Foundations of Machine Learning

Subject Code: CS5590

Assignment-1

Submitted by:

Antala Aviraj
Roll No: CS24MTECH14011

Department of Computer Science and Engineering
IIT-Hyderabad
Date: September 4, 2024

September 4, 2024

1) K-NN

Answer A: Training Error Analysis

Analysis of k-NN as k Varies from n to 1

- **When $k = n$:**

  - The algorithm considers all n data points to classify a particular data point.
  - Given that both classes have $\frac{n}{2}$ data points, the majority vote will always be tied.
  - This tie results in a high training error.

- **As k decreases:**
  - Training error decreases because the algorithm captures more local points.
  - The tied condition is broken as k becomes smaller, leading to lower training error.

- **When $k = 1$:**
  - Since each point is its own neighbor, the training error becomes zero.
  - Each point is classified correctly, resulting in zero training error.

Answer B: Generalization Error Analysis

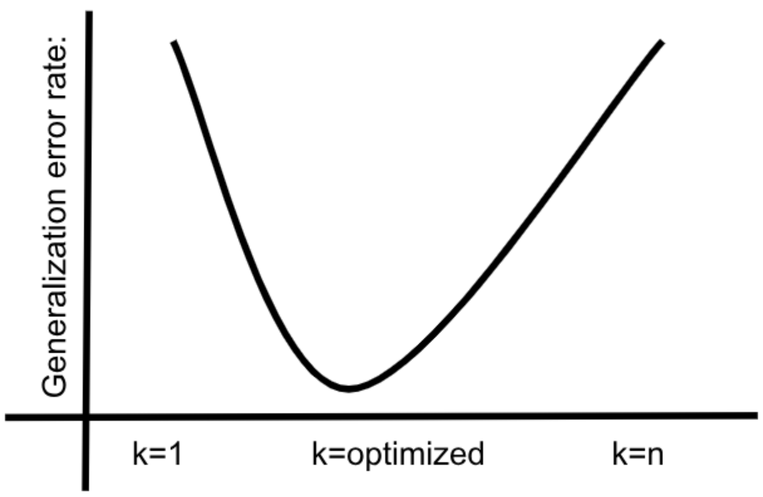
- **Generalization Error:**
  - Refers to the error on unseen data.

- **When $k = n$:**
  - The algorithm considers all data points in the class as neighbors.
  - It tries to capture all points for classification, ignoring many local patterns.
  - Given $\frac{n}{2}$ data points in both classes, it results in a tie, leading to high generalization error.
  - This scenario leads to an underfitting problem.

- **As k decreases:**
  - The model starts to capture more local patterns, solving the underfitting problem.
  - Generalization error decreases as the model becomes more flexible.
  - There exists an optimal k value where the generalization error is minimized.

- **When $k = 1$:**
  - The model becomes more complex, trying to fit all data points, leading to an overfitting problem.
  - As k approaches 1, the generalization error increases.

- **Conclusion:**
  - High generalization error occurs when $k = n$ (underfitting) and $k = 1$ (overfitting).



Sketch of Generalization Error as k Varies

Answer C: Univariate Decision Tree

In the context of the question, a univariate decision tree is a tree where each internal node involves a univariate decision of the form "is $x > a$" or "is $x < b$". The tree makes decisions based on a single feature at a time and creates axis-aligned splits in the graph or space.

What are Axis-Aligned Splits?

Suppose we have a 2D feature space with features x and y. An axis-aligned split would involve decisions like:

