Universidade Federal do Rio de Janeiro
Instituto Alberto Luiz Coimbra de
Pós-Graduação e Pesquisa de Engenharia

Programa de Engenharia de Sistemas e Computação

CPS863 - Aprendizado de Máquina
Prof. Dr. Edmundo de Souza e Silva
(PESC/COPPE/UFRJ)

Lista de Exercícios 3

Luiz Henrique Souza Caldas
email: lhscaldas@cos.ufrj.br

13 de novembro de 2024

Questão 1

Neste trabalho, você irá ajustar e avaliar três modelos diferentes em um conjunto de dados com três features:

1. **GaussI**: Um modelo de mistura de Gaussianas (GMM) com uma Gaussiana por classe, onde as matrizes de covariância são todas iguais à matriz identidade, i.e., $p(x|y = c) = N(x|\mu_c, I)$.
2. **GaussX**: Um modelo de mistura de Gaussianas (GMM) com uma Gaussiana por classe, sem restrições nas matrizes de covariância, i.e., $p(x|y = c) = N(x|\mu_c, \Sigma_c)$.
3. **LogReg**: Um modelo de regressão logística com características lineares e quadráticas, i.e., função polinomial de grau 2.

A questão inclui dois conjuntos de dados: um conjunto de treino e um conjunto de teste. Cada amostra possui três features. Siga os passos a seguir para cada modelo:

1. Calcule a log-likelihood, de forma literal para cada modelo (explique como calcular os parâmetros de cada).

Resposta:

- **Modelo GaussI: Mistura de Gaussianas com Matriz Identidade**

  No modelo GaussI, todas as classes c são representadas por uma gaussiana multivariada com média μ_c e matriz de covariância igual à identidade I. Sendo uma mistura de gaussianas, o peso π_c da classe c representa a proporção de amostras pertencentes àquela classe na mistura. A função de densidade de probabilidade para uma amostra x é dada por:

  $$p(x) = \sum_c \pi_c \, N(x|\mu_c, I) = \sum_c \pi_c \, \frac{1}{(2\pi)^{d/2}} \exp\left(-\frac{1}{2}(x-\mu_c)^T(x-\mu_c)\right)$$

  Cálculo da Log-Likelihood: Dada uma amostra $X = \{x_1, x_2, \ldots, x_n\}$, a log-likelihood do modelo é:

  $$\mathcal{L}(\theta|X) = \sum_{i=1}^{n} \log\left(\sum_c \pi_c \, p(x_i|y=c)\right)$$

  Substituindo $p(x_i|y=c)$:

  $$\mathcal{L}(\theta|X) = \sum_{i=1}^{n} \log\left(\sum_c \pi_c \, \frac{1}{(2\pi)^{d/2}} \exp\left(-\frac{1}{2}(x_i-\mu_c)^T(x_i-\mu_c)\right)\right)$$

  Cálculo dos Parâmetros: Utilizando o Maximum Likelihood Estimation (MLE), os parâmetros do modelo são ajustados derivando-se a log-likelihood em relação a μ_c e π_c e igualando a zero. Para cada classe c:

  - A média μ_c é estimada pela média das amostras de treino da classe c:

    $$\mu_c = \frac{1}{n_c}\sum_{i=1}^{n_c} x_i$$

  - O peso π_c é estimado como a proporção de amostras da classe c no conjunto de treino:

    $$\pi_c = \frac{n_c}{n}$$

    onde n_c é o número de amostras da classe c e n é o total de amostras.

Resposta (continuação):

- **Modelo GaussX: Mistura de Gaussianas sem Restrições na Covariância**

  Neste modelo, cada classe c é representada por uma gaussiana com média μ_c e matriz de covariância Σ_c. Assim, a função de densidade de probabilidade é:

  $$p(x) = \sum_c \pi_c \, N(x|\mu_c, \Sigma_c) = \sum_c \pi_c \, \frac{1}{(2\pi)^{d/2}|\Sigma_c|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_c)^T \Sigma_c^{-1}(x-\mu_c)\right)$$

  Cálculo da Log-Likelihood: A log-likelihood para o modelo é:

  $$\mathcal{L}(\theta|X) = \sum_{i=1}^{n} \log\left(\sum_c \pi_c \, p(x_i|y=c)\right)$$

  Substituindo $p(x_i|y=c)$:

  $$\mathcal{L}(\theta|X) = \sum_{i=1}^{n} \log\left(\sum_c \pi_c \, \frac{1}{(2\pi)^{d/2}|\Sigma_c|^{1/2}} \exp\left(-\frac{1}{2}(x_i-\mu_c)^T \Sigma_c^{-1}(x_i-\mu_c)\right)\right)$$

