UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
ELE0606 - TÓPICOS ESPECIAIS EM INTELIGÊNCIA ARTIFICIAL

Docente:
José Alfredo Ferreira Costa

Autor:
Vinícius Venceslau Venancio da Penha

Métodos de Classificação

Natal - RN
15 de outubro de 2023

RESUMO

Este trabalho explora a Regra de Bayes, uma técnica estatística fundamental com aplicações em estatística, aprendizado de máquina e inteligência artificial. A Regra de Bayes permite o cálculo de probabilidades condicionais, atualizando crenças com base em novas evidências. O documento aborda os princípios da Regra de Bayes e fornece um exemplo intuitivo.

Ademais, descreve a aplicação da Regra de Bayes no Classificador Bayesiano, com foco em Estimação Paramétrica e Estimação Não-Paramétrica. O trabalho também introduz o Naive Bayes, um método de classificação baseado na suposição de independência condicional entre variáveis preditoras, e apresenta exemplos de sua aplicação em duas bases de dados: "Wine"e "Heart Disease."

Infere-se, portanto, que o trabalho fornece códigos em Python, detalha a organização e documentação adequada do código, e conclui com uma visão geral dos tipos comuns de Naive Bayes.

Sumário

1 Regra de Bayes

A Lei de Bayes, em homenagem ao matemático Thomas Bayes, é uma técnica estatística fundamental com amplas aplicações em áreas como estatística, aprendizado de máquina e inteligência artificial.

Esse método fornece uma poderosa maneira de calcular probabilidades condicionais, permitindo que crenças sejam atualizadas com base em novas evidências. Nesse viés, o teorema desempenha um papel central em muitos campos, incluindo classificação, diagnóstico médico, filtragem de spam e reconhecimento de padrões.

1.1 Princípios

A Regra de Bayes é fundamentada em princípios estatísticos sólidos e é expressa na seguinte equação:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \tag{1}$$

Nesse sentido, têm-se:

- $P(A|B)$ é a probabilidade posterior do evento A ocorrer, dadas as evidências B.
- $P(B|A)$ é a probabilidade da evidência B ocorrer, supondo que o evento A seja verdadeiro.
- $P(A)$ é a probabilidade prévia do evento A, antes de considerar as evidências B.
- $P(B)$ é a probabilidade da evidência B ocorrer, independentemente do evento A.

Diante disso, a seguir, será exibido um exemplo intuitivo a respeito da metologia descrita anteriormente.

Exemplo Intuitivo:

Suponha a existência de um teste médico para uma doença rara, e deseja-se calcular a probabilidade de uma pessoa realmente ter a doença, dado que o teste foi positivo.

Nesse cenário, $P(A|B)$ representaria a probabilidade da pessoa ter a doença após um resultado positivo, $P(B|A)$ seria a probabilidade do teste ser positivo se a pessoa tiver a doença, $P(A)$ seria a probabilidade prévia da pessoa ter a doença e $P(B)$ seria a probabilidade de um resultado positivo, independentemente da presença da doença.

Além disso, com o intuito de produzir uma melhor visualização e/ou simplificação matemática do método em análise, pode-se expressar $P(A|B)$ da seguinte maneira:

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

Uma vez que $P(B|A) \cdot P(A)$ é igual a interseção entre os conjuntos A e B, ou seja, $P(A \cap B)$.

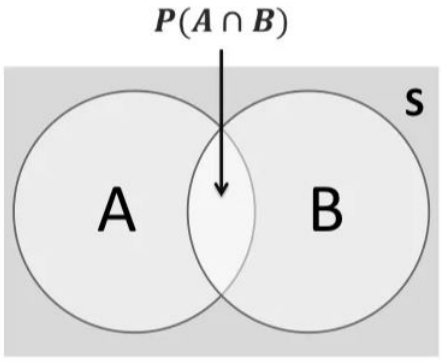

Figura 1: Interseção entre dois conjuntos.

Adicionalmente, o **Classificador Bayesiano** é uma técnica de classificação de dados amplamente utilizada no campo da ciência de dados. Ele se baseia na Teoria da Decisão Bayesiana, que é uma abordagem estatística para a classificação de padrões.

Diante disso, a principal ideia por trás do Classificador Bayesiano é usar a Regra de Bayes para calcular a probabilidade de um determinado ponto de dados pertencer a uma classe específica com base nas informações disponíveis.