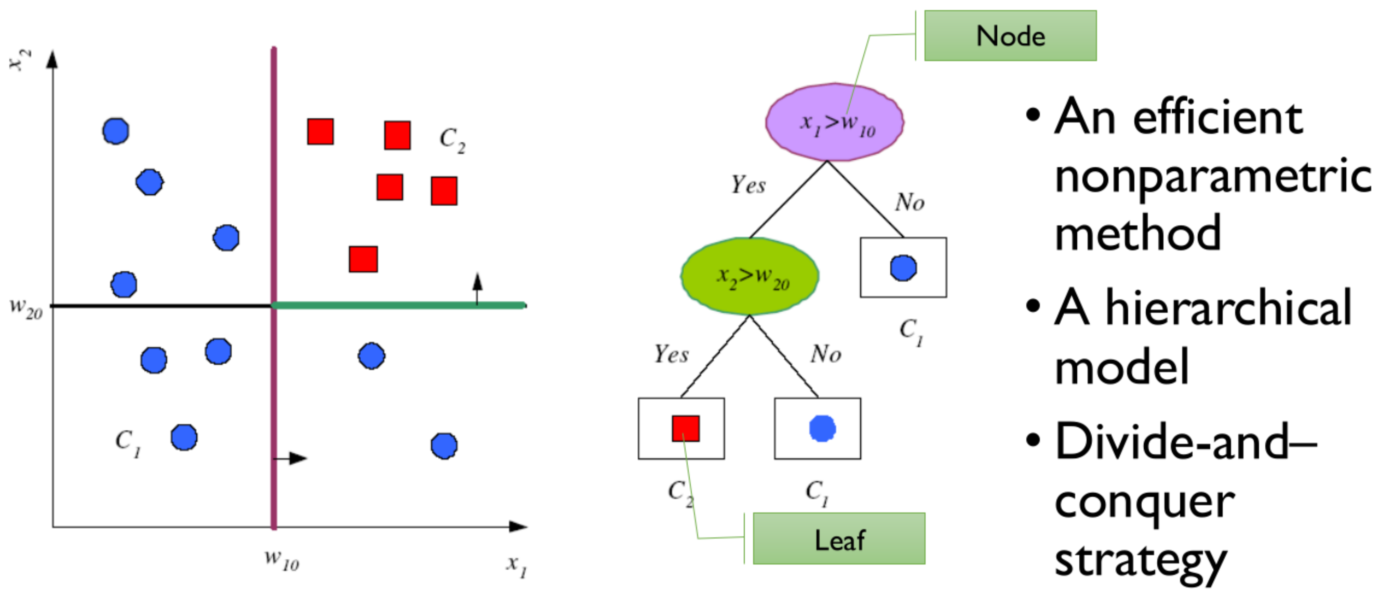
Axis-Aligned Splits Example

- $W_{10} > X_1$ creates a vertical split along the y-axis.
- $W_{20} > X_2$ creates a horizontal split along the x-axis.

This creates a box or rectangular type of region. Based on this, we can say that univariate decision trees have linear boundaries or are axis-aligned.

Example:

Consider a simple case where the data points are arranged such that the nearest neighbor decision boundary forms a curve or an irregular shape. A univariate decision tree, however, would only be able to create straight-line boundaries along the feature axes.



- An efficient nonparametric method
- A hierarchical model
- Divide-and–conquer strategy

Conclusion

It is not possible to make a univariate decision tree that exactly replicates 1-NN classifiers using the Euclidean distance measure.

2) Bayes Classifier

Answer A)

For Gaussian probability we need to first calculate mean value for both class

(μ_1) value for class 1:

Given data: $\{0.5, 0.1, 0.2, 0.4, 0.3, 0.2, 0.2, 0.1, 0.35, 0.25\}$

$$\mu_1 = \frac{1}{10}\sum_{i=1}^{10} x_i = \frac{0.5+0.1+0.2+0.4+0.3+0.2+0.2+0.1+0.35+0.25}{10}$$

$$\mu_1 = \frac{2.6}{10} = 0.26$$

(μ_2) value for class 2:

Given data: $\{0.9, 0.8, 0.75, 1.0\}$

$$\mu_2 = \frac{1}{4}\sum_{i=1}^{4} x_i = \frac{0.9+0.8+0.75+1.0}{4}$$

$$\mu_2 = \frac{3.45}{4} = 0.8625$$

Calculate the class probabilities p_1 and p_2

$$p_1 = \frac{10}{14} = 0.714$$
$$p_2 = \frac{4}{14} = 0.286$$

Now we have to check for that x=0.6 belong to class 1 or class 2
We want to calculate this part for our answer:

$$P(C_1 \mid x = 0.6) = \frac{P(x \mid C_1)P(C_1)}{P(x \mid C_1)P(C_1) + P(x \mid C_2)P(C_2)}$$

