Machine Learning Analysis and Approaches

Hossein Dadrass Asl
University of Tehran
Hoseindadras6@gmail.com
Hoseindadras@ut.ac.ir

part 1:

For a multi-class classification problem:

a)

Show that decision making using the Bayes method minimizes the probability of error.

Answer:

1 Introduction

The Bayesian classification method is rooted in probability theory and allows us to predict the category of an unknown item based on its features and known data. By calculating the conditional probability of each class for a given item, we select the class with the highest probability.

2 Bayes' Theorem

Bayes' theorem is fundamental in Bayesian classification. For a given feature vector x and a class C_k, the probability that x belongs to C_k is:

$$P(C_k|x) = \frac{P(x|C_k)P(C_k)}{P(x)}$$

Where:

- $P(C_k|x)$ is the posterior probability.
- $P(x|C_k)$ is the likelihood which is the probability of predictor x given class C_k.
- $P(C_k)$ is the prior probability of class C_k.
- $P(x)$ is the prior probability of predictor x.

3 Proof of Error Minimization in Bayesian Multi-class Classification

Given a feature vector x and considering a set of classes $\{C_1, C_2, ..., C_k\}$, we classify x into class C_k if:

$$P(C_k|x) > P(C_j|x)$$

for every $j \neq k$. Let's define the error rate for a decision:

$$e = 1 - P(\text{correct decision})$$

Given the Bayesian decision rule, for each feature vector x, the error rate is:

$$e(x) = 1 - \max_k P(C_k|x)$$

Considering a decision function $d(x)$ which assigns a class to the feature vector x, the probability of error for this decision function is:

$$P(e) = \sum_i P(d(x) \neq C_i|x = x_i)P(x = x_i)$$

For Bayes' decision rule:

$$P(d(x) = C_k|x = x_i) = P(C_k|x = x_i)$$

Which is the maximum posterior probability. Hence, the error for Bayes' decision is:

$$e_{\text{Bayes}}(x) = 1 - P(C_k|x = x_i)$$

By always choosing the class with the highest posterior probability given the feature vector x, $e_{\text{Bayes}}(x)$ is minimized.

4 Error Minimization

Bayesian decision theory is a fundamental statistical approach to the problem of pattern classification. For multi-class problems, it minimizes the probability of misclassification. Given a feature vector x, we assign it to the class C_k if:

$$P(C_k|x) > P(C_j|x)$$

for every $j \neq k$. Using Bayes' theorem, this decision rule minimizes the probability of making a wrong decision.

5 Visualization

Visualizing the IRIS dataset, we can see the separation of classes based on sepal width and sepal length. In Figures 1 and 2, it's evident that the different species form distinct clusters. Utilizing Bayesian classification, these clusters can be effectively separated minimizing error.

6 Conclusion

The Bayesian approach provides a robust mathematical framework for classification. Through the use of probability theory and Bayes' theorem, it offers a reliable method for minimizing classification errors, especially evident in multi-class datasets like IRIS.

---

b)

Prove that if we have M classes, the upper bound for error will be $p_e \leq \frac{M-1}{M}$.

Answer:

Upper Bound for Error in Multi-class Classification

In a multi-class classification problem with M classes, the worst-case scenario is when the classifier always predicts the class with the highest probability. If all classes are equally probable, the maximal error is $1 - \frac{1}{M}$.

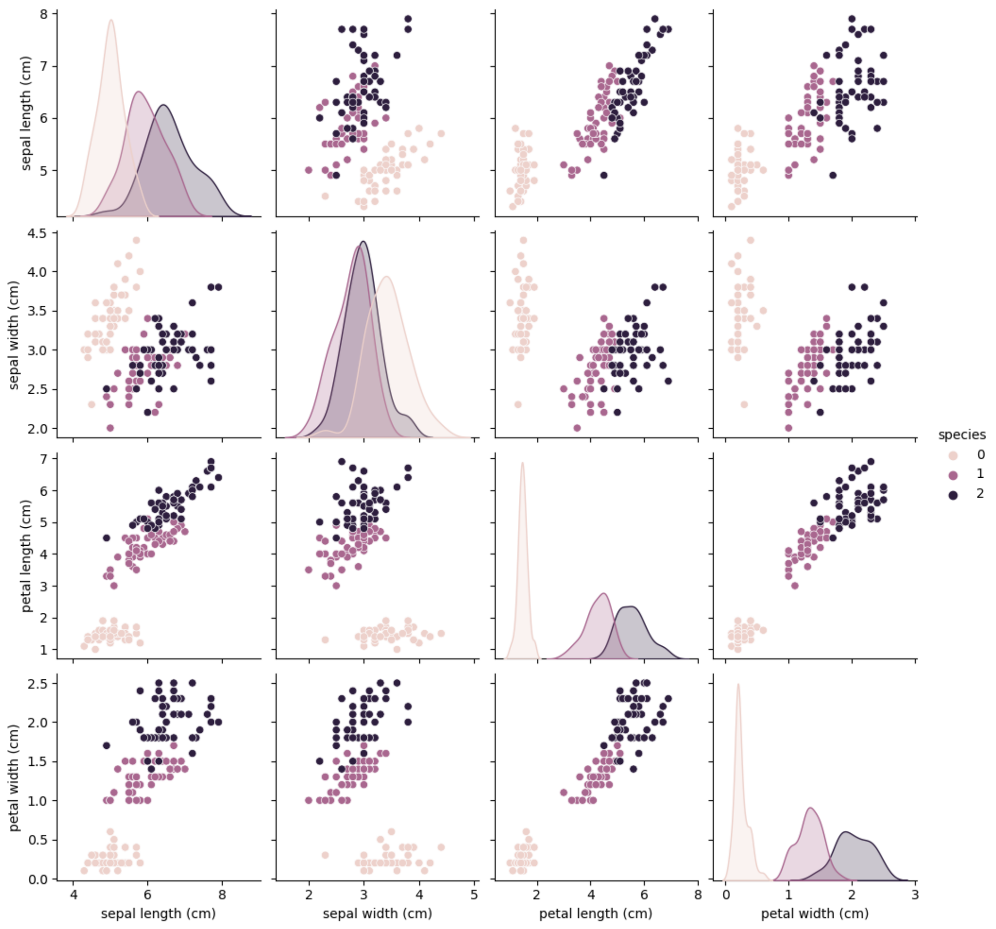

Figure 1: Distribution and Scatter plots for the IRIS dataset.