Em suma, a abordagem Bayesiana considera a probabilidade condicional das classes, dadas as observações (dados). Ele calcula a probabilidade de que uma determinada instância de dados pertença a uma classe específica, considerando as probabilidades prévias das classes e as probabilidades das observações condicionadas a cada classe.

Estimação Paramétrica:

A classificação Bayesiana pode ser realizada de duas maneiras, e uma delas é a **Estimação Paramétrica**. Neste método, avalia-se o formato da distribuição dos dados em observação. O modelo de disposição dos dados pode seguir distribuições como Normal, Weibull, Exponencial, entre outras.

Portanto, na Estimação Paramétrica, o primeiro passo é determinar o padrão de distribuição mais adequado para os dados. Isso é feito através da análise exploratória dos dados e testes estatísticos. Uma vez que o modelo de distribuição é selecionado, o próximo passo é estimar os parâmetros dessa distribuição. Os parâmetros são valores que definem a forma da distribuição, como a média e o desvio padrão em uma distribuição normal.

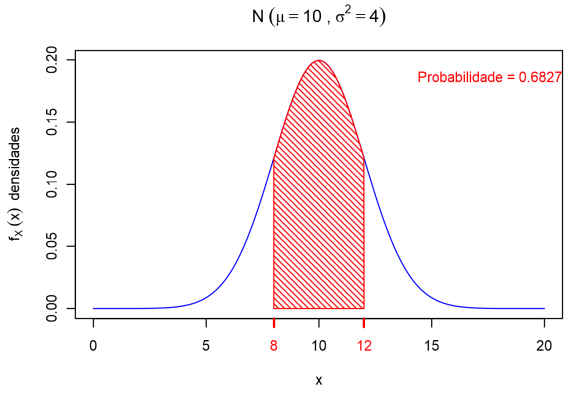

Figura 2: Exemplo de distribuição Normal.

Estimação Não-Paramétrica:

A outra forma de classificação é a **Estimação Não-Paramétrica**. Neste caso, o Classificador Bayesiano estima as probabilidades diretamente dos dados de treinamento, sem assumir um modelo de distribuição específico.

Esse método é útil quando os dados são complexos e não podem ser facilmente modelados por uma distribuição paramétrica. Outrossim, a Estimação Não-Paramétrica é mais flexível, pois não restringe a forma da distribuição dos dados, permitindo que o classificador se adapte a uma variedade de situações.

Por fim, ambos os métodos têm suas vantagens e desvantagens, e a escolha entre eles depende da natureza dos dados e das suposições que pode-se fazer sobre a distribuição subjacente.

Por conseguinte, em estatística e análise de dados, o termo "kernel"refere-se a uma função utilizada na estimação não paramétrica de densidade de probabilidade ou na suavização de dados.

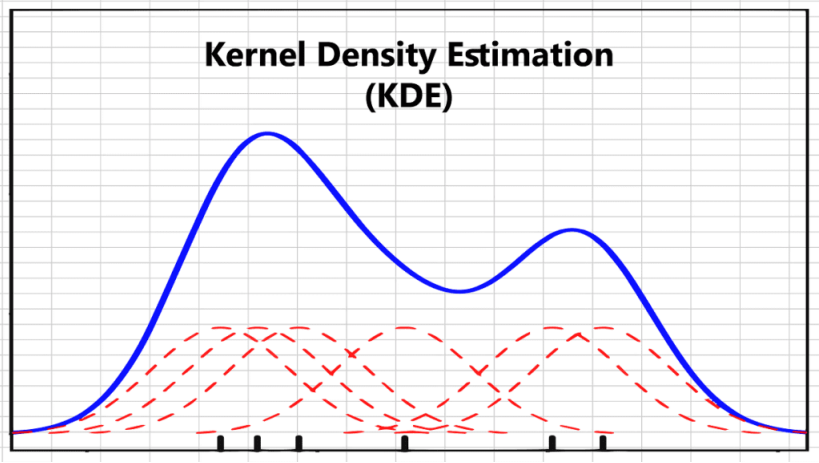

Figura 3: Ilustração da função Kernel.

1.2 Simulação

Nas subseções abaixo, serão abordadas as implementações práticas dos princípios previamente discutidos neste relatório. Por meio da utilização de código Python, será ilustrada a aplicação efetiva desses conceitos em cenários do mundo real. A simulação proporcionará uma exposição concreta das teorias e estratégias discutidas, demonstrando de forma palpável a sua relevância e aplicabilidade.