For that we need to calculate this:

$$P(x \mid C_i) = \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\frac{(x-\mu_i)^2}{2\sigma_i^2}\right)$$

Now we have:

$$\sigma_1^2 = 0.0149$$
$$\sigma_2^2 = 0.0092$$
$$\mu_1 = 0.26$$
$$\mu_2 = 0.8625$$
$$p_1 = 0.714$$
$$p_2 = 0.286$$

We need to calculate that $x = 0.6$ given that weather it belong to C_1, C_2

$$P(x = 0.6 \mid C_1) = \frac{1}{\sqrt{2\pi \times 0.0149}} \exp\left(-\frac{(0.6-0.26)^2}{2 \times 0.0149}\right)$$

$$P(x = 0.6 \mid C_1) = \frac{1}{\sqrt{0.0935}} \exp\left(-\frac{0.1156}{0.0298}\right)$$

$$P(x = 0.6 \mid C_1) = \frac{1}{0.3059} \exp\left(-3.8782\right) = 0.0676$$

For $P(C_1 \mid x = 0.6)$

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

$$P(x = 0.6 \mid C_2) = \frac{1}{\sqrt{0.0578}} \exp\left(-\frac{0.0689}{0.0184}\right)$$

$$P(x = 0.6 \mid C_2) = \frac{1}{0.2404} \exp\left(-3.7456\right) = 0.0983$$

Now let's calculate: $P(C_1 \mid x = 0.6)$

$$P(C_1 \mid x = 0.6) = \frac{0.0676 \times 0.714}{0.0676 \times 0.714 + 0.0983 \times 0.286}$$
$$P(C_1 \mid x = 0.6) \approx \frac{0.04826}{0.04826 + 0.02811}$$
$$= \frac{0.04826}{0.07637}$$
$$= 0.63$$

So the answer for the$P(C_1 \mid x = 0.6)$ **is approx to 0.63**

Answer B)

In question Two matrix is given one is politics and other is sports:

$$\mathbf{x}_{\text{politics}} = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{x}_{\text{sport}} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Also there are 8 attribute is present in the matrix
value of the attribute vector is:

$$x = (goal, football, golf, defence, offence, wicket, office, strategy)$$

Let's calculate the probability for $P(politics)$,$P(sports)$

$$P(politics) = \frac{6}{6+6} = 0.5 \tag{1}$$

$$P(sports) = \frac{6}{6+6} = 0.5 \tag{2}$$

In question we observe one thing that in the sports matrix wicket column is whole 0 whenever this type of data occur our naive Bayes solution is more biased to the other matrix in our case politics ,so we need to apply here pseudocounts or laplace smoothing.

