FGV EMAp
ESCOLA DE MATEMÁTICA APLICADA

Aula Prática 4

Álgebra Linear Numérica

Gerardo Mikael Do Carmo Pereira

Professor: Antônio Carlos Saraiva Branco

RIO DE JANEIRO
2024

Conteúdo

1 Introdução

O presente trabalho refere-se à aula prática 4 onde aplicamos algoritmos de regressão e classificação utilizando o método dos mínimos quadrados. As tarefas realizadas visam proporcionar um entendimento prático do funcionamento desses algoritmos. Ao final, esperamos obter classificações de dados com baixo erro e implementar melhorias nos algoritmos, resultando em uma maior acurácia e precisão.

2 Implementação

2.1 <Tarefa 1> Implementação do método de mínimos quadrados no modelo econômico de Charles Cobb e Paul Douglas

O modelo que iremos utilizar quantifica o quão bem está a economia através da produção total que se refere ao valor total dos bens produzidos no ano, usaremos a quantidade de trabalho (L) e a quantidade de capital investido K. para fazer esse cálculo. A função do modelo leva a forma:

$$P = bL^{\alpha}K^{1-\alpha}$$

b e α serão os parâmetros que serão calculados e utilizados para fazer as previsões para K e L dados.

O método dos mínimos quadrados é usado em sistemas lineares, logicamente não funcionará no modelo em questão. Faremos a linearização do modelo utilizando logaritmos e suas propriedades. Tomemos:

$$\begin{aligned} P &= bL^{\alpha}K^{1-\alpha} \\ \log P &= \log b + K \log L + (1-\alpha)\log K \end{aligned}$$

Partindo desse pressuposto podemos calcular os parâmetros com o sistema linearizado pelos logs através do método, poderemos fazer sua previsão.

O algoritmo implementado aplica o que vimos em sala de aula nos dados fornecidos.

Primeiro, definimos o modelo linear:

$$Ax \approx b$$

A função objetivo a ser minimizada é a soma dos erros quadrados:

$$S(x) = \|Ax - b\|^2$$

Expandimos a função objetivo:

$$S(x) = (Ax - b)^T(Ax - b)$$

Calculamos a derivada da função objetivo em relação a x:

$$\frac{\partial S(x)}{\partial x} = \frac{\partial}{\partial x}\left[(Ax - b)^T(Ax - b)\right]$$

Usamos a regra da cadeia:

$$\frac{\partial S(x)}{\partial x} = 2A^T(Ax - b)$$

Igualamos a derivada a zero para encontrar o ponto de mínimo:

$$2A^T(Ax - b) = 0$$

Simplificamos a equação:

$$A^T Ax^* = A^T b$$

Aplicaremos então a equação no conjunto que chamaremos de "simple_economic_model.csv", usaremos também a função "AcheOsCoeficientes.sci"para obter os coeficientes da função de produção linearizada obteremos:

```
--> data_simple_model = csvRead("simple_economic_model.csv", ";");

--> [beta, alpha, one_minus_alpha] = AcheOsCoeficientes(data_simple_model);

 "Valor de b:"

  2.49D-285

 "Valor de α:"

  120.79200

 "Valor de (1-α):"

  41.192935
```

Figura 1: Valores dos coeficientes que usaremos para a previsão de P

Com os valores obtidos pela função "AcheOsCoeficientes.sci"faremos a previsão dos anos de 1910 e 1920 com os valores de L e K previamente dados:

```
--> log_p = log(beta)+alpha*log(147)+one_minus_alpha*log(208)
 log_p  =

   167.34702

--> log_p = log(beta)+alpha*log(194)+one_minus_alpha*log(407)
 log_p  =

   228.50960
```

Figura 2: Previsões obtidas com a função linearizada

As previsões tem uma boa proximidade com os valores reais, podemos afirmar que o método tem um bom funcionamento no conjunto de dados fornecido.

2.2 <Tarefa 2> Implementação do método de mínimos quadrados num modelo de classificação

De forma semelhante a primeira parte vamos calcular os coeficientes no treinamento e após o cálculo aplicaremos na fórmula de h(x) e faremos a classificação baseada no valor de h(x).

O código implementa um sistema para treinar coeficientes usando o método dos mínimos quadrados. Primeiramente, a função Treinar_Coeficientes_cancer_data_otimizado ajusta o modelo linear aos dados de entrada usando a eliminação gaussiana para encontrar os coeficientes que minimizam o erro quadrático.

Depois a função Fazer_Previsoes_cancer_data utiliza os coeficientes treinados para fazer previsões. Multiplica as características dos dados de entrada pelos coeficientes para obter um vetor de previsões, que são transformadas em rótulos de classe usando uma regra de decisão simples. h(x) > 0 implica classe 1 e h(x) <0 implica classe -1.

Por fim para avaliar o desempenho do modelo, a função Calcular_Medidas_Desempenho utiliza as previsões geradas e os rótulos reais. Ela constrói uma matriz de confusão que conta os verdadeiros positivos, verdadeiros negativos, falsos positivos e falsos negativos. A partir dessa matriz, são calculadas medidas de desempenho como acurácia, precisão, recall, probabilidade de falso alarme e probabilidade de falsa omissão. Essas medidas fornecem uma visão abrangente de quão bem o modelo está performando na tarefa de classificação.