Nessa conjuntura, vale salientar que todos os códigos desenvolvidos pelo discente, encontram-se disponibilizados na plataforma do Github no repositório adequado, uma vez que se tem o intuito de facilitar a visualização dos programas produzidos.

Por fim, todo material de apoio usado para confeccionar os códigos estão expostos na referência deste documento.

1.2.1 Base de Dados - Wine

```
import pandas as pd
import numpy as np
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from scipy.stats import multivariate_normal
from sklearn.metrics import accuracy_score, classification_report,
    confusion_matrix

#Carregar o conjunto de dados Wine:
wine_data = load_wine()
wine_df = pd.DataFrame(wine_data['data'], columns=wine_data['
    feature_names'])
wine_df['classe'] = wine_data['target']  # Adicionar a coluna de classe

#Preparar os dados:
X = wine_df.drop('classe', axis=1).to_numpy()
y = wine_df['classe'].to_numpy()

#Normalizar os dados:
scaler = StandardScaler()
X = scaler.fit_transform(X)

#Dividir os dados em conjuntos de treinamento e teste:
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
     random_state=42)

#Inicializar a matriz de probabilidades:
num_classes = len(np.unique(y))
probabilidadeDaClasse = np.zeros(num_classes)

for i in range(num_classes):
    indiceDaClasse = np.where(y_train == i)
    probabilidadeDaClasse[i] = len(indiceDaClasse[0]) / len(y_train)

#Matriz para armazenar as probabilidades de classe para cada observacao
    de teste:
matrizProbabilidade = np.zeros((X_test.shape[0], num_classes))

#Calcular as probabilidades de classe para as observacoes de teste:
for i in range(num_classes):
    indiceDaClasse = np.where(y_train == i)
    dadosDeTreino = X_train[indiceDaClasse]
    media = np.mean(dadosDeTreino, axis=0)
    covariancia = np.cov(dadosDeTreino, rowvar=False)

    for j in range(X_test.shape[0]):
        observation = X_test[j]
        matrizProbabilidade[j, i] = multivariate_normal.pdf(observation,
    mean=media, cov=covariancia, allow_singular=True) *
    probabilidadeDaClasse[i]

#Prever as classes com base nas probabilidades:
y_pred = np.argmax(matrizProbabilidade, axis=1)

#Calcular a acuracia das previsoes:
```

```
accuracy = accuracy_score(y_test, y_pred)
print('Acuracia:', accuracy)

#Gerar Matriz Confusao:
confusion = confusion_matrix(y_test, y_pred)
report = classification_report(y_test, y_pred, target_names=['Classe 0',
    'Classe 1', 'Classe 2'])

print("\nMatriz de Confusao:\n", confusion)
print("\nRelatorio de Classificacao:\n", report)
```

Listing 1: Classificador Bayesiano - Wine.

```
Acuracia: 0.9722

Matriz de Confusao:
 [[14  0  0]
 [ 0 14  0]
 [ 0  1  7]]

Relatorio de Classificacao:
               precision    recall  f1-score   support

    Classe 0       1.00      1.00      1.00        14
    Classe 1       0.93      1.00      0.97        14
    Classe 2       1.00      0.88      0.93         8

    accuracy                           0.97        36
   macro avg       0.98      0.96      0.97        36
weighted avg       0.97      0.97      0.97        36
```

Listing 2: Saída 1.

