MÉTODO DOS MÍNIMOS QUADRADOS

Relatório da Aula Prática 4

Nome: Felipe Marques Esteves Lamarca
Matrícula: 211708306

Sumário

1 Introdução

Este relatório utiliza o Scilab para implementar o Método dos Mínimos Quadrados em duas situações distintas. Na primeira, utiliza-se a função de Charles Cobb e Paul Douglas, que modela o crescimento da economia norte-americana entre 1899 e 1922. Nesse caso, modelamos o problema para que possamos estimar os parâmetros α e b a partir do Método dos Mínimos Quadrados para, então, tentar prever os valores da produção em determinados anos desse período.

Na segunda parte do relatório, implementa-se um método simplificado de diagnóstico de câncer de mama a partir de dados previamente fornecidos. Primeiro treinamos o modelo a partir de um arquivo de treinamento e, depois, fazemos o teste com um arquivo de teste. Por fim, calculamos algumas informações adicionais a partir da criação de uma Matriz de Confusão.

2 Parte 1: A Função de Cobb-Douglas

Leve em consideração que a função utilizada pelos estudiosos para modelar o crescimento da economia dos Estados Unidos no início do século XX respeitava a seguinte forma:

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

Nessa função, P diz respeito à produção total, L representa a quantidade de trabalho e K é a quantidade de capital investido. Os parâmetros α e b são constantes a serem determinadas pelo Método dos Mínimos Quadrados, que implementaremos a seguir. Antes, no entanto, precisaremos trabalhar um pouco a função acima — afinal, possuímos os dados de P, L e K, e desejamos estimar α e b. Segue:

$$P = bL^{\alpha}K^{1-\alpha}$$
$$\ln(P) = \ln(bL^{\alpha}K^{1-\alpha})$$
$$\ln(P) = \ln(b) + \ln(L^{\alpha}) + \ln(K^{1-\alpha})$$
$$\ln(P) = \ln(b) + \alpha\ln(L) + \ln(K) - \alpha\ln(K)$$
$$\ln(P) - \ln(K) = \ln(b) + \alpha(\ln L - \ln K)$$
$$\ln(\frac{P}{K}) = \ln(b) + \alpha\ln(\frac{L}{K})$$

Ora, agora estamos lidando com um sistema razoavelmente simples. Podemos implementá-lo no Scilab e estimar α e b a partir do Método dos Mínimos Quadrados. É verdade que estamos aplicando a função `ln()` ao parâmetro b, mas bastará aplicar a função `exp()` ao final para obter seu valor. Veja a função abaixo:

```
function[alpha, b] = CobbDouglas(infos)
    bsys = log(infos(:, 1)) - log(infos(:, 3));
    A = [ones(bsys), log(infos(:, 2)) - log(infos(:, 3))];
    x = zeros(2);

```

```
    // aplicamos o Metodo dos Minimos Quadrados
    // A_transpose * A * x = A_transpose * bsys
    x = Gaussian_Elimination_4(A'*A, A'*bsys);
    b = exp(x(1));
    alpha = x(2);
endfunction
```

Verifique que utilizamos uma "velha conhecida", a função `Gaussian_Elimination_4()`, criada na primeira Aula Prática e utilizada com alguma recorrência em outros relatórios desta disciplina. Trata-se de uma forma prática de encontrar o vetor `x` que resolve o sistema.

Agora que a função está implementada, vamos importar o arquivo `.csv` com os dados de P, L e K para cada ano entre 1899 e 1922. Utilizaremos essas informações da tabela para determinar os valores de α e b e então verificar se as predições do valor de P para 1910 e 1920 são razoáveis.