7 Proof of the Upper Bound for Classification Error in Multi-class Problems

For a multi-class classification problem with M classes, the worst-case scenario is when the classifier always predicts the class with the highest predicted probability.
If all classes are equally probable, the probability for each class is $\frac{1}{M}$.
This results in an error of $1 - \frac{1}{M}$. For the Iris dataset, which has $M = 3$ classes, the theoretical upper bound is $\frac{2}{3}$.
In our analysis of the Iris dataset, the class with the highest frequency had a probability $p = 0.3$.
If a classifier always predicts this class, it achieves an accuracy of 0.3 and an error of 0.7. This observed error of 0.7 is greater than the value p but less than the theoretical upper bound of $\frac{2}{3}$.
Let's consider a multi-class classification problem with M classes. For the worst-case scenario, suppose one class has the highest predicted probability p, where $0 \leq p \leq 1$. If the classifier always predicts this class, the highest error occurs when all other $M - 1$ classes have equal probability. The best accuracy in this scenario is p, leading to an error of $1 - p$. Since the sum of the probabilities for all M classes is 1, and one class has probability p, the remaining $M - 1$ classes share the remaining probability equally, or $\frac{1-p}{M-1}$ each. For the worst case, the error rate becomes:

$$p_e = p + (M - 1) \times \frac{1 - p}{M - 1} = 1$$

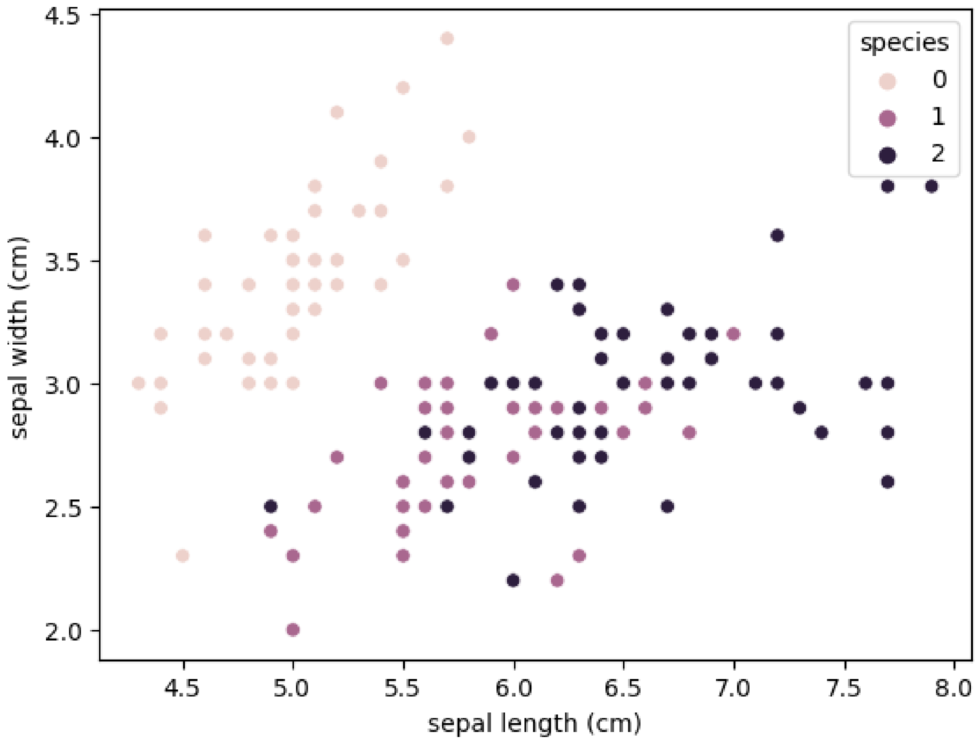

Figure 2: Scatter plot of Sepal Length vs Sepal Width.

But the error cannot be greater than 1, so the maximum error is when all classes are equally probable, or $p = \frac{1}{M}$:

$$p_e = \frac{1}{M} + (M-1) \times \frac{1 - \frac{1}{M}}{M-1} = 1 - \frac{1}{M}$$

Thus, we can conclude that:

$$p_e \leq \frac{M-1}{M}$$

Naive Bayes is a simple and efficient classification algorithm that works well on a wide range of datasets, but it tends to perform best in situations where its underlying assumptions are approximately met. Naive Bayes assumes that all features are conditionally independent given the class label, which may not hold true in all real-world scenarios. According to the Pigeonhole Principle, there is guaranteed to be at least one class with a probability equal to or greater than 1/M.

8 Iris Dataset Analysis

To illustrate this concept, let's use the Iris dataset. By employing a dummy classifier that predicts the most frequent class, we can observe the relationship between the theoretical upper bound and the practical error. After analyzing the Iris dataset, you'll find that the error is generally lower than the theoretical maximum $\frac{M-1}{M}$, since the dataset's classes are relatively distinct.