1.2.2 Base de Dados - Heart Disease

```
#Permitir o google colab acessar os arquivos do Drive:
from google.colab import drive
drive.mount('/content/drive')

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from scipy.stats import multivariate_normal
from sklearn.metrics import accuracy_score, classification_report,
    confusion_matrix

#Carregamento da base de dados a partir do Google Drive:
caminho_arquivo = "/content/drive/My Drive/heart.csv"
banco_de_dados = pd.read_csv(caminho_arquivo)

#Criar um Dataframe:
heart_df = banco_de_dados

#Preparar os dados:
X = heart_df.drop('target', axis=1).to_numpy()
y = heart_df['target'].to_numpy()

#Normalizar os dados:
scaler = StandardScaler()
X = scaler.fit_transform(X)

#Dividir os dados em conjuntos de treinamento e teste:
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

#Inicializar a matriz de probabilidades:
num_classes = len(np.unique(y))
probabilidadeDaClasse = np.zeros(num_classes)

for i in range(num_classes):
    indiceDaClasse = np.where(y_train == i)
    probabilidadeDaClasse[i] = len(indiceDaClasse[0]) / len(y_train)

#Matriz para armazenar as probabilidades de classe para cada observacao
    de teste:
matrizProbabilidade = np.zeros((X_test.shape[0], num_classes))

#Calcular as probabilidades de classe para as observacoes de teste:
for i in range(num_classes):
    indiceDaClasse = np.where(y_train == i)
    dadosDeTreino = X_train[indiceDaClasse]
    media = np.mean(dadosDeTreino, axis=0)
    covariancia = np.cov(dadosDeTreino, rowvar=False)

    for j in range(X_test.shape[0]):
        observation = X_test[j]
        matrizProbabilidade[j, i] = multivariate_normal.pdf(observation,
    mean=media, cov=covariancia, allow_singular=True) *
   probabilidadeDaClasse[i]

#Prever as classes com base nas probabilidades:
```

```
y_pred = np.argmax(matrizProbabilidade, axis=1)

#Calcular a acuracia das previsoes:
accuracy = accuracy_score(y_test, y_pred)
print(f'Acuracia: {accuracy:.4f}')

#Gerar Matriz Confusao:
confusion = confusion_matrix(y_test, y_pred)
report = classification_report(y_test, y_pred, target_names=['Classe 0',
    'Classe 1'])

print("\nMatriz de Confusao:\n", confusion)
print("\nRelatorio de Classificacao:\n", report)
```

Listing 3: Classificador Bayesiano - Heart Disease.

```
Drive already mounted at /content/drive; to attempt to forcibly remount,
    call drive.mount("/content/drive", force_remount=True).
Acuracia: 0.8634

Matriz de Confusao:
 [[85 21]
 [ 7 92]]

Relatorio de Classificacao:
               precision    recall  f1-score   support

    Classe 0       0.92      0.80      0.86       106
    Classe 1       0.81      0.93      0.87        99

    accuracy                           0.86       205
   macro avg       0.87      0.87      0.86       205
weighted avg       0.87      0.86      0.86       205
```

Listing 4: Saída 2.

2 Regra Simplificada - Naive Bayes

2.1 Princípios

Naive Bayes é frequentemente usado para tarefas de classificação, onde o objetivo é prever a classe de um objeto ou evento com base em informações disponíveis. A fórmula geral para classificação Bayesiana é:

$$P(C_k|X) = \frac{P(X|C_k) \cdot P(C_k)}{P(X)} \tag{2}$$

Onde:

- $P(C_k|X)$ é a probabilidade da classe C_k dado um conjunto de características X.
- $P(X|C_k)$ é a probabilidade de observar X dado que a classe é C_k.
- $P(C_k)$ é a probabilidade prévia da classe C_k.
- $P(X)$ é a probabilidade de observar X.

O termo "Naive"em Naive Bayes se refere à suposição de independência condicional entre as variáveis preditoras (características) dadas as classes.

Dessa forma, isso significa que a presença ou valor de uma característica não depende das outras características quando a classe é conhecida. Essa suposição simplifica o cálculo das probabilidades, tornando-o mais viável computacionalmente.

Existem vários tipos de Naive Bayes, sendo os mais comuns:

1. **Naive Bayes Gaussiano:** Assume que as características seguem uma distribuição normal (Gaussiana).
2. **Naive Bayes Multinomial:** Usado para dados discretos, como contagens de palavras em um documento. É amplamente usado em classificação de texto.
3. **Naive Bayes Bernoulli:** Usado para características binárias, onde cada característica é representada como uma variável booleana.

2.2 Simulação

Os códigos em Python estão sendo apresentados, com ênfase na organização da formatação, utilização de comentários esclarecedores e adoção de nomes descritivos para variáveis, juntamente com a estruturação do código em blocos lógicos por meio de funções e classes para facilitar a manutenção e compreensão.

Assim, a documentação adequada desempenha um papel importante na promoção da colaboração eficaz e na comunicação clara de soluções e resultados em projetos Python.

