Fundação Getúlio Vargas
Escola de Matemática Aplicada
Álgebra Linear Numérica

Método dos Mínimos Quadrados

Aluno:
- Jeann da Rocha Silva
Professor:
- Antonio Carlos Saraiva Branco

May
2024

1) (Texto do livro Cálculo - Volume 2 – James Stewart) Em 1928, Charles Cobb e Paul Douglas publicaram um estudo no qual modelaram o crescimento da economia norte-americana durante o período de $1899 - 1922$. Eles consideraram uma visão simplificada da economia em que a saída da produção é determinada pela quantidade de trabalho envolvido e pela quantidade de capital investido. Apesar de existirem muitos outros fatores afetando o desempenho da economia, o modelo mostrou-se bastante preciso. A função utilizada para modelar a produção era da forma

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

onde P é a produção total (valor monetário dos bens produzidos no ano); L é a quantidade de trabalho (número total de pessoas-hora trabalhadas no ano); e K é a quantidade de capital investido (valor monetário das máquinas, equipamentos e prédios); b e α são parâmetros (constantes) a serem determinados. Cobb e Douglas usaram os dados da tabela a seguir e o Método dos Mínimos Quadrados para obter os valores de b e de α.

Ano	P	L	K
1899	100	100	100
1900	101	105	107
1901	112	110	114
1902	122	117	122
1903	124	122	131
1904	122	121	138
1905	143	125	149
1906	152	134	163
1907	151	140	176
1908	126	123	185
1909	155	143	198
1910	159	147	208
1911	153	148	216
1912	177	155	226
1913	184	156	236
1914	169	152	244
1915	189	156	266
1916	225	183	298
1917	227	198	335
1918	223	201	366
1919	218	196	387
1920	231	194	407
1921	179	146	417
1922	240	161	431

a) **Faça como Cobb e Douglas: use o Método dos Mínimos Quadrados para estimar os valores dos parâmetros b e α. Mostre a sua modelagem para o problema ser resolvido pelo Método dos Mínimos Quadrados. Use o Scilab para fazer os cálculos e mostre os prints das telas do Scilab.**

b) **Agora, use a função de Cobb-Douglas encontrada no item a) e teste a sua adequação calculando os valores da produção nos anos de 1910 e 1920. Comente!**

---

a) Primeiro precisamos transformar o modelo de Produção de Cobb-Douglas para um no formato linear. Isto é possível tomando-se o logaritmo de ambos os lados do modelo original, como segue abaixo.

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

Denotando $\log(b)$ por a_0 e α por α_1, queremos encontrar a função linear $h(x) = \alpha_0 + \alpha_1 x$ que melhor minimiza a soma dos erros de distância dos pontos $(\log(L) - \log(K), \log(P) - \log(K))$ para os pontos $(\log(L) - \log(K), h(\log(L) - \log(K)))$. Com isso, implementamos a função *Minimos_Quadrados_Cobb_Douglas* que dada a matriz A contendo os dados P, L e K, retorna o vetor $x = (a_0, a_1)$ que define h segundo o argumento dado acima. Lembrando que $\alpha_0 = \log(b) \Leftrightarrow b = e^{\alpha_0}$ e $\alpha_1 = \alpha$. Como é necessário resolver um sistema linear para o método dos mínimos quadrados, consta no mesmo arquivo (*Cobb_Douglas.sci*) a função *Gaussian_Elimination_4* de eliminação gaussiana. Testando a implementação, obtemos

```
--> A=csvRead("tabela_CobbDouglas.csv");

--> [X,Y,x]=Minimos_Quadrados_Cobb_Douglas(A);

--> x
 x  =

   0.0070440
   0.7446062
```

