FoML assignment I : KNN, Decision Trees, Model Selection, Naive Bayes classifier

Tamal Mondal

September 24, 2021

1 K nearest neighbour question

1.1 KNN training error when K varies

Given that we have n data points with 2 class labels and exactly $n/2$ data points belong to one of the class. As we are using all n data samples during training, when $K = n$ then we will have the maximum error. when we reduces K, the error rate will reduce and finally when $K = 1$, error rate will be 0(zero) as every point will be it's own neighbour.

1.2 KNN generalization error when K varies

As we are analyzing generalization error(by having some data separated for testing), when $K = 1$, error rate will be high as due to smaller value of K, the prediction will be susceptible to noise. As we increase the value of K, the error rate will keep decreasing. When K is close to n, error rate may again start to raise little bit as with high value of K, data imbalance may effect the predictions. Figure 1 depicts the nature of the curve for generalization error when K varies.

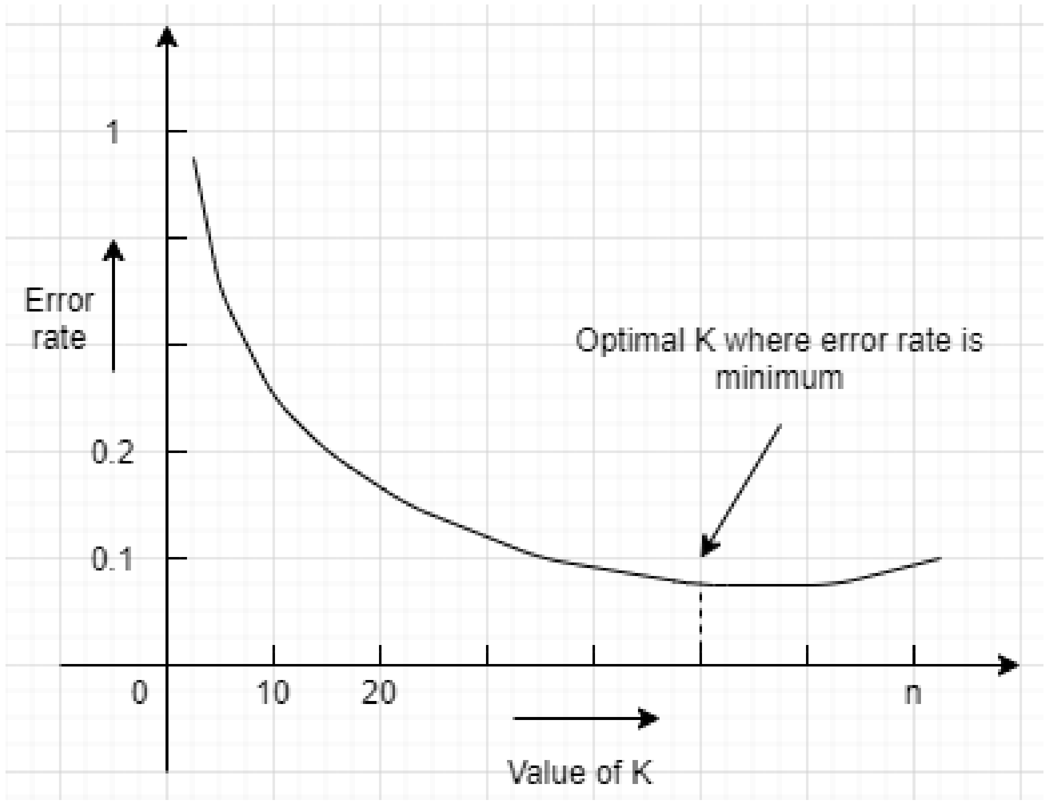

Figure 1: Generalization error of KNN when K varies from 1 to n

1.3 Problems with KNN for high dimensional data

Following are the two major problems with KNN while using with high dimensional data:

1. As KNN is a lazy learning algorithm, it needs to store all the data points at the runtime to predict for a new data point or query. So when dimensionality increases, we need more storage and computation also becomes difficult or slow at runtime.

2. It's being observed and can be proved that for high dimensional data, all the data points appear to be on the surface of a sphere or the data points get evenly distributed and appears to be equidistant. So it becomes difficult for KNN algorithm to differentiate and predict properly. This problem is called "curse of dimensionality".

1.4 Decision tree as KNN

The question given is if we can compare 1-NN and an univariate decision tree such that both classifies in the same way.

If we consider the most basic one dimensional scenario where decision boundary is well defined, in such specific case, 1-NN and decision tree both can classify in the same way as eventually they both can have the same decision boundary which is just a line parallel to Y axis.