Now,let's calculate posterior probability for politics,

$$P(goal|politics) = \frac{2+1}{6+8} = \frac{3}{14}$$

$$P(football|politics) = \frac{1+1}{6+8} = \frac{2}{14}$$

$$P(golf|politics) = \frac{1+1}{6+8} = \frac{2}{14}$$

$$P(defence|politics) = \frac{5+1}{6+8} = \frac{6}{14}$$

$$P(offence|politics) = \frac{5+1}{6+8} = \frac{6}{14}$$

$$P(wicket|politics) = \frac{1+1}{6+8} = \frac{2}{14}$$

$$P(office|politics) = \frac{4+1}{6+8} = \frac{5}{14}$$

$$P(strategy|politics) = \frac{5+1}{6+8} = \frac{6}{14}$$

Now,let's calculate posterior probability for sports,

$$P(goal|sports) = \frac{4+1}{6+8} = \frac{5}{14}$$

$$P(football|sports) = \frac{4+1}{6+8} = \frac{5}{14}$$

$$P(golf|sports) = \frac{1+1}{6+8} = \frac{2}{14}$$

$$P(defence|sports) = \frac{4+1}{6+8} = \frac{5}{14}$$

$$P(offence|sports) = \frac{1+1}{6+8} = \frac{2}{14}$$

$$P(wicket|sports) = \frac{1+1}{6+8} = \frac{2}{14}$$

$$P(office|sports) = \frac{0+1}{6+8} = \frac{1}{14}$$

$$P(strategy|sports) = \frac{1+1}{6+8} = \frac{2}{14}$$

Now we have to calculate that particular vector x belongs to politics class,

$P(x = (1, 0, 0, 1, 1, 1, 1, 0)|politics)$

For this,

$$\begin{aligned} P(x \mid \text{politics}) = & P(\text{goal} \mid \text{politics}) \cdot P(\text{football} \mid \text{politics})' \cdot P(\text{golf} \mid \text{politics})' \cdot P(\text{defence} \mid \text{politics}) \\ & \cdot P(\text{offence} \mid \text{politics}) \cdot P(\text{wicket} \mid \text{politics}) \cdot P(\text{office} \mid \text{politics}) \cdot P(\text{strategy} \mid \text{politics})' \\ & = \frac{4}{14} \cdot \left(1 - \frac{2}{14}\right) \cdot \left(1 - \frac{1}{14}\right) \cdot \frac{6}{14} \cdot \frac{5}{14} \cdot \frac{2}{14} \cdot \frac{7}{14} \cdot \left(1 - \frac{5}{14}\right) \\ & = \frac{4}{14} \cdot \frac{12}{14} \cdot \frac{13}{14} \cdot \frac{6}{14} \cdot \frac{5}{14} \cdot \frac{2}{14} \cdot \frac{7}{14} \cdot \frac{9}{14} \\ & = 0.000843 \end{aligned} \tag{3}$$

$$\begin{aligned} P(x \mid \text{sports}) = & P(\text{goal} \mid \text{sports}) \cdot P(\text{football} \mid \text{sports})' \cdot P(\text{golf} \mid \text{sports})' \cdot P(\text{defence} \mid \text{sports}) \\ & \cdot P(\text{offence} \mid \text{sports}) \cdot P(\text{wicket} \mid \text{sports}) \cdot P(\text{office} \mid \text{sports}) \cdot P(\text{strategy} \mid \text{sports})' \\ & = \frac{5}{14} \cdot \left(1 - \frac{5}{14}\right) \cdot \left(1 - \frac{2}{14}\right) \cdot \frac{5}{14} \cdot \frac{2}{14} \cdot \frac{2}{14} \cdot \frac{1}{14} \cdot \left(1 - \frac{2}{14}\right) \\ & = \frac{5}{14} \cdot \frac{9}{14} \cdot \frac{12}{14} \cdot \frac{5}{14} \cdot \frac{2}{14} \cdot \frac{2}{14} \cdot \frac{1}{14} \cdot \frac{12}{14} \\ & = 0.000088 \end{aligned} \tag{4}$$

Probability of $P(politics|x = (1, 0, 0, 1, 1, 1, 1, 0))$ is

$$\begin{aligned} P(politics|x) &= \frac{P(x|politics)P(politics)}{P(x|politics)P(politics) + P(x|sports)P(sports)} \\ &= \frac{0.000843 \times 0.5}{0.000843 \times 0.5 + 0.000088 \times 0.5} \\ &= 0.9054 \end{aligned} \tag{5}$$

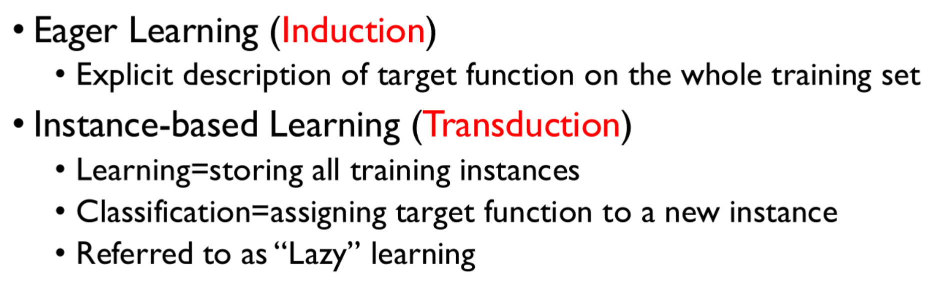
So the probability of dataset x belongs to politics is approx 0.9054