2.2.1 Base de Dados - Wine

```python
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

wine = load_wine()
X = wine.data
y = wine.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=13)

nb_classifier = GaussianNB()
nb_classifier.fit(X_train, y_train)

y_pred = nb_classifier.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
confusion = confusion_matrix(y_test, y_pred)
report = classification_report(y_test, y_pred, target_names=wine.target_names)

print("Acuracia:", accuracy)
print("\nMatriz de Confusao:\n", confusion)
print("\nRelatorio de Classificacao:\n", report)
```

Listing 5: Código Naive Bayes - Wine.

```
Acuracia: 0.9861111111111112

Matriz de Confusao:
 [[24  0  0]
 [ 0 30  1]
 [ 0  0 17]]

Relatorio de Classificacao:
               precision    recall  f1-score   support

     class_0       1.00      1.00      1.00        24
     class_1       1.00      0.97      0.98        31
     class_2       0.94      1.00      0.97        17

    accuracy                           0.99        72
   macro avg       0.98      0.99      0.99        72
weighted avg       0.99      0.99      0.99        72
```

Listing 6: Saída 3.

2.2.2 Base de Dados - Heart Disease

```
#Permitir o google colab acessar os arquivos do Drive:
from google.colab import drive
drive.mount('/content/drive')

#Importar bibliotecas:
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report,
    confusion_matrix

#Carregamento da base de dados a partir do Google Drive:
caminho_arquivo = "/content/drive/My Drive/heart.csv"
banco_de_dados = pd.read_csv(caminho_arquivo)

#Criar um Dataframe:
heart_df = banco_de_dados

#Armazenamento dos rotulos de classe em uma variavel:
heart_classe = heart_df['target']

#Remocao da coluna 'target' do DataFrame:
heart_df.drop(['target'], axis=1, inplace=True)

#Divisao dos dados em conjuntos de treinamento e teste:
X_train, X_test, y_train, y_test = train_test_split(heart_df,
    heart_classe, test_size=0.3, random_state=13)

#Criar um classificador Naive Bayes Gaussiano:
nb_classifier = GaussianNB()

#Treinar o classificador:
nb_classifier.fit(X_train, y_train)

#Fazer previsoes no conjunto de teste:
y_pred = nb_classifier.predict(X_test)

#Avaliar o desempenho do classificador:
accuracy = accuracy_score(y_test, y_pred)
confusion = confusion_matrix(y_test, y_pred)
report = classification_report(y_test, y_pred, target_names=['Classe 0',
    'Classe 1'])

print("Acuracia:", accuracy)
print("\nMatriz de Confusao:\n", confusion)
print("\nRelatorio de Classificacao:\n", report)
```

Listing 7: Código Naive Bayes - Heart Disease.

```
Drive already mounted at /content/drive; to attempt to forcibly remount,
    call drive.mount("/content/drive", force_remount=True).
Acuracia: 0.814935064935065

Matriz de Confusao:
 [[113  39]
 [ 18 138]]

Relatorio de Classificacao:
               precision    recall  f1-score   support

    Classe 0       0.86      0.74      0.80       152
    Classe 1       0.78      0.88      0.83       156

    accuracy                           0.81       308
   macro avg       0.82      0.81      0.81       308
weighted avg       0.82      0.81      0.81       308
```

Listing 8: Saída 4.

3 Conclusão

A Regra de Bayes é uma ferramenta valiosa que fornece uma estrutura sólida para calcular probabilidades condicionais em diversas aplicações. Ela é fundamental para entender e implementar algoritmos de aprendizado de máquina, como o Naive Bayes, que se baseiam em princípios bayesianos.

Além disso, a Regra de Bayes é uma das bases da estatística e tem sido amplamente aplicada em diagnóstico médico, previsão de eventos e muito mais. É importante compreender o seu funcionamento e aplicabilidade em diversas situações para utilizar efetivamente esse poderoso método estatístico e de aprendizado de máquina.

4 Referências

LUIZ, E.; OLIVEIRA. **Reconhecimento de Padrões Teoria da Decisão Bayesiana**. [s.l: s.n.]. Disponível em: https://www.inf.ufpr.br/lesoliveira/padroes/decisaobayesiana.pdf. Acesso em: 5 de outubro de 2023.

Ciência de Dados: Teoria da Decisão Bayesiana na classificação de dados. Disponível em: https://www.youtube.com/watch?v=8zAKWEOdGsg. Acesso em: 5 de outubro de 2023.

Data Mining Classification: Alternative Techniques. Disponível em: https://www-users.cs.umn.edu/~kumar001/dmbook/slides/chap4_naive_bayes.pptx. Acesso em: 5 de outubro de 2023.

OLIVEIRA, L. **Teorema de Bayes e Probabilidade**. Disponível em: https://medium.com/data-hackers/teorema-de-bayes-probabilidade-d5ead2df1379. Acesso em: 14 de outubro de 2023.