For a general case, if the number of dimension is more than 1, it's simply not possible as 1-NN and decision tree algorithm will not draw the decision boundary in the same way. The fact is easily understandable when we see the Voronoi diagram in figure 2 that depicts the decision boundary using 1-NN and the figure 3 that depicts the decision boundary using decision tree for a 2 dimensional scenario. 1-NN can draw quite complex and non-linear decision boundary but in case of decision tree, it's a combination of lines which are parallel to X and Y axis.

In our scenario in the question, it's mentioned that there are 2 dimensions and data for the 2 classes are overlapped to some extent. As per the above discussion, I believe a univariate decision tree will not be exactly same as 1-NN in this case.

2 Naive Bayes classifier question

2.1 Gaussian Naive Bayes classifier

Given,
Training examples from class $1(C_1) = \{0.5, 0.1, 0.2, 0.4, 0.3, 0.2, 0.2, 0.1, 0.35, 0.25\}$
Training examples from class $2(C_2) = \{0.9, 0.8, 0.75, 1.0\}$
Variance of Class $1(\sigma_{c1}^2) = 0.0149$ and Variance of Class $2(\sigma_{c2}^2) = 0.0092$

Now,
Probability of class 1 $= P(C_1) = \frac{10}{10+4} = 0.714$
Probability of class 2 $= P(C_2) = \frac{4}{10+4} = 0.285$
Mean of Class $1(\mu_{c1}) = \frac{0.5+0.1+0.2+0.4+0.3+0.2+0.2+0.1+0.35+0.25}{10} = 0.26$
Mean of Class $2(\mu_{c2}) = \frac{0.9+0.8+0.75+1.0}{4} = 0.8625$

We need to find probability that the test point $x = 0.6$ belongs to class $1(P(C_1|x = 0.6))$.
Probability density of "x = 0.6", given it's class 1 $= P(x = 0.6|C_1)$

$$= \frac{1}{\sqrt{2\pi\sigma_{c1}^2}} exp^{\frac{-1}{2}\cdot\frac{(x-\mu_{c1})^2}{\sigma_{c1}^2}}$$
$$= \frac{1}{\sqrt{2\pi \times 0.0149}} exp^{\frac{-1}{2}\cdot\frac{(0.6-0.26)^2}{0.0149}}$$
$$= 0.06754$$

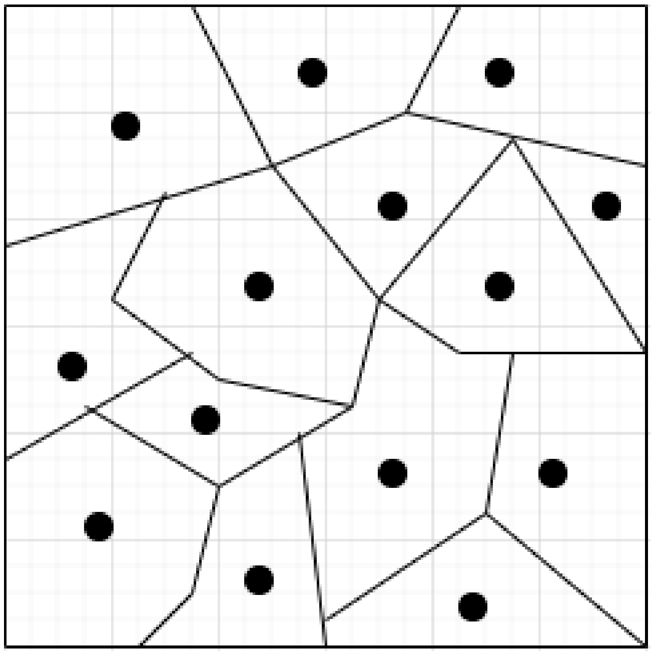
Figure 2: Voronoi diagram that depicts decision boundary of 1NN