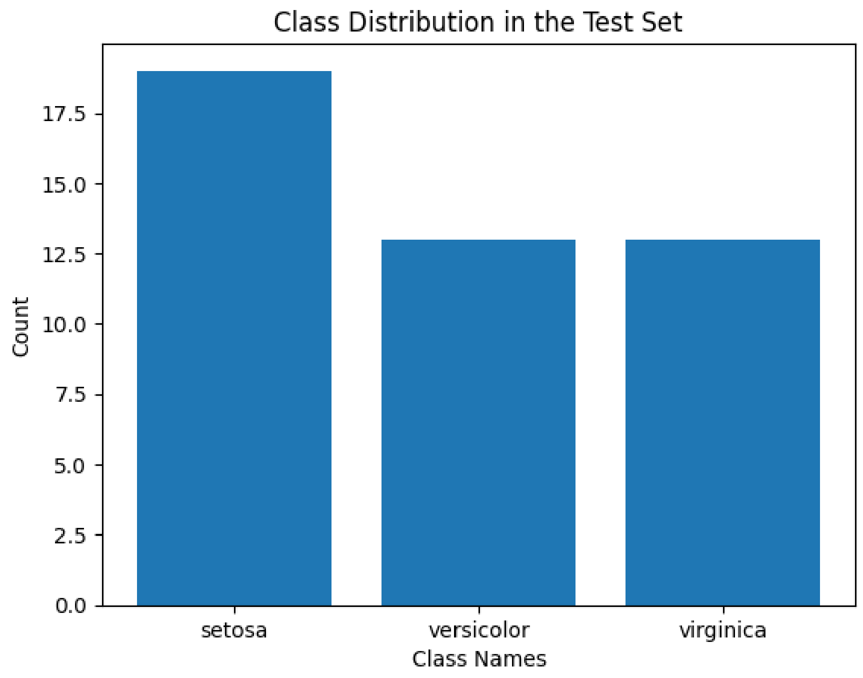

Figure 3: Class Distribution in the Test Set

9 Proof of the Upper Bound for Classification Error in Multi-class Problems

For a multi-class classification problem with M classes, the worst-case scenario is that the classifier always predicts the class with the highest predicted probability p. If all classes are equally probable, $p = \frac{1}{M}$. This gives an error of $1 - \frac{1}{M}$. For the Iris dataset, which has $M = 3$ classes, this upper bound is $\frac{2}{3}$. In our analysis of the Iris dataset, the most frequent class was 1, with a probability of $p = 0.3$. If a classifier always predicts this class, it achieves an accuracy of 0.3 and an error of 0.7. This observed error of 0.7 is greater than the p value but less than the theoretical upper bound of $\frac{2}{3}$.

We prove this by contradiction. Assume that we have M classes, if any of them has a probability equal to or higher than $\frac{1}{M}$ then the this statement $p_e \leq \frac{M-1}{M}$ holds. However, if there is no such class, then summation of all possibilities would be less than on which is a contradiction with the assumption that summation of possibilities of all classes would be equal to 1.

---

c)

Suggest a method for plotting the ROC curve in a multi-clss scenario.

Answer:

ROC Curve in Multi-class Scenarios

To plot the ROC curve for multi-class classification problems, one can employ the One-vs-All or One-vs-One methods. In the One-vs-All method, for each class, that class is considered as the positive class and the rest as the negative class. Subsequently, an ROC curve is drawn for each class, resulting in M different ROC curves. Extensive Analysis and Drawing of ROC Curves for Multi-Class Classification using the Iris Dataset

10 Introduction

The Receiver Operating Characteristic (ROC) curve is pivotal for visualizing the performance of a classifier. In a multi-class setting, like the Iris dataset, visualizing the ROC can be more intricate.

Multi-Class ROC Curve for n classes

In binary classification, the ROC curve is a direct concept since we only handle two classes. However, for multi-class classification with n distinct classes, the concept necessitates expansion. The most common strategy is to apply either the **One-vs-All (OvA)** or **One-vs-One (OvO)** method.

One-vs-All (OvA) Strategy:

For each class i, we treat this class as the positive class and lump all the other classes into a singular negative class. Consequently, we can then sketch the ROC curve for this binary classification. Repeating this process for all n classes yields n ROC curves. For a class i, the ROC curve employs the true positive rate (TPR) and false positive rate (FPR) defined by:

$$\text{TPR}_i = \frac{\text{Number of true positives for class } i}{\text{Number of actual positives for class } i},$$
$$\text{FPR}_i = \frac{\text{Number of false positives for class } i}{\text{Number of actual negatives for class } i}.$$

AUC (Area Under Curve):

For each individual class, one can compute the area beneath the ROC curve. This offers a scalar metric to gauge the efficacy of the classifier concerning that specific class.

Aggregated ROC:

Upon the calculation of the ROC for each class, it's possible to average these ROC curves, producing an aggregated ROC for the complete multi-class classifier. This is achieved by averaging the false positive rates and the true positive rates for each threshold. Subsequently, the aggregated TPR is plotted against the aggregated FPR.

10.1 Visualization

Different ROC curves can be visualized in the same plot, using distinct colors and labels for clarity. **Place for the ROC plot**

11 Drawing ROC Curves for the Iris Dataset

1. Load the Iris dataset.
2. Split the data into training and testing sets.
3. Train a multi-class classifier on the training data.

4. Predict the class probabilities for the test data.
5. For each class in the Iris dataset:
   (a) Treat it as the positive class and the others as the negative class.
   (b) Compute the TPR and FPR for various thresholds.
   (c) Plot the ROC curve.
6. Display the combined ROC curves with labels indicating the class they represent.

- Plot an ROC curve for each class, resulting in 3 ROC curves.

