given training vectors and a label vector ,a decision tree recursively partitions the space such that the samples with the same labels are grouped together.

I also tried KNN which gave a score of 0.51277. It has a low score as the data is high dimensional and the matrix is sparse so 2 non related data points may have a small distance, thus giving a smaller accuracy.

Decision tree worked better than KNN as decision tree is not hindered by high dimensional data but it performed worse than BNB as the matrix is sparse and so the information gain is very less at each split and there would be overfitting also.

BNB was better than decision tree as it is not hindered by the sparsity of the matrix and BNB strongly penalizes the absence of a feature so cuisines like American and Indian that have not many common missing ingredients are classified better than decision tree. It doesn't have even higher accuracy because cuisines like Japanese, Chinese and Korean have more common missing ingredients than a Western cuisine and an Eastern one, so they are classified with a lesser accuracy.

A possible improvement can be 1st classifying the cuisines into 4 categories based on geographical locations and then individually classifying cuisines in each location.