Vamos ler o `.csv` e analisá-lo por um instante:

```
-->infos = csvRead('data.csv')

 infos  =
   100.   100.   100.
   101.   105.   107.
   112.   110.   114.
   122.   117.   122.
   124.   122.   131.
   122.   121.   138.
   143.   125.   149.
   152.   134.   163.
   151.   140.   176.
   126.   123.   185.
   155.   143.   198.
   159.   147.   208.
   153.   148.   216.
   177.   155.   226.
   184.   156.   236.
   169.   152.   244.
   189.   156.   266.
   225.   183.   298.
   227.   198.   335.
   223.   201.   366.
   218.   196.   387.
   231.   194.   407.
   179.   146.   417.
   240.   161.   431.
```

Cada coluna corresponde a dados de P, L e K, respectivamente. A primeira linha faz referência ao ano de 1899 e seguimos a sequência até a última linha, referente a 1922. Na prática, ao aplicarmos a função `CobbDouglas()` criada, receberemos os melhores valores para os parâmetros α e b. Assim, poderemos finalmente utilizar os valores de L e K dos anos desejados (no caso, 1910 e 1920) para tentar predizer o valor de P em cada um deles. Veja:

```
-->[alpha, b] = CobbDouglas(infos)

 alpha  =
```

```
   0.7446062

 b  =
   1.0070689
```

Encontrados os valores de α e b, vamos testá-los para tentar predizer o valor de P em 1910 (note que este ano corresponde à linha 12 da tabela). Diremos que a predição foi boa se recebermos um valor razoavelmente próximo de 159:

```
-->b * infos(:, 2)(12)^alpha * infos(:, 3)(12)^(1-alpha)

 ans  =
   161.76185
```

O que o código acima faz é justamente aplicar a função $P = bL^{\alpha}K^{1-\alpha}$ com os valores de b e α encontrados, além dos valores de K e L da linha 12 da tabela, referentes ao ano de 1910. E, de fato, recebemos um número próximo de 159, o que significa que a predição parece ter funcionado para o ano em questão. Vejamos como a função se comporta para 1920, caso no qual deveríamos receber um valor próximo de 231:

```
-->b * infos(:, 2)(22)^alpha * infos(:, 3)(22)^(1-alpha)

 ans  =
   236.07215
```

Dessa vez recebemos um número um pouco mais distante, mas ainda assim razoavelmente próximo do valor esperado. A predição mais uma vez funcionou bem.

3 Parte 2: Diagnóstico de Câncer de Mama

3.1 Implementação

Nesse caso, temos duas fontes de dados: a primeira, `cancer_train.csv` é o arquivo que utilizaremos para treinar nosso modelo e encontrar um vetor que resolva o sistema $A^T A\overline{\alpha} = Ab$; a segunda é o arquivo `cancer_test.csv`, cujos dados serão utilizados para verificar se o vetor encontrado funciona razoavelmente bem no processo de predição. Implementamos, portanto, duas funções distintas: uma para o treinamento e outra para a testagem. Veja a seguir:

```
function[acertos_training_pct, alphabar] =
    MinimosQuadrados_Training(treinamento)

    ytraining = treinamento(:, 11);
    Xtraining = [ones(ytraining), treinamento(:, 1:10)];
    alphabar = Gaussian_Elimination_4(Xtraining' * Xtraining,
    Xtraining' * ytraining);
    prevision = Xtraining * alphabar;

    conf_prevision = prevision .* ytraining;
    acertos_training_pct = (sum(conf_prevision >= 0) / size(
    Xtraining, "r"));

endfunction
```

Essa função cria um vetor y de treinamento, que é justamente a coluna corresponde à classificação das pacientes entre aquelas com câncer de mama e aquelas sem a doença. A matriz X recebe uma coluna composta por 1's e todas as outras 10 colunas são as outras informações da tabela e que serão utilizadas para prever se a paciente está ou não com câncer. Utilizando a função `Gaussian_Elimination_4()`, encontramos o vetor $\overline{\alpha}$, que será utilizado para a previsão logo em seguida (tanto no caso dos dados de treinamento, quanto no caso dos dados de teste).