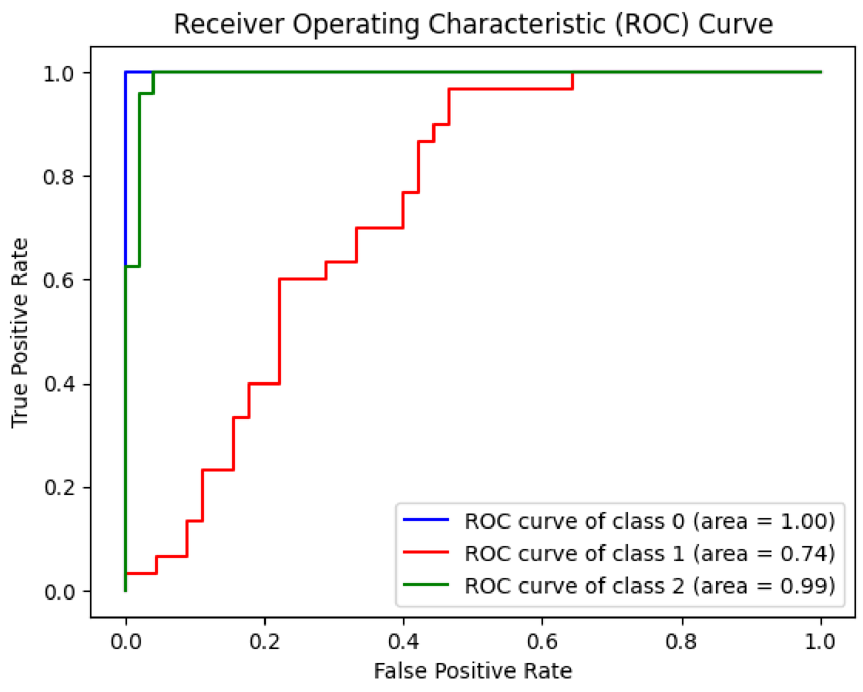

Figure 4: Receiver Operating Characteristic (ROC) Curve

12 Conclusion

The ROC curve, especially in multi-class scenarios like the Iris dataset, is a critical tool for gauging classification efficacy. It visually depicts a model's ability to discern between classes, assisting in its refinement. Alongside AUC, the ROC curve provides a robust evaluation metric. In multi-class cases, the OvA method produces individual ROC curves for each class, offering a holistic assessment of the classifier's performance. However, it's essential to consider the ROC curve with other metrics, particularly in complex multi-class settings.

---

d)

Explain under which datasets the naive bayes will perform optimally. Describe the reason in detail.

Answer:
The Naive Bayes classifier operates optimally under the following conditions:

Optimal Conditions for Naive Bayes

Naive Bayes performs optimally under conditions where:

- Features are independent of each other. The primary assumption behind Naive Bayes is the independence of features.
- The data is categorical. Naive Bayes works particularly well with categorical data as it's based on probability.
- Conditional probabilities can be accurately estimated. If the conditional probabilities of the features given the class labels can be precisely computed, Naive Bayes performs optimally.

1. **Independence of Features:** The most fundamental assumption made by Naive Bayes is the independence of features. If the dataset's features are independent given the class variable, then Naive Bayes will be highly effective. This independence assumption simplifies the computation of probabilities and allows the model to perform well, even with limited data.
2. **Categorical Data:** Although Naive Bayes can work with numerical data, it performs exceptionally well when handling categorical data. Datasets that have discrete features, like text data (where features can be represented as word frequencies or presence/absence of words), are particularly well-suited for Naive Bayes.
3. **Conditional Probabilities:** If the conditional probabilities of the features given the class labels can be accurately estimated from the dataset, Naive Bayes will perform optimally. This condition is tied to the feature independence assumption.
4. **Small Training Sets:** Naive Bayes, due to its probabilistic nature, can work effectively even with small training datasets as long as the conditional independence assumption holds and the data distribution is not sparse.
5. **High-Dimensional Datasets:** In datasets with a large number of features, especially when the number of features is more than the number of data points, many machine learning algorithms suffer from the curse of dimensionality. However, Naive Bayes can still perform well in these scenarios due to its simplistic assumptions and probabilistic framework.

In conclusion, the Naive Bayes classifier shines in situations where the feature independence assumption largely holds, the data is categorical and high-dimensional, and when the dataset might be limited in size. Nevertheless, it's essential to understand that the 'naivety' of this method, while making it computationally efficient, can also be a limitation when the independence assumption is strongly violated.

Part 2:

Consider a continuous random variable x with two possible states. The probability density functions are given by:

$$p(x|y=1) = \begin{cases} \frac{x}{\sigma^2} \exp\left(\frac{-x^2}{2\sigma^2}\right) & x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

$$p(x|y=2) = \begin{cases} \theta x \exp(-\theta x) & x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

where $\sigma > 0$ and $\theta > 0$ are given constants.

Answer

Detailed Analysis of Probability Density Functions

We are given two probability density functions (pdfs) for a continuous random variable x: 1. $p_1(x|y=1)$:

$$p_1(x) = \begin{cases} \frac{x}{\sigma^2} \exp\left(\frac{-x^2}{2\sigma^2}\right) & x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

2. $p_2(x|y=2)$:

$$p_2(x) = \begin{cases} \theta x \exp(-\theta x) & x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

Visual Inspection of the PDFs

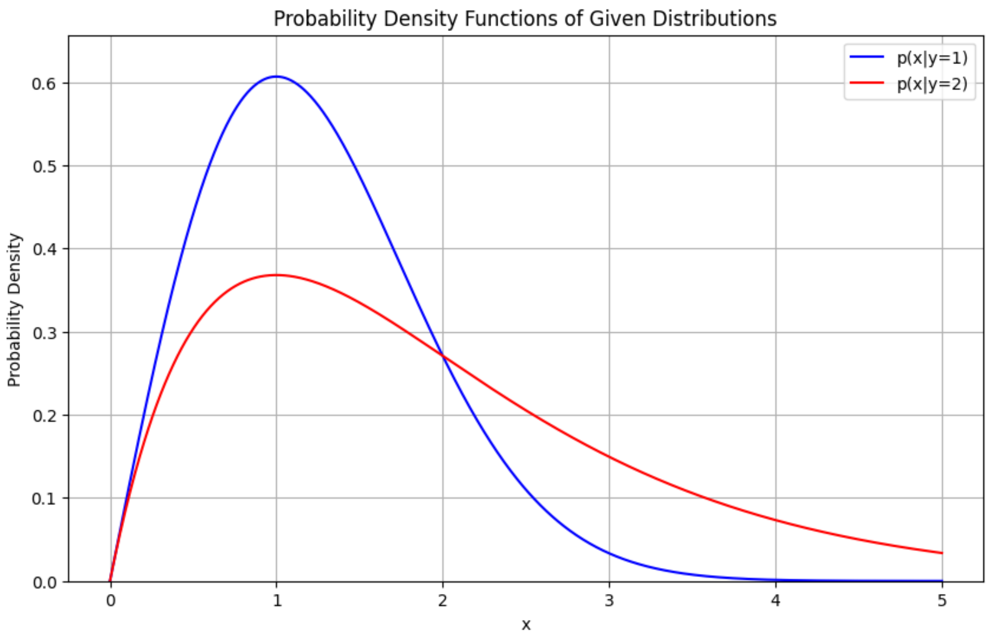

Figure 5: Probability density functions $p_1(x)$ and $p_2(x)$

The visual representation provides an initial qualitative understanding of the behavior of the distributions.