Como queremos α e b, fazemos $\alpha = a_1$ e $b = e^{\alpha_0}$, como segue abaixo:

```
--> alfa = x(2); b = exp(x(1));

--> [alfa; b]
 ans  =

   0.7446062
   1.0070689
```

b) Uma vez obtidos α e b, aplicamos os dados referentes a L e K no modelo de Cobb Douglas ($P = bL^{\alpha}K^{1-\alpha}$) para obter P e calculamos, em seguida, a raiz quadrada da soma dos erros das coordenadas previstas de P pelas coordenadas reais para contabilizar o erro de previsão. Isto é feito na função *Cobb_Douglas*. O resultado entre os anos de 1910 e 1920 (linhas 12 até 22 da tabela) é

```
--> [P_prev, erro]=Cobb_Douglas(alfa, b, A(12:22,:));

--> erro
 erro  =

   9.9176786
```

Visualmente, os dados reais e os dados obtidos lado a lado foram

```
--> [A(12:22, 2) P_prev]
 ans  =

   159.   161.76185
   153.   164.15515
   177.   171.87727
   184.   174.62256
   169.   172.74202
   189.   180.04169
   225.   208.73428
   227.   228.06104
   223.   235.90134
   218.   234.84028
   231.   236.07215
```

O dados reais são os da esquerda e os obtidos são os da direita. É fácil observar que eles são próximos, diferindo por algumas unidades. Em virtude disso, tomamos a raiz quadrada dos erros quadrados médios para calcular o erro da previsão. Caso contrário, o cálculo podia gerar um erro enorme devido aos quadrados que são somados para contabilizar o erro. Como a tabela apresenta poucos dados, é natural que a previsão tenha erros de algumas unidades nas previsões. Para tabelas maiores, esse detalhe seria corrigido. Em resumo, o modelo aparenta ser eficiente.

2) Agora vamos usar o método dos mínimos quadrados para implementar um método rudimentar de "machine learning" para diagnosticar câncer de mama a partir de um conjunto de características fornecidas para cada paciente. São dados dois arquivos: um arquivo para "treinamento" (cancer_train_2024.csv) do modelo e um arquivo para "teste" (cancer_test_2024.csv). Os dois arquivos contém 280 registros cada um, partes do "Wiscosin Diagnostic Breast Cancer dataset". Cada registro de cada arquivo contém 11 valores: os 10 primeiros correspondem a valores reais de 10 características dos núcleos celulares observados em imagens digitalizadas de uma fina camada de massa mamária coletada de cada paciente. O décimo primeiro valor é $+1$ se a paciente tem câncer de mama e -1, caso contrário.

Sendo x o vetor das 10 características de cada paciente (variáveis independentes) e y o valor ($+1$ ou -1) que indica o diagnóstico (variável dependente), a ideia é, usando o arquivo de treinamento, obter o hiperplano $y = h(x)(y = \alpha_0 + \sum_{i=1}^{10} \alpha_i x_i)$ que "melhor se ajuste aos dados fornecidos" usando o método dos mínimos quadrados.

Uma vez obtido o hiperplano, o mesmo será usado para classificar cada paciente da seguinte forma: se $h(x) \geq 0$, então o diagnóstico é $+1$ (tem câncer), caso contrário, o diagnóstico é -1 (não tem câncer).

Use o seu classificador (hiperplano) e calcule a porcentagem de acertos sobre o arquivo de treinamento (de certa forma é uma medida do ajuste do seu modelo aos dados de treinamento) e sobre o arquivo de teste (de certa forma é uma medida da capacidade de generalização do seu modelo).

Construa uma Matriz de Confusão (Confusion Matrix) (pesquise a respeito) com o conjunto de teste e calcule as diversas medidas daí decorrentes, tais como: acurácia, precisão, recall, probabilidade de falso alarme, probabilidade de falsa omissão de alarme. Interprete essas medidas e comente os resultados obtidos.