  Cálculo dos Parâmetros: Utilizando também o MLE, os parâmetros do modelo são ajustados derivando-se a log-likelihood em relação a μ_c, Σ_c e π_c e igualando a zero. Para cada classe c:

  - A média μ_c é calculada pela média das amostras de treino da classe c:

    $$\mu_c = \frac{1}{n_c} \sum_{i=1}^{n_c} x_i$$

  - A matriz de covariância Σ_c é estimada por:

    $$\Sigma_c = \frac{1}{n_c} \sum_{i=1}^{n_c} (x_i-\mu_c)(x_i-\mu_c)^T$$

  - O peso π_c é dado pela proporção de amostras da classe c:

    $$\pi_c = \frac{n_c}{n}$$

Resposta (continuação):

- **Modelo LogReg: Regressão Logística com Features Lineares e Quadráticas**

  Para o modelo de regressão logística, definimos uma função de probabilidade que modela a probabilidade de $y = 1$ ou $y = 2$ como uma função logística das features lineares e quadráticas.

  A probabilidade de uma amostra x pertencer à classe $y = 1$ é:

  $$p(y = 1|x) = \sigma(w^T x + w_q^T x^2 + b)$$

  onde:

  - $\sigma(z) = \frac{1}{1+e^{-z}}$ é a função sigmoide,
  - w e w_q são vetores de pesos para as features lineares e quadráticas, respectivamente,
  - b é o bias (intercepto).

  Cálculo da Log-Likelihood: A log-likelihood para o modelo de regressão logística com n amostras é:

  $$\begin{aligned}\mathcal{L}(\theta|X, y) = \sum_{i=1}^{n} \Big(&\delta_{y_i,1} \log \sigma(w^\top x_i + w_q^\top x_i^2 + b) \\ &+\delta_{y_i,2} \log(1 - \sigma(w^\top x_i + w_q^\top x_i^2 + b))\Big)\end{aligned}$$

  onde $\delta_{y_i,1}$ e $\delta_{y_i,2}$ são funções indicadoras que valem 1 quando $y_i = 1$ e $y_i = 2$, respectivamente, e 0 caso contrário.

  Cálculo dos Parâmetros: Os parâmetros w, w_q e b são obtidos por maximização da log-likelihood, que geralmente é resolvida através de métodos de otimização numérica, como gradiente descendente ou variantes (ex.: método de Newton-Raphson).

2. Para cada modelo (GaussI, GaussX e LogReg), obtenha os parâmetros usando o conjunto de treino.

Resposta:

- **Modelo GaussI**:
  - **Média** μ_c: Calculamos a média μ_c para cada classe c somando todas as amostras da classe e dividindo pelo número total de amostras dessa classe:

    $$\mu_c = \frac{1}{n_c}\sum_{i=1}^{n_c} x_i$$

    onde n_c é o número de amostras da classe c.
  - **Peso** π_c: O peso π_c da mistura para a classe c é a proporção de amostras da classe c no conjunto de treino:

    $$\pi_c = \frac{n_c}{n}$$

    onde n é o total de amostras.
  - **Covariância**: A matriz de covariância é a identidade I, sendo fixa para todas as classes.
- **Modelo GaussX**:
  - **Média** μ_c: Calculamos a média μ_c da mesma forma que no GaussI.
  - **Peso** π_c: O peso π_c é estimado da mesma forma que no GaussI
  - **Covariância** Σ_c: Estimamos a matriz de covariância Σ_c para cada classe c como:

    $$\Sigma_c = \frac{1}{n_c}\sum_{i=1}^{n_c}(x_i - \mu_c)(x_i - \mu_c)^T$$
- **Modelo LogReg**:
  - Ajustaremos os parâmetros w, w_q, e b para maximizar a log-likelihood do modelo no conjunto de treino, onde as classes são $y = 1$ e $y = 2$. Para isso, utilizamos métodos de otimização, como o método quasi-Newton Broyden-Fletcher-Goldfarb-Shanno (BFGS).