Colocaremos em funcionamento no conjuntos de dados de câncer:

```
--> cancer_data_test = csvRead('cancer_test_2024.csv', ';');

--> cancer_data_train = csvRead('cancer_train_2024.csv', ';');

--> coeficientes = Treinar_Coeficientes_cancer_data(cancer_data_train);

--> Vetor_previsoes = Fazer_Previsoes_cancer_data(coeficientes, cancer_data_train);

--> [confusion_matrix, acuracia, precisao, recall, prob_falso_alarme, prob_falsa_omissao] = Calcular_Medidas_Desempe
 confusion_matrix  =

   152.    19.
   8.      101.
 acuracia  =

   0.9035714
 precisao  =

   0.9266055
 recall  =

   0.8416667
 prob_falso_alarme  =

   0.05
 prob_falsa_omissao  =

   0.1583333
```

Figura 3: Previsões de classificações obtidas através dos algoritmos (conjunto treino)

```
--> [confusion_matrix, acuracia, precisao, recall, prob_falso_alarme, prob_falsa_omissao] = Calcular_Medidas_Desempe
confusion_matrix  =

  116.   55.
  78.    31.
acuracia  =

  0.525
precisao  =

  0.2844037
recall  =

  0.3604651
prob_falso_alarme  =

  0.4020619
prob_falsa_omissao  =

  0.6395349
```

Figura 4: Previsões de classificações obtidas através dos algoritmos (conjunto teste)

Com base nas métricas obtidas nos conjuntos de treinamento e teste, podemos observar que o modelo está se comportando de maneira diferente em cada um deles. No conjunto de treinamento, o modelo apresenta uma acurácia alta (90.36% que é a porcentagem de acertos então já conclui uma parte desta atividade), o que indica sua eficácia geral na classificação dos dados utilizados para treiná-lo. A precisão também é alta (92.66%), significando que a maioria das previsões positivas feitas pelo modelo está correta. O recall (84.17%) é relativamente bom, mostrando que o modelo consegue identificar uma proporção significativa dos casos positivos verdadeiros.

No entanto, no conjunto de teste, observamos uma grande queda em todas as métricas. A acurácia (52.5%) indica que o modelo não está performando tão bem quanto no treinamento, e a precisão (28.44%) mostra que a maioria das previsões positivas no conjunto de teste é, na verdade, incorreta. O recall (36.05%) revela que o modelo está capturando apenas uma fração dos casos positivos verdadeiros. Além disso, a probabilidade de falso alarme (40.21%) e a probabilidade de falsa omissão (63.95%) destacam que o modelo está cometendo muitos erros de classificação, tanto falsos positivos quanto falsos negativos.

Essas diferenças entre os conjuntos de treinamento e teste sugerem que o modelo pode estar sobreajustado aos dados de treinamento, o que compromete sua capacidade de generalização para novos dados.

2.3 <Tarefa 3> Melhorando a generalização do algorítmo

Os resultados para o conjunto de testes não foram bons. Para melhorar a generalização do modelo, o código foi então modificado para considerar apenas os coeficientes mais significativos durante o treinamento. Primeiramente, na função Treinar_Coeficientes_cancer_data_otimizado, os coeficientes são inicialmente calculados usando todos os atributos disponíveis. Em seguida, são selecionados os num_de_indices coeficientes com os maiores valores absolutos, que indicam maior impacto na predição. Esses índices são armazenados em maiores_indices.

Após a seleção dos índices mais importantes, a matriz de características Novo_A é reconstruída apenas com as colunas correspondentes aos índices selecionados. Essa abordagem reduz o número de características utilizadas para o cálculo dos coeficientes finais, focando apenas nas mais relevantes para a predição.

Na função Fazer_Previsoes_cancer_data, as previsões são feitas usando apenas as colunas de dados especificadas pelos maiores_indices. Isso implica que o modelo final e as previsões são baseados apenas nas características que foram identificadas como mais relevantes durante o treinamento.

Essas mudanças irão melhorar a capacidade do modelo de generalizar para novos dados, evitando possíveis problemas de sobreajuste aos dados de treinamento. Ao focar nos coeficientes mais importantes, o modelo terá um desempenho melhor em novos conjuntos de dados.

Testaremos novamente com o conjunto de treino e teste:

```
--> [coeficientes_otim, maiores_indices] = Treinar_Coeficientes_cancer_data_otimizado(cancer_data_train, 5);

--> Vetor_previsoes_otim = Fazer_Previsoes_cancer_data_otimizado(coeficientes_otim, cancer_data_train, maiores_indices);

--> [confusion_matrix, acuracia, precisao, recall, prob_falso_alarme, prob_falsa_omissao] = Calcular_Medidas_Desempenho(Vetor_pre
confusion_matrix  =

  152.   26.
   8.    94.
acuracia  =

  0.8785714
precisao  =

  0.9215686
recall  =

  0.7833333
prob_falso_alarme  =

  0.05
prob_falsa_omissao  =

  0.2166667
```

Figura 5: Previsões de classificações otimizada obtidas através dos algoritmos (conjunto treino)

```
--> Vetor_previsoes_otim = Fazer_Previsoes_cancer_data_otimizado(coeficientes_otim, cancer_data_test, maiores_indices);

--> [confusion_matrix, acuracia, precisao, recall, prob_falso_alarme, prob_falsa_omissao] = Calcular_Medidas_Desempenho(Vetor_pr
confusion_matrix  =

  183.   13.
   11.   73.
acuracia  =

  0.9142857
precisao  =

  0.8690476
recall  =

  0.8488372
prob_falso_alarme  =

  0.0567010
prob_falsa_omissao  =

  0.1511628
```

Figura 6: Previsões de classificações otimizada obtidas através dos algoritmos (conjunto teste)

Conseguimos uma ótima melhoria priorizando os features mais relevantes!