O código para este problema de Machine Learning consta no arquivo *Manchine_Learning.sci*, que contém a função *Minimos_Quadrados_Machine_Learning* que retorna os coeficientes da função $h(x) = a_0 + a_1x + ... + a_{10}x_{10}$ pelo método dos mínimos quadrados. Contém também a função *Classificador* que retorna os dados esperados (± 1) e a porcentagem de acertos do modelo utilizado (seja ele de treino ou de teste). Por fim, há também a função *Gaussian_Elimination_4*, necessária para o funcionamento do método. Aplicando a base de dados de treino, obtemos o vetor dos coeficientes abaixo:

```
--> A = csvRead("cancer_train_2024.csv", ";");

--> [alfa, X]=Minimos_Quadrados_Manchine_Learning(A);

--> alfa
 alfa  =

  -6.2101493
   15.902409
   1.5568757
  -5.0718598
  -7.1846562
   1.2702227
  -0.9298812
   0.5285964
   1.9535131
  -0.0470564
   0.7701829
```

Comparando com os dados de treino, obtemos

```
--> [diag_Prev, percent_acerto]=Classificador(alfa, X, A(:,11));

--> percent_acerto
 percent_acerto  =

   0.9214286
```

Agora, com os dados de teste

```
--> B=csvRead("cancer_test_2024.csv", ";");

--> X=[ones(280, 1) B(:,1:10)];

--> [diag_Prev, percent_acerto]=Classificador(alfa, X, B(:,11));

--> percent_acerto
 percent_acerto  =

   0.8892857
```

Logo, observa-se que o modelo aparenta ser eficiente, tendo 92% de acertos no conjunto de treino e 89% no conjunto de teste. Plotando a matriz de confusão e as respectivas métricas geradas a partir das funções *Confusion_Matrix* e *Metrics* no arquivo *Metrics_and_Confusion.sci*, obtemos

```
--> Confusin_Matrix(B(:,11), diag_Prev)
 ans  =

   80.   25.
   6.    169.

--> M=Confusin_Matrix(B(:,11), diag_Prev)
 M  =

   80.   25.
   6.    169.

--> Metrics(M)

  "Accuracy: 0.8892857"

  "Precision: 0.7619048"

  "Recall: 0.9302326"

  "F1 Score: 0.8376963"

  "Probability False Alarm: 0.2380952"

  "Probability Alarm False Omiss: 0.0342857"
```

Pelos valores observados acima, observa-se que o método é eficaz, mas podemos perder algumas informações de modo a não mudar muito os resultados obtidos. Isto é importante, principalmente em questão de eficiência, pois estaremos lidando com uma quantidade menor de variáveis. Faremos isso, justificando devidamente, no próximo item.

3) Com relação ao item anterior e ao hiperplano encontrado, todas as variáveis de entrada são igualmente relevantes para o diagnóstico? O que você acha? Se você achar que alguma(s) variável(is) não tem muita influência na classificação, retire essa(s) variável(is) e teste a sua hipótese.

Observa-se que os dados do arquivo de treino, bem como os do arquivo de teste são valores entre 0 e 1. Além disso, alguns dos coeficientes encontrados pelo Método dos Mínimos Quadrados são valores entre -1 e 1. Como o produto por valores de módulo menor que 1 tende a diminuir, ainda mais se estamos efetuando o produto entre valores entre 0 e 1 e valores entre -1 e 1, podemos inferir que os coeficientes entre 0 e 1 encontrados não agregam informação, visto que eles fazem o valor dos dados informados tender a 0 e, portanto, não trazem distinção entre o lado positivo e negativo (embora, convencionamos que 0 é entendido como o caso $+1$). Com base nisso, vamos desconsiderar estas variáveis e efetuar o método novamente para encontrar novos coeficientes e, assim, determinar a nova matriz de confusão e métricas que indiquem se nossas suspeitas estão corretas. Observando os coeficientes obtidos, vemos que se tratam de

$$\alpha_7 \approx -0.93, \alpha_8 \approx -0.53, \alpha_{10} \approx -0.05 \text{ e } \alpha_{11} \approx 0.77$$

isto é, das colunas $6, 7, 9$ e 10 da tabela. Para efeitos práticos, vamos implementar este caso no mesmo arquivo *Manchine_Learning.sci* na função *Minimos_Quadrados_Manchine_Learning_2*, que é similar ao já feito, apenas com o detalhe da remoção destes coeficientes (colunas da tabela). Passemos a verificar os dados obtidos.

```
--> [alfa, X]=Minimos_Quadrados_Manchine_Learning_2(A);

--> alfa
 alfa  =

  -5.3481949
   17.617910
   1.4800467
  -8.4905884
  -6.0154589
   1.1269100
   2.2239952
```