Resposta (continuação):

- **GaussI** Média classe 1: [9.87137785, 10.10450844, 10.03867547]
  Média classe 2: [5.00268659 4.96179356 5.02277374]
  Matriz de covariância (identidade):

$$\begin{bmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}$$

  Peso da classe 1: 0.7142857142857143
  Peso da classe 2: 0.2857142857142857

- **GaussX** Média classe 1: [9.87137785, 10.10450844, 10.03867547]
  Covariância classe 1:

$$\begin{bmatrix} 1.06499084 & 0.23280792 & 0.0827585 \\ 0.23280792 & 0.97380852 & 0.31446581 \\ 0.0827585 & 0.31446581 & 1.11015715 \end{bmatrix}$$

  Média classe 2: [5.00268659, 4.96179356, 5.02277374]
  Covariância classe 2:

$$\begin{bmatrix} 1.01006542 & -0.1147753 & 0.03187256 \\ -0.1147753 & 0.90176172 & 0.18048772 \\ 0.03187256 & 0.18048772 & 0.92142339 \end{bmatrix}$$

  Peso da classe 1: 0.7142857142857143
  Peso da classe 2: 0.2857142857142857

- **LogReg** Coeficientes (w): [-369.8030754, -419.75537071, -308.97310576]
  Intercepto (b): 8620.82826196496

3. Para cada modelo (GaussI, GaussX e LogReg), calcule a log-likelihood usando o conjunto de teste e os parâmetros obtidos.

Resposta:

Foram utilizadas, em um código Python, as fórmulas de log-likelihood deduzidas no item 1 para calcular o valor da log-likelihood de cada modelo no conjunto de dados de treinamento (não foi possível utilizar o conjunto de teste, pois o mesmo está sem labels). Abaixo estão os resultados obtidos para cada modelo:

- **Log-Likelihood GaussI**: -3416.1258637693318

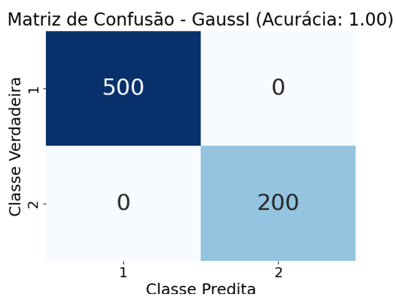
- **Log-Likelihood GaussX**: -3370.0326509880647

- **Log-Likelihood LogReg**: -6.263139306057298e-07

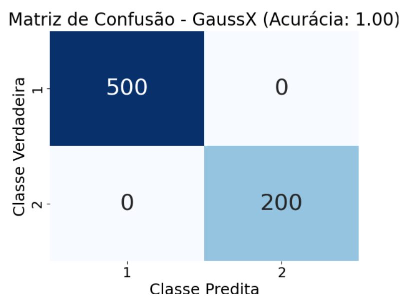
4. Avalie o desempenho de cada modelo usando o conjunto de teste e compare os resultados. Discuta qual modelo apresentou o melhor desempenho e tente dar a sua explicação sobre o motivo.

Resposta:

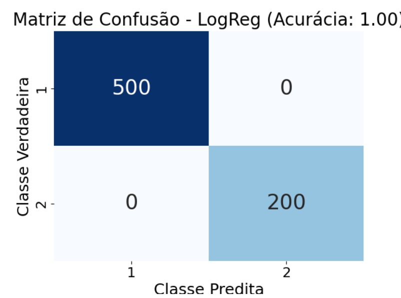
Como não foi possível utilizar o conjunto de teste, não foi possível avaliar o desempenho dos modelos. No entanto, podemos comparar os valores de log-likelihood obtidos no conjunto de treinamento. O modelo de regressão logística apresentou o maior valor de log-likelihood, indicando que ele se ajustou melhor aos dados de treinamento.
As matrizes de confusão e as acurácias obtidos no conjunto de treinamento para cada modelo podem ser visualizados na Figura 1.



(a) Matriz de confusão do modelo GaussI.



(b) Matriz de confusão do modelo GaussX.



(c) Matriz de confusão do modelo LogReg.

Figura 1: Matrizes de confusão dos modelos GaussI, GaussX e LogReg.

Questão 2

Nesta questão, você usará o classificador Naive Bayes para classificar mensagens SMS como spam ou ham (não spam) usando o conjunto de dados SMS Spam Collection. Esse conjunto de dados contém uma série de mensagens SMS etiquetadas como spam ou ham e será utilizado para treinar e avaliar o desempenho do modelo Naive Bayes.

1. Treinar um classificador Naive Bayes para classificar mensagens de texto.
2. Avaliar o desempenho do modelo em um conjunto de teste.
3. Discutir o impacto da suposição de independência do Naive Bayes e como ela afeta os resultados.

Dataset O conjunto de dados que será utilizado é o "SMS Spam Collection", disponível no Repositório de Aprendizado de Máquina da UCI. Você pode baixá-lo do link abaixo: `https://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection`

O conjunto de dados é composto por:

1. Coluna 1: A etiqueta ("spam" ou "ham").
2. Coluna 2: A mensagem SMS em texto.

Siga os passos a seguir para realizar o trabalho.