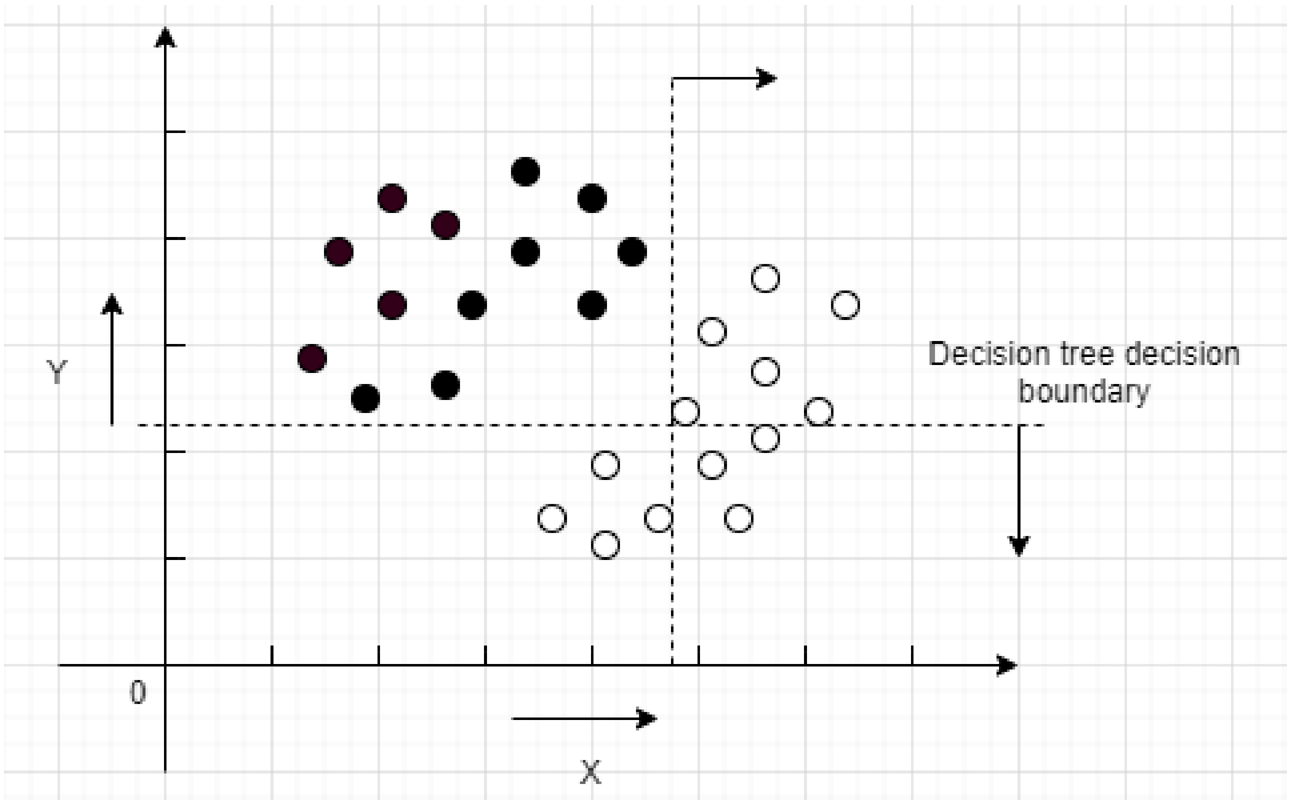

Figure 3: Decision boundary of a decision tree

Similarly, probability density of "x = 0.6", given it's class 2 = $P(x = 0.6|C_2) = 0.098316$

Now, using Bayes theorem we know,

$$\begin{aligned} P(C_1|x = 0.6) &= \frac{P(x = 0.6|C_1)P(C_1)}{P(x = 0.6|C_1)P(C_1) + P(x = 0.6|C_2)P(C_2)} \\ &= \frac{0.06754 \times 0.714}{0.06754 \times 0.714 + 0.098316 \times 0.285} \\ &= 0.63 \end{aligned}$$

2.2 Naive Bayes text classifier

Prior probability of "politics" = $P(politics) = \frac{6}{12} = \frac{1}{2}$
Prior probability of "sport" = $P(sport) = \frac{6}{12} = \frac{1}{2}$

$P(goal|politics) = \frac{2}{6}\ and\ P(goal|sport) = \frac{4}{6}$
$P(football|politics) = \frac{1}{6}\ and\ P(football|sport) = \frac{4}{6}$
$P(golf|politics) = \frac{1}{6}\ and\ P(golf|sport) = \frac{1}{6}$
$P(defence|politics) = \frac{5}{6} =\ and\ P(defence|sport) = \frac{4}{6}$
$P(offence|politics) = \frac{5}{6}\ and\ P(offence|sport) = \frac{1}{6}$
$P(wicket|politics) = \frac{1}{6}\ and\ P(wicket|sport) = \frac{1}{6}$
$P(office|politics) = \frac{4}{6}\ and\ P(office|sport) = \frac{0}{6} = 0$
$P(strategy|politics) = \frac{5}{6}\ and\ P(strategy|sport) = \frac{1}{6}$

Now, we need to find the probability of document x = (1,0,0,1,1,1,1,0) is about "politics" using a maximum likelihood naive Bayes classifier.