4)Method Comparison

Answer A)

1. k-NN

- **Training time complexity:**
  k-NN is a lazy algorithm and also a transductive type of algorithm, meaning it is instance-based learning.



- Eager Learning (Induction)
  - Explicit description of target function on the whole training set
- Instance-based Learning (Transduction)
  - Learning=storing all training instances
  - Classification=assigning target function to a new instance
  - Referred to as "Lazy" learning

So it requires only $O(1)$ time for training the model.

- **Inference time complexity:**
  For each new data point that we want to classify or predict, we need to compute the distance between this point and each of the N training points.
  - Time complexity for calculating the distance: $O(d)$
  - This needs to be calculated for N points, so the total time complexity is: $O(N \times d)$

2. Decision Tree

- **Training time complexity:**
  In a decision tree, there are two types of features possible:
  - **Categorical:**
  - **Continuous:**

  Now let's assume some $(\frac{d}{2})$ features are categorical and some $(\frac{d}{2})$ are continuous. For finding the threshold, we need to sort the continuous features. For N training data, that requires $O(N \log N)$ time. For $(\frac{d}{2})$ features, the total time is $O(d \times N \times \log N)$ (ignoring the constant).

  Also, we can convert those $(\frac{d}{2})$ categorical features into continuous by some method. Now, if all the features are continuous, then the total time is $O(d \times N \times \log N)$. If all the features are categorical and we do a binary split, then the total time is $O(d \times N \times 2^k - 1)$.

- **Inference time complexity:**
  Once we already have a decision tree with depth d, we only need to traverse it for testing the particular data point. So the total time is $O(\log d)$.

3. Naive Bayes

- **Training time complexity:**

  Let's assume we have:
  N: the total number of training data
  D: the total number of features per sample
  C: the number of distinct classes

  We have to calculate the conditional probabilities for each of the D features. For each feature, we have to go through all N data to compute the probabilities.

  So the time required is $O(N \times D)$. We have to do that for all C classes. So the total time required is $O(N \times D \times C)$.

- **Inference time complexity:**
  For a particular testing instance that contains D features, we have to calculate the probability for all C classes.

  So the time complexity is $O(D \times C)$, because here N becomes 1.

Classifier Comparison

Table for complexity

Classifier	Training Complexity	Inference Complexity
k-NN	$O(1)$	$O(N \times d)$
Decision Tree	$O(d \times n \times \log n)$	$O(\log d)$
Naive Bayes	$O(N \times D \times C)$	$O(D \times C)$

Table 1: Training and Inference Complexities for Different Classifiers

Answer B)

1] Large Dataset, Large Number of Features

1. k-NN

Strengths:

- For large features, k-NN can adapt complex and non-linear decision boundaries.
- It also provides good accuracy in terms of large datasets and high-dimensional datasets.

- With more data, density is high between the data points, or it generates a more dense model, which provides a good approximation and true distance to the meaningful neighbor.

Weaknesses:

- For large datasets and a large number of features, the time required to test the model is high.
- For each test data point, the algorithm must compute the distance to all N data points. Thus, time complexity increases with large datasets.
- k-NN requires storing all the data points in memory because it is a transductive type of classifier. Large datasets thus require a huge amount of memory.
- With more features, k-NN suffers from the curse of dimensionality.

2. Decision Tree

Strengths:

- With large datasets and a large number of features, it has more options to split features, allowing it to generate complex but accurate decision trees, which results in good accuracy.

Weaknesses:

- With large datasets and a large number of features, the decision tree can become very complex.
- Due to the large number of features, it can also lead to overfitting problems.