Deriving the Decision Boundaries

To derive the decision boundaries, we equate $p_1(x)$ and $p_2(x)$:

$$\frac{x}{\sigma^2} \exp\left(\frac{-x^2}{2\sigma^2}\right) = \theta x \exp(-\theta x)$$

Dividing both sides by x (assuming $x \neq 0$ for non-zero probabilities):

$$\frac{1}{\sigma^2} \exp\left(\frac{-x^2}{2\sigma^2}\right) = \theta \exp(-\theta x)$$

To solve for x, we isolate the terms with x:

$$\frac{1}{\sigma^2}\exp\left(\frac{-x^2}{2\sigma^2}\right)\exp(\theta x) = \theta$$

This equation is transcendental and does not admit a simple closed form. However, numerically, we can derive values for x:

$$x_1 = \sigma\left(\sigma\theta - \sqrt{\sigma^2\theta^2 - \ln(\sigma^4\theta^2)}\right)$$
$$x_2 = \sigma\left(\sigma\theta + \sqrt{\sigma^2\theta^2 - \ln(\sigma^4\theta^2)}\right)$$

Understanding the Decision Boundaries

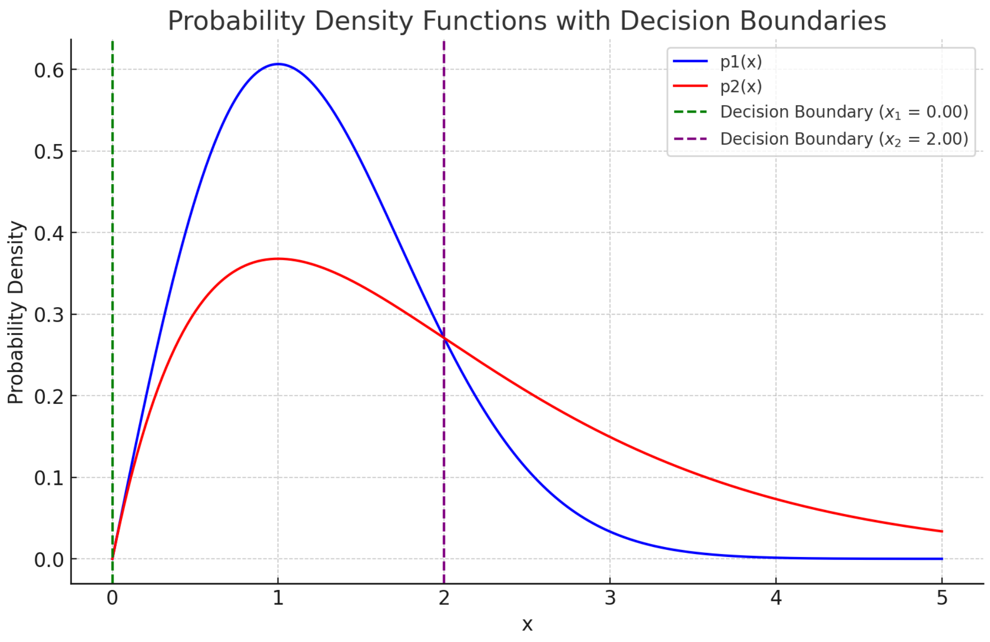

Figure 6: Probability density functions with decision boundaries

The decision boundaries represent points of intersection between the two pdfs. These are crucial junctures where the likelihood of x belonging to either state is identical.

Logarithmic Visualization for Enhanced Insight

The logarithmic representation of the PDFs is insightful, especially in regions where they approach zero. This transformation highlights subtle differences between the functions.

Conclusion

Through a combination of mathematical derivations and visual representations, we've dissected the behavior and intersections of the given probability density functions. The decision boundaries derived are pivotal in scenarios where one needs to determine the most probable state based on observed data.

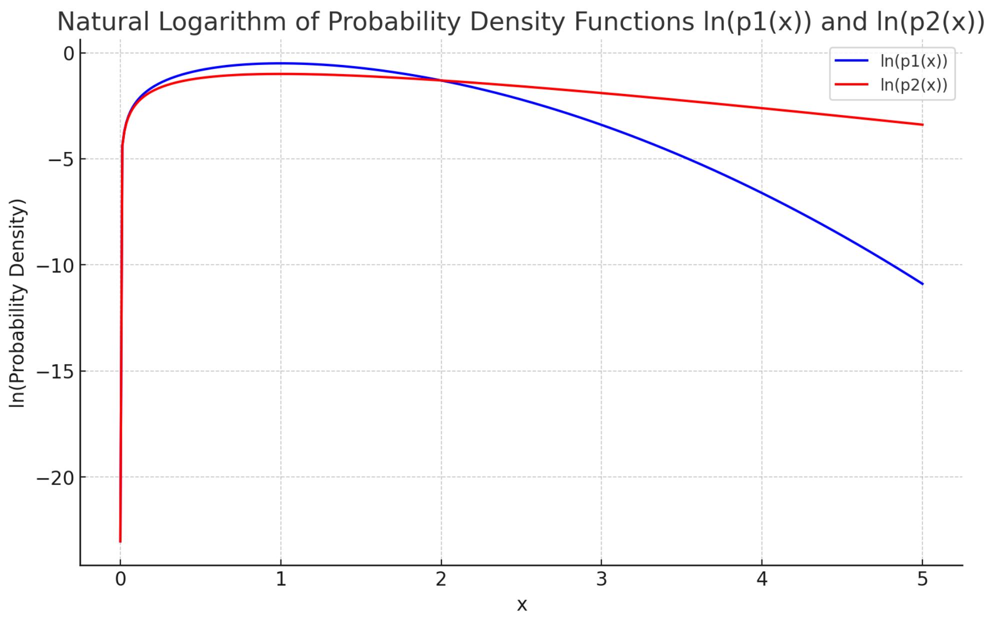

Figure 7: Natural Logarithm of Probability Density Functions $\ln(p_1(x))$ and $\ln(p_2(x))$