Passo 1: Preparação dos Dados:

1. Carregue o conjunto de dados e converta as etiquetas para formato binário: "ham"= 0 e "spam"= 1.

> Resposta:
>
> Para carregar o conjunto de dados e converter as etiquetas para formato binário, utilizamos o Pandas para ler o arquivo e mapeamos "ham" para 0 e "spam" para 1:
>
> df['label'] = df['label'].map({`''ham'': 0, ''spam'': 1`})

2. Divida o conjunto de dados em um conjunto de treino (70%) e um conjunto de teste (30%).

> Resposta:
>
> Para dividir o conjunto de dados em conjunto de treino (70%) e conjunto de teste (30%), utilizamos a função `train_test_split` da biblioteca `sklearn.model_selection`:
>
> $X\_train, X\_test, y\_train, y\_test =$ train_test_split(df['sms'], df['label'],
> test_size=0.3, random_state=42)

3. Utilize o modelo de bag-of-words para transformar o texto das mensagens em uma representação numérica.,

> Resposta:
>
> Para transformar o texto das mensagens em uma representação numérica, utilizamos o modelo de bag-of-words com a classe `CountVectorizer` da biblioteca `sklearn.feature_extraction.text`:
>
> $$\begin{aligned} \text{vectorizer} &= \text{CountVectorizer()} \\ X\_train\_bow &= \text{vectorizer.fit\_transform(X\_train)} \\ X\_test\_bow &= \text{vectorizer.transform(X\_test)} \end{aligned}$$
>
> O modelo bag-of-words transforma texto em uma representação numérica, criando um vetor para cada sms onde cada posição representa a frequência de uma palavra específica do vocabulário.

Passo 2: Treinamento do Modelo

1. Treine um classificador Naive Bayes multinomial usando o conjunto de treino.

> Resposta:
>
> Para treinar um classificador Naive Bayes multinomial, a classe `NaiveBayesClassifier` inicia armazenando variáveis para as probabilidades de palavras em cada classe, as probabilidades a priori das classes e o vocabulário. No método `train`, primeiro conta-se a quantidade de mensagens de cada classe e calcula-se a probabilidade a priori de cada uma (ex.: probabilidade de uma mensagem ser "ham" ou "spam"). Em seguida, o método conta quantas vezes cada palavra aparece em mensagens de cada classe e aplica suavização de Laplace para calcular as probabilidades condicionais das palavras em cada classe.

2. Use o modelo treinado para prever se as mensagens do conjunto de teste são spam ou ham.

> Resposta:
>
> O método `predict` calcula a probabilidade de uma mensagem pertencer a cada classe usando as probabilidades a priori e as probabilidades condicionais das palavras para as classes "ham" e "spam". Ele percorre cada palavra da mensagem e, se a palavra estiver no vocabulário, soma o logaritmo da probabilidade da palavra na classe correspondente à probabilidade acumulada da classe. O uso do logaritmo evita problemas de underflow, comuns ao multiplicar várias probabilidades pequenas, além de transformar o produto das probabilidades em uma soma, o que facilita o cálculo. Ao final, a classe com a maior probabilidade acumulada é atribuída como a predição da mensagem.

3. Calcule a precisão (accuracy), precisão (precision), revocação (recall) e a pontuação F1 (F1-score) para o conjunto de teste.

Resposta:

As métricas de desempenho são calculadas da seguinte forma:
- **Acurácia (Accuracy)**: proporção de previsões corretas entre todas as previsões, calculada como

$$\text{Accuracy} = \frac{\text{Verdadeiros Positivos} + \text{Verdadeiros Negativos}}{\text{Total de Amostras}}$$

- **Precisão (Precision)**: proporção de previsões positivas corretas entre todas as previsões positivas, dada por

$$\text{Precision} = \frac{\text{Verdadeiros Positivos}}{\text{Verdadeiros Positivos} + \text{Falsos Positivos}}$$

- **Revocação (Recall)**: proporção de verdadeiros positivos entre todos os casos que realmente são positivos, calculada como

$$\text{Recall} = \frac{\text{Verdadeiros Positivos}}{\text{Verdadeiros Positivos} + \text{Falsos Negativos}}$$

- **F1-score**: média harmônica entre precisão e revocação, utilizada para balancear as duas métricas:

$$\text{F1-score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Resultados:

- Acurácia: 0.9904
- Precisão: 0.9815
- Revocação: 0.9464
- F1-score: 0.9636

4. Explique como o modelo Naive Bayes classifica uma mensagem como spam ou ham. Por que o Naive Bayes pode ser eficaz mesmo assumindo independência entre as palavras?