$$\begin{aligned} P(x|politics) &= P(\frac{goal}{politics})\overline{P}(\frac{football}{politics})\overline{P}(\frac{golf}{politics})P(\frac{defence}{politics})P(\frac{offence}{politics})P(\frac{wicket}{politics})P(\frac{office}{politics})\overline{P}(\frac{strategy}{politics}) \\ &= \frac{2}{6}.(1 - \frac{1}{6}).(1 - \frac{1}{6}).\frac{5}{6}.\frac{5}{6}.\frac{1}{6}.\frac{4}{6}.(1 - \frac{5}{6}) \\ &= \frac{5^4.8}{6^8} \end{aligned}$$

$$\begin{aligned} P(x|sport) &= P(\frac{goal}{sport})\overline{P}(\frac{football}{sport})\overline{P}(\frac{golf}{sport})P(\frac{defence}{sport})P(\frac{offence}{sport})P(\frac{wicket}{sport})P(\frac{office}{sport})\overline{P}(\frac{strategy}{sport}) \\ &= \frac{4}{6}.(1 - \frac{4}{6}).(1 - \frac{1}{6}).\frac{4}{6}.\frac{1}{6}.\frac{1}{6}.0.(1 - \frac{1}{6}) \\ &= 0 \end{aligned}$$

So the probability that given document "x" is about "politics" using maximum likelihood naive Bayes classifier,

$$\begin{aligned} P(politics|x) &= \frac{P(politics \cap x)}{P(x)} \\ &= \frac{P(\frac{x}{politics})P(politics)}{P(\frac{x}{politics})P(politics) + P(\frac{x}{sport})P(sport)} \\ &= 1 (As,\ P(x|sport)\ is\ 0) \end{aligned}$$

3 Decision tree building question

3.1 Building the decision tree

The recursive python code to build the decision tree is present in the Jupiter notebook that is submitted with this report. This is an univariate decision tree that uses entropy to measure information gain to decide best split at each node.

For the numeric features, to find the right threshold to split, I have used the approach to try all possible unique values of the feature, calculate the information gain and finally used the value that maximizes the information gain. The logs for different results are present in the Tamal-Mondal-result.txt file.

3.2 Calculate average accuracy using 10-fold Cross-validation

After 10-fold cross-validation, the accuracies of the 10 iteration were as follows: [0.836, 0.856, 0.823, 0.836, 0.829, 0.850, 0.840, 0.858, 0.827, 0.836] and so the average accuracy was around **83.95%** for the given wine dataset. Here the decision tree grows until all the leaves become pure and there is no restriction on tree height.

The cross-validation method is also written from the scratch that also does the stratified sampling for every folds(to maintain the same class proportions). The cross-validation code is also present in the same Jupiter notebook.

3.3 Improvement strategies

1. **Using pre-pruning:** Pruning is one of the most common way to improve the accuracy of a decision tree. Here I have used the idea of pre-pruning or early-stopping by restricting the height of the tree and the number of data points needed to make a split. In both the ways we are trying to avoid overfitting by not making the leaves completely pure but taking a generalized decision by majority voting about class label on the leaves.

   I did experiment for a set of possible heights and split values with 10-Fold cross-validation using the given wine dataset. The best average accuracy I got was around **84.04%** by using maximum-height = 23 and minimum samples to split = 2 which is a small improvement in the accuracy. Any maximum height value lesser than 23 was deteriorating the performance.

2. **Using Gini index:** Entropy is not the only way to measure information gain, another popular way to compute information gain is using Gini index. The gini index of a node is defined as $1 - \sum_{i=1}^{n} P_i^2$, where P_i is the probability of ith class. Now accuracy wise generally we don't see much difference while using entropy or gini index. The main difference is that calculating Gini index is somewhat simpler and computationally efficient than entropy as entropy has logarithmic terms. This is the reason to prefer Gini index over entropy when the dataset is large.

   Using Gini index in place of Entropy to calculate information gain, after 10-fold cross-validation the average accuracy was **78.57%**, which is less than our base decision tree model that uses entropy.

3. **Using post-pruning:** Post-pruning is when we let the decision tree grow as much as it can with default settings(so no early stopping) and then depending on certain criteria we prune the tree in a bottom up fashion. As criterion, I have used miss-classification error which means if splitting the node results in same or more miss-classification error then we prune the sub-tree or convert that decision node to a leaf. This way we can reduce the size of the decision tree to take more generalized decision and avoid overfitting.

   Using 10-fold cross-validation, before post-pruning the average accuracy was **82.8%** and after post-pruning the average accuracy was **82.97%**.