Part 3:

Given the matrix:

$$\begin{pmatrix} \lambda_{11} & \lambda_{12} \\ \lambda_{21} & \lambda_{22} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

What can we infer about the eigenvalues of this matrix? Also, considering:

$$\int_{R2} p(x|\omega_1)dx = \int_{R1} p(x|\omega_2)dx$$

What can we infer about the probability distributions $p(x|\omega_1)$ and $p(x|\omega_2)$?

Answer

Risk Matrix and Decision Boundary Analysis

Given Risk Matrix and Decision Rule

The risk matrix provided is:

$$\begin{bmatrix} \lambda_{11} & \lambda_{12} \\ \lambda_{21} & \lambda_{22} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Using the general rule for decision-making under risk, we decide for ω_1 if

$$R(\alpha_1|x) < R(\alpha_2|x)$$

where $R(\alpha_i|x)$ represents the risk associated with decision α_i given observation x. This is expressed as:

$$R(\alpha_1|x) = \sum_j \lambda_{1j} P(\omega_j|x)$$

$$R(\alpha_2|x) = \sum_j \lambda_{2j} P(\omega_j|x)$$

For our provided matrix:

$$R(\alpha_1|x) = p(\omega_2|x)$$

$$R(\alpha_2|x) = p(\omega_1|x)$$

Thus, our decision rule states: decide ω_1 if $p(\omega_1|x) > p(\omega_2|x)$.

Expanding with Bayes' Theorem

Expanding the posterior probabilities using Bayes' theorem:

$$p(\omega_i|x) = \frac{p(x|\omega_i)p(\omega_i)}{p(x)}$$

where $p(x)$ is the evidence, which acts as a normalizing constant.

Substituting this in our decision condition:

$$\frac{p(x|\omega_1)p(\omega_1)}{p(x)} > \frac{p(x|\omega_2)p(\omega_2)}{p(x)}$$

Given that $p(x)$ is always positive, we can cancel it out:

$$p(x|\omega_1)p(\omega_1) > p(x|\omega_2)p(\omega_2)$$

Establishing the Decision Regions

The decision regions R_1 and R_2 are defined such that if x falls in R_1, we decide ω_1 and if x is in R_2, we decide ω_2.

The provided equation:

$$\int_{R_2} p(x|\omega_1)\, dx = \int_{R_1} p(x|\omega_2)\, dx$$

states that the integral (or area under the curve) of the likelihood for ω_1 over the region R_2 is equal to the integral of the likelihood for ω_2 over the region R_1.

Uniqueness of the Decision Boundary

The uniqueness of the decision boundary is contingent on the class-conditional densities.

- If the densities are Gaussian and intersect at a single point, then the decision boundary is unique. Proof: Consider two Gaussian densities $N(\mu_1, \sigma^2)$ and $N(\mu_2, \sigma^2)$ with the same variance. Their intersection point is found when:

  $$e^{-\frac{(x-\mu_1)^2}{2\sigma^2}} = e^{-\frac{(x-\mu_2)^2}{2\sigma^2}}$$

  Taking the natural logarithm on both sides simplifies the equation, resulting in a unique x-value of intersection.

- However, if the class-conditional densities are multi-modal, or if they have different variances, the decision boundary might not be unique.

In summary, the uniqueness of the decision boundary is heavily influenced by the properties and parameters of the class-conditional densities.

Part 4:

Answer

Derivation of the MAP Estimator for μ

Step 1: Set up the Posterior Distribution

Derivation of the MAP Estimator for μ

Using Bayes theorem:

$$p(\mu|x) \propto p(x|\mu)p(\mu)$$

Step 2: Combine the Likelihood and the Prior Given the prior and likelihood, the unnormalized posterior is:

$$p(\mu|x) \propto \left(\exp\left(-\frac{1}{2\sigma^2}\sum_{k=1}^{N}(x_k-\mu)^2\right)\right) \times \mu \exp(-\mu^2/2\sigma_\mu^2)$$

Step 3: Taking the Logarithm To find the mode of the posterior (MAP estimate), we differentiate the log posterior:

$$\ln(p(\mu|x)) \propto -\frac{1}{2\sigma^2}\sum_{k=1}^{N}(x_k-\mu)^2 + \ln(\mu) - \frac{\mu^2}{2\sigma_\mu^2}$$

Step 4: Differentiate w.r.t. μ and set to zero Differentiating, we have:

$$\frac{d}{d\mu}\ln(p(\mu|x)) = \frac{N\mu - \sum_{k=1}^{N} x_k}{\sigma^2} + \frac{1}{\mu} - \frac{\mu}{\sigma_\mu^2}$$

Setting this to zero and solving for μ, we derive the MAP estimator:

$$\mu_{MAP} = \frac{Z}{2R}\left(1+\sqrt{1+\frac{4R}{Z^2}}\right)$$

where

$$Z = \frac{1}{\sigma^2}\sum_{k=1}^{N} x_k$$

and

$$R = \frac{N}{\sigma^2} + \frac{1}{\sigma_\mu^2}$$

Part 5:

Given the probability density function:

$$f_Y(y|\theta) = \begin{cases} \frac{1}{\theta^r} y^{r-1} e^{-\frac{y^r}{\theta}}, & \theta > 0, y > 0 \\ 0, & \text{elsewhere} \end{cases}$$

where r is a constant.
a) Derive the log likelihood function for this distribution.
b) Explain under what circumstances and why the MAP estimator converges to the ML estimator.