Resposta:

O modelo Naive Bayes classifica uma mensagem como "spam" ou "ham" calculando a probabilidade de cada classe dada a mensagem, $P(\text{classe}|\text{mensagem})$. Utilizando o teorema de Bayes, essa probabilidade é calculada como proporcional a $P(\text{mensagem}|\text{classe}) \cdot P(\text{classe})$. O modelo assume independência entre as palavras, então a probabilidade condicional $P(\text{mensagem}|\text{classe})$ é a multiplicação das probabilidades individuais de cada palavra dada a classe.
Mesmo assumindo independência entre as palavras (o que geralmente não é realista, pois palavras em uma frase tendem a ser relacionadas), o Naive Bayes ainda é eficaz em muitos casos, pois as frequências das palavras nas classes "spam" e "ham" tendem a capturar padrões de linguagem característicos de cada categoria. Assim, mesmo que as palavras não sejam realmente independentes, o modelo consegue distinguir com precisão "spam" de "ham" com base em combinações de palavras típicas de cada classe.

5. Analise as métricas de avaliação (precisão, revocação, F1-score) obtidas. O modelo foi capaz de detectar bem as mensagens spam? Explique com base nas métricas.

Resposta:

As métricas de avaliação indicam que o modelo Naive Bayes foi eficaz em detectar mensagens "spam". A acurácia de 0.9904 mostra que a maioria das mensagens foi classificada corretamente. A precisão de 0.9815 indica que quase todas as mensagens classificadas como "spam" realmente eram spam, minimizando falsos positivos. A revocação de 0.9464 sugere que o modelo foi capaz de identificar uma alta proporção dos spams existentes, mas ainda deixou de classificar alguns. O F1-score de 0.9636, combinando precisão e revocação, indica um bom equilíbrio entre as duas métricas. Portanto, com base nesses valores, o modelo conseguiu detectar bem as mensagens "spam", com poucos erros de classificação.

6. O Naive Bayes faz uma suposição de independência entre as palavras da mensagem. Discuta como essa suposição pode afetar a classificação de mensagens. Por que, apesar dessa suposição, o modelo ainda pode ter uma boa performance?

Resposta:

A suposição de independência entre as palavras significa que o Naive Bayes trata cada palavra como se ela não tivesse relação com as demais no contexto da mensagem. Essa suposição é irrealista, pois palavras em uma mensagem geralmente possuem dependências contextuais. Por exemplo, em uma mensagem de spam, frases como "Free entry" e "win a prize" têm um significado conjunto que indica spam mais fortemente do que cada palavra isoladamente. Ignorar essas dependências pode fazer com que o modelo perca nuances contextuais que ajudariam na classificação.

Apesar disso, o Naive Bayes ainda pode ter boa performance, pois o padrão de ocorrência de certas palavras é característico para cada classe. Palavras como "win", "prize", "entry", e "free" aparecem frequentemente em mensagens de spam, enquanto termos mais comuns, como "Ok" e "call", tendem a ocorrer em mensagens ham. Assim, mesmo sem captar todas as relações contextuais, o modelo consegue diferenciar as classes com base nas frequências características de palavras, o que costuma ser suficiente para bons resultados.

7. Discuta um cenário em que a suposição de independência do Naive Bayes pode prejudicar significativamente a precisão do modelo.

Resposta:

Um cenário em que a suposição de independência do Naive Bayes pode prejudicar a precisão do modelo ocorre em mensagens onde o contexto entre as palavras é essencial para determinar o sentido. Por exemplo, no dataset SMSSpamCollection, mensagens que contenham sequências como "call me now" ou "urgent call" podem ter interpretações diferentes dependendo do tom e das palavras associadas.

Em "ham", expressões como "call me later" ou "can we chat tomorrow" são comuns e contextualmente neutras. Já em "spam", frases como "urgent call now" ou "win now, call today" sugerem mais urgência e intenção de induzir uma resposta imediata. Naive Bayes, assumindo independência, pode ignorar essas nuances de contexto e tratar palavras como "call" e "now" separadamente, o que reduz a precisão ao classificar mensagens ambíguas. Em mensagens onde o sentido resulta da combinação de palavras e contexto, a performance do Naive Bayes é limitada.

Codigos

Os códigos utilizados para a resolução dos exercícios estão disponíveis no repositório do GitHub: `https://github.com/lhscaldas/cps863/`