3. Naive Bayes

Strengths:

- With large datasets, it only needs to store probabilities, which requires less space.
- It is scalable for handling large amounts of data.
- It works well with a high number of features because it assumes that each feature is independent of the others.

Weaknesses:

- A major problem is that Naive Bayes classifiers assume that all features are conditionally independent.
- If features are highly correlated with each other, it gives inaccurate results.

2] Large Dataset, Small Number of Features

1. k-NN

Strengths:

- With a smaller number of features, k-NN is still effective because the curse of dimensionality is decreased.
- Also, with a small number of features, the decision metrics created by k-NN are more precise.

Weaknesses:

- For large datasets, the time required to test the model is high.
- k-NN requires storing all the data points in memory because it is a transductive type of classifier. Therefore, if datasets are large, it requires a huge amount of memory.
- For each test data point, we need to calculate the distance, which results in slow prediction time.

2. Decision Tree

Strengths:

- With a small number of features, training time is faster compared to a large number of features.
- The decision tree becomes simpler compared to the cases with a larger number of features.

Weaknesses:

- Due to the large number of data points, the decision tree can still lead to overfitting problems.

3. Naive Bayes

Strengths:

- With large datasets, it only needs to store probabilities, which requires less space. So with a small number of features, it handles large datasets very well.
- It is also scalable for handling large amounts of data.

Weaknesses:

- A major problem is that Naive Bayes classifiers assume that all features are conditionally independent.
- Because of this assumption, if features are highly correlated with each other, it gives inaccurate results. This issue persists regardless of the number of features.

3] Small Dataset, Large Number of Features

1. k-NN

Strengths:

- With small datasets, k-NN still provides good results and captures local neighbors effectively.
- Since the dataset is small, it requires less memory compared to other cases.

Weaknesses:

- With more features, k-NN suffers from the curse of dimensionality. The distance between points becomes less meaningful in high-dimensional space.
- Additionally, with small datasets and a large number of features, overfitting is also possible.

2. Decision Tree

Strengths:

- With a large number of features, it has more options to split features, allowing it to generate complex but accurate decision trees which result in good accuracy. This is true in both cases where the number of features is large.

Weaknesses:

- Due to the small number of data points and a large number of features, decision trees can lead to overfitting problems.
- The trees become very complex and overfit the data.

3. Naive Bayes

Strengths:

- With a large number of features, it still requires less time to train or test the data.
- It works well with high-dimensional features because it assumes that features are independent of each other.

Weaknesses:

- A major problem is that Naive Bayes classifiers assume that all features are conditionally independent.
- Because of this assumption, if features are highly correlated with each other, it gives inaccurate results. This issue is present regardless of the number of features.

4] Small Dataset, Small Number of Features

1. k-NN

Strengths:

- With a small number of features, k-NN can measure distance effectively between data points without the curse of dimensionality becoming a problem.
- It requires less space to store the training data.
- The risk of overfitting is also less with a small data set and small feature set.

Weaknesses:

- Even with a small dataset, it requires more time to compute distances.
- Selecting the optimal k for small data is more difficult.
- A small k might lead to overfitting, while a large k may lead to underfitting.

2. Decision Tree

Strengths:

- With small datasets and a small number of features, decision tree classifiers generate simple models that are easy to analyze.
- The risk of overfitting is less with a small dataset and a small number of features.

Weaknesses:

- For generalization, a small number of data points does not provide enough examples to fully capture all possibilities.
- With a small number of features and data, it is also possible that the accuracy is not as good.

3. Naive Bayes

Strengths:

- It works very well with a small number of datasets because it doesn't require a large amount of data to calculate probabilities (it also works well with large datasets).
- It handles high-dimensional features well because it assumes that features are independent of each other.
- For small datasets, training the model is much faster.

Weaknesses:

- A major problem is that Naive Bayes classifiers assume that all features are conditionally independent.
- Because of this assumption, if features are highly correlated with each other, it gives inaccurate results. This issue persists regardless of the number of features.