Answer

---

a) Deriving the Log Likelihood Function

1. Given the observed data points $y_1, y_2, \ldots, y_n$, the likelihood function is given by:

$$L(\theta) = \prod_{i=1}^{n} f_Y(y_i|\theta)$$

2. Substituting the given pdf into our likelihood expression:

$$L(\theta) = \prod_{i=1}^{n} \frac{1}{\theta^r} y_i^{r-1} e^{-\frac{y_i^r}{\theta}}$$

$$= \frac{1}{\theta^{rn}} \prod_{i=1}^{n} y_i^{r-1} e^{-\frac{y_i^r}{\theta}}$$

3. Taking the natural logarithm of the likelihood function:

$$l(\theta) = \ln(L(\theta))$$

$$= -rn\ln(\theta) + (r-1)\sum_{i=1}^{n} \ln(y_i) - \sum_{i=1}^{n} \frac{y_i^r}{\theta}$$

b) Convergence of MAP to ML

1. The MAP estimator is:

$$\hat{\theta}_{MAP} = \arg\max_{\theta} p(\theta|y)$$

   While the ML estimator is:

$$\hat{\theta}_{ML} = \arg\max_{\theta} p(y|\theta)$$

2. Using Bayes' theorem:

$$p(\theta|y) = \frac{p(y|\theta) \times p(\theta)}{p(y)}$$

3. The MAP estimator in terms of likelihood and prior:

$$\hat{\theta}_{MAP} = \arg\max_{\theta} p(y|\theta) \times p(\theta)$$

4. If the prior distribution $p(\theta)$ is uniform, then:

$$\hat{\theta}_{MAP} = \arg\max_{\theta} p(y|\theta)$$

   which is the same as the ML estimator. This is because when we lack a specific prior belief (uniform prior), our estimation is solely based on the likelihood, making MAP identical to ML.

Part 6

13 Naive Bayes Classifier Explanation

13.1 What is the Naive Bayes Classifier?

The Naive Bayes classifier is a probabilistic machine learning model utilized for classification tasks. This model is fundamentally based on the Bayes theorem. It shines particularly when dealing with an extremely large number of features and is frequently employed in text analysis.

13.2 Why is it called "Naive"?

The term "naive" in Naive Bayes stems from the assumption that the features in a dataset are mutually independent. This is a "naive" presumption since, in real-world scenarios, features might be interdependent. This forms the structural difference from a full Bayesian classifier, which doesn't make this assumption.

13.3 Advantages of Naive Bayes

- Simple and swift.
- Efficient even with missing attributes.
- Outperforms especially with categorical input variables as opposed to numerical ones.

13.4 Disadvantages of Naive Bayes

The core disadvantage is the assumption of independent features. In many real-world tasks, features can be interrelated, which can lead to diminished model accuracy.

13.5 When is it suitable to use Naive Bayes?

Naive Bayes is apt for:

- Multi-class problems.
- Text classification tasks such as spam detection and sentiment analysis.
- Large datasets where features are independent.

14 Data Preprocessing

14.1 Inspection

Upon initial inspection of the Wisconsin Breast Cancer dataset:

- The "id" column is a unique identifier for each record and is unlikely to have predictive power.
- The "diagnosis" column is our target, where "M" indicates Malignant and "B" indicates Benign.
- The column "Unnamed: 32" with NaN values seems to be redundant and should be removed.
- The remaining columns are features detailing the characteristics of the cell nuclei present in the breast cancer biopsies.

14.2 Preprocessing Steps Implemented

1. Removed unnecessary columns: "id" and "Unnamed: 32".
2. Encoded the "diagnosis" column, converting "M" to 1 and "B" to 0.
3. Verified that there are no missing values in the dataset.

The purpose of this question is to learn and implement the Naïve Bayes classifier.

part a

First, explain about the naive bayes class and its structural difference Describe a classification.

Answer 6.a

Introduction

The Naive Bayes classifier is a probabilistic classifier based on Bayes' theorem, which is particularly suited for high-dimensional datasets. It makes a simplifying (naive) assumption that each feature is independent of the others, given the class variable. This assumption simplifies the computation, and that's why it's called "naive".

Bayes' Theorem

Bayes' theorem describes the probability of an event based on prior knowledge of conditions related to the event. It is mathematically expressed as:

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

Where:

- $P(A|B)$ is the posterior probability of class A given predictor B.
- $P(B|A)$ is the likelihood which is the probability of predictor B given class A.
- $P(A)$ is the prior probability of class A.
- $P(B)$ is the prior probability of predictor B.

Why Naive Bayes Over Bayes?

The naive assumption of feature independence in the Naive Bayes classifier significantly simplifies the application of Bayes' theorem. Without this assumption, we would need to calculate the values for every combination of feature values, which is computationally expensive. This naive assumption is the "fee" we pay for the simplification. Despite its naive design and oversimplified assumptions, Naive Bayes can perform surprisingly well and is particularly suitable for:

- Large datasets where the naive assumption helps in computational efficiency.
- Text classification problems like spam detection or sentiment analysis.
- Situations where the independence assumption holds or where feature dependencies cancel each other out.

Drawbacks

The main drawback of Naive Bayes is its assumption of independent predictors. In real-world scenarios, it's almost impossible to have a set of predictors that are completely independent. Therefore, the performance of Naive Bayes might be suboptimal on datasets where this assumption is strongly violated.

Delineating Structural Characteristics and Decision Rationale

Distinctive Structural Features

While both the Naïve Bayes classifier and its Bayesian counterpart draw their foundations from Bayes' theorem, they diverge significantly in their treatment of features. The Naïve Bayes classifier makes a bold assumption: every feature is independent of all other features. This independence, though a simplification, bestows the model with computational efficiency, making it especially suitable for high-dimensional datasets.