Vemos que todos os coeficientes deram maior que 1 em módulo, o que já indica uma leve garantia do que estava-se considerando. Vamos analisar a porcentagem de acerto para os dados de treino

```
[diag_Prev, percent_acerto]=Classificador(alfa, X, A(:,11));

--> percent_acerto
 percent_acerto  =

   0.9178571
```

Muito próxima do caso anterior, diferindo por cerca de 1%, o que não qualifica uma mudança severa nos resultados. Agora, passemos aos dados de teste.

```
--> B=csvRead("cancer_test_2024.csv", ";");

--> X=[ones(280, 1) B(:, 1:5) B(:, 8)];

--> [diag_Prev, percent_acerto]=Classificador(alfa, X, B(:,11));

--> percent_acerto
 percent_acerto  =

   0.8535714
```

Novamente, vemos um resultado similar, porém um pouco inferior ao obtido para o caso de todos os coeficientes (cerca de 3% menos). De novo, não há muito diferença nos resultados. Por fim, analisemos as métricas

```
--> M=Confusin_Matrix(B(:,11), diag_Prev)
 M  =

   81.   36.
   5.    158.

--> Metrics(M)

  "Accuracy: 0.8535714"

  "Precision: 0.6923077"

  "Recall: 0.9418605"

  "F1 Score: 0.7980296"

  "Probability False Alarm: 0.3076923"

  "Probability Alarm False Omiss: 0.0306748"
```

Como se vê os resultados são similares e, claramente, o tempo de execução é bem menor por se tratar de menos variáveis. Isto mostra que variáveis pequenas sob o efeito de produto por constantes pequenas não configuram muita informação para os dados. Logo, a perda de informação ao retirar tais constantes é razoável perante o tempo de execução (isto, obviamente, para contextos mais amplos, com bases de dados bem maiores).

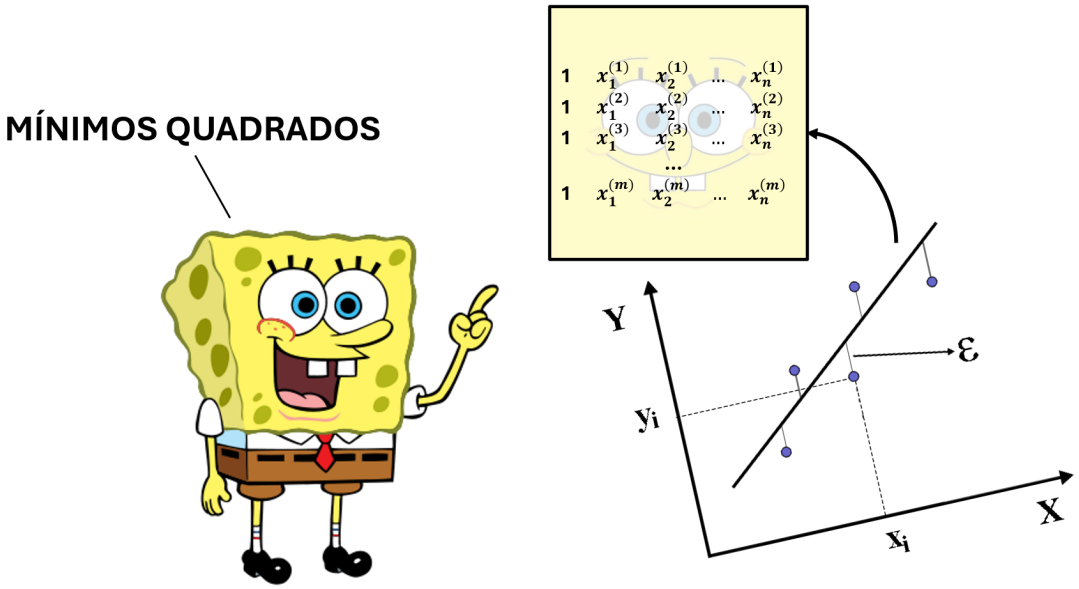

Figura 1: Bob Esponja Calça Mínima Quadrada