Motivations for Naïve Bayes Adoption

The reasons for gravitating towards the Naïve Bayes classifier are multifold:

- **Computational Efficiency**: The model's independence assumption pares down computational overheads, particularly salient for high-dimensional datasets.
- **Empirical Proficiency**: Despite its foundational simplicity, the Naïve Bayes classifier often matches, if not outperforms, more intricate models in specific domains, notably text classification.
- **Scalable Design**: The linear computational complexity with respect to features renders it ideal for expansive datasets.

Trade-offs with Naïve Bayes

Adopting the Naïve Bayes classifier is not without its compromises:

- **Independence Assumption**: The model's bedrock assumption, though a boon for computation, doesn't always align with real-world data intricacies, potentially leading to suboptimal classifications.
- **Data Sparsity Issues**: When specific feature-class combinations are underrepresented, the classifier can potentially produce overconfident, yet incorrect, predictions.

---

part b

This data set contains two classes: one is the naïve bayes class From the base and without using the library Implement and use the classifier you designed , Recall and confusion matrix Check and analyze.

Answer 6.b

Naive Bayes Implementation from Scratch

Implementation

The Naive Bayes classifier was implemented without the use of external machine learning libraries. The process followed was:

1. Computed the mean, standard deviation, and class probabilities for each class in the training dataset.

2. Used the Gaussian Probability Density Function (PDF) to compute the likelihood of a data point belonging to a particular class.
3. Used Bayes' theorem to compute the posterior probabilities for each class and predicted the class with the highest probability.

The Gaussian PDF is given by:

$$f(x|\mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Where:

- x is the feature value.
- μ is the mean of the feature for a class.
- σ is the standard deviation of the feature for a class.

Evaluation

The classifier was trained on the training data and predictions were made on the test data. The results were evaluated using accuracy, recall, and a confusion matrix.
Results:

- **Accuracy:** 96.49%
- **Recall:** 93.02%
- **Confusion Matrix:**

$$\begin{matrix} 70 & 1 \\ 3 & 40 \end{matrix}$$

Where the rows of the confusion matrix represent the actual classes (Negative, Positive) and the columns represent the predicted classes (Negative, Positive).
Interpretation:

- True Negatives (TN): 70
- False Positives (FP): 1
- False Negatives (FN): 3
- True Positives (TP): 40

x‘x Compare the results of two sections using the SKLEARN library

Answer 6.c

15 Naive Bayes Implementation using SKLEARN

15.1 Implementation

The Naive Bayes classifier was implemented using the GaussianNB module from the ‘SKLEARN‘ library, which provides tools for constructing a Gaussian Naive Bayes classifier.

1. Initialized a GaussianNB classifier.
2. Trained the classifier using the training data.
3. Predicted class labels for the test data.

15.2 Evaluation

The predictions made by the classifier on the test data were evaluated using accuracy, recall, and a confusion matrix. **Results:**

- **Accuracy:** 96.49%
- **Recall:** 93.02%
- **Confusion Matrix:**

$$\begin{matrix} 70 & 1 \\ 3 & 40 \end{matrix}$$

Where the rows of the confusion matrix represent the actual classes (Negative, Positive) and the columns represent the predicted classes (Negative, Positive). **Interpretation:**

- True Negatives (TN): 70
- False Positives (FP): 1
- False Negatives (FN): 3
- True Positives (TP): 40

15.3 Comparison with From-Scratch Implementation

Upon comparing the results from the from-scratch implementation and the ‘SKLEARN‘ implementation, it was observed that both methods produced identical results. This demonstrates the accuracy and reliability of the custom implementation in comparison to a widely used machine learning library.

Part 7:

Image Classification: Sea vs Jungle

Problem Statement

Design a two-class classifier to distinguish between images related to the sea and the jungle in the dataset. Implement the algorithm on the data and report accuracy, confusion matrix, precision, and recall. Use features like color for separation. Identify and describe any misclassifications.

Answer

Step 1: Examining the Images

We list the extracted files and observe that the images related to the sea are prefixed with ”s” and those related to the jungle with ”j”.

Step 2: Displaying Sample Images

To understand the features of the images, we display a few of them.

Figure 8: Sample images from the dataset showing sea and jungle pictures.

Step 3: Classifying the Images

Based on the observation that sea images have more blue and jungle images have more green, we classify them using the average colors.

```
def classify_image(image_path):
    img = mpimg.imread(image_path)
    avg_green = img[:, :, 1].mean()
    avg_blue = img[:, :, 2].mean()
    return 's' if avg_blue > avg_green else 'j'
```

Step 4: Evaluating the Classifier

We use metrics like accuracy, confusion matrix, precision, and recall to evaluate our classifier.

```
accuracy = accuracy_score(true_labels, predicted_labels)
confusion = confusion_matrix(true_labels, predicted_labels, labels=['s', 'j'])
precision = precision_score(true_labels, predicted_labels, pos_label='s')
recall = recall_score(true_labels, predicted_labels, pos_label='s')
```

The results are as follows:

$$\begin{aligned} \text{Accuracy} &: 96.34\% \\ \text{Confusion Matrix} &: \begin{bmatrix} 39 & 1 \\ 2 & 40 \end{bmatrix} \\ \text{Precision (for sea)} &: 95.12\% \\ \text{Recall (for sea)} &: 97.5\% \end{aligned}$$

Step 5: Displaying Misclassified Images

Lastly, we identify and display the images that were misclassified to understand the mistakes.

```
misclassified_indices = [i for i, (true, pred) in enumerate(zip(true_labels, predicted_lab
```

The misclassified images are: ‘j44.jpg‘, ‘j45.jpg‘, and ‘s24.jpg‘.

Conclusion

The classifier shows a high level of accuracy. The misclassifications are logical given the features used for classification.