O restante do código faz a conferência da previsão, multiplicando o vetor `prevision` pelo vetor de teste e calculando o percentual de acertos. Dessa forma, podemos verificar se o algoritmo implementado está fazendo um bom trabalho.

A função para a testagem é diferente em alguns poucos pontos:

```
function[acertos_testing_pct] = MinimosQuadrados_Testing(testagem,
    alphabar)

    ytesting = testagem(:, 11);
    Xtesting = [ones(ytesting), testagem(:, 1:10)];
    prevision = Xtesting * alphabar;

    conf_prevision = prevision .* ytesting;
    acertos_testing_pct = (sum(conf_prevision >= 0) / size(Xtesting
    , "r"));

endfunction
```

Nesse caso, não desejamos mais calcular um $\overline{\alpha}$ — vamos utilizar o vetor calculado no treinamento e, por isso, a função recebe esse vetor como uma argumento. No mais, as variáveis são criadas da mesma forma, mas dessa vez com base no arquivo e teste.

Vamos, agora, calcular o $\overline{\alpha}$ com a função de treinamento. Se o algoritmo estiver funcionando corretamente, devemos observar também um percentual elevado de acertos na predição:

```
-->[acertos_training, alphabar] = MinimosQuadrados_Training(
    treinamento)

 acertos_training  =
   0.93

 alphabar  =
  -6.7579731
   29.311052
   2.0765803
  -18.730222
  -7.3665161
   1.2222756
   0.2283419
   0.0503253
   2.2385058
   0.0249405
   0.7704282
```

O $\overline{\alpha}$ calculado não nos diz muito sobre o funcionamento do algoritmo. Mais importante que ele é verificarmos que a implementação acertou 93% das classificações das pacientes entre as duas possibilidades: ter a doença ou não ter a doença. Mas note:

esse percentual elevado nos sugere que o processo de previsão está implementado corretamente, mas não garante que ele será eficaz em fazer previsões a partir de dados desconhecidos (de teste). Para isso, vamos utilizar o $\overline{\alpha}$ nos dados do arquivo de teste e verificar o percentual de acertos:

```
-->[acertos_testing] = MinimosQuadrados_Testing(testagem, alphabar)

 acertos_testing  =
   0.7115385
```

Nesse caso, o percentual é notavelmente menor. Nosso algoritmo, para os dados que recebeu no teste, foi capaz de prever 71% dos resultados corretamente. Na prática, a capacidade de generalização do modelo não é muito grande e poderia ser melhorada caso tivéssemos acesso a mais dados a respeito dos pacientes.

3.2 Matriz de Confusão

Uma Matriz de Confusão mapeia, na forma matricial, o número de True Positives (TP) e False Positives (FP) na primeira linha e o número de True Negatives (TN) e False Negatives (FN) na segunda linha. Vamos, agora, criar uma Matriz de Confusão a partir do conjunto de dados referente ao teste a partir do código abaixo:

```
ytesting = testagem(:, 11);
Xtesting = [ones(ytesting), testagem(:, 1:10)];

M = [Xtesting*alphabar ytesting]

TP = sum((M(:,1)) >= 0 & M(:,2) >= 0);
FP = sum((M(:,1)) >= 0 & M(:,2) <= 0);
TN = sum((M(:,1)) <= 0 & M(:,2) <= 0);
FN = sum((M(:,1)) <= 0 & M(:,2) >= 0);

MatrizConfusao = [TP FP; FN TN]
```

Primeiro definimos nossos X e y de teste. Criamos uma matriz M que servirá para comparação entre os dados que obtivemos com a predição e o valor ”verdadeiro”contido no vetor y — aquele que aponta se a paciente tem ou não câncer de mama. Fazemos, em seguida, uma comparação entre os valores do vetor resultante da predição e o vetor y:

TRUE POSITIVE se o valor encontrado e o valor do vetor y forem positivos. Nesse caso o paciente tem a doença e previmos que ele de fato possuía o câncer;

FALSE POSITIVE se o valor encontrado for positivo e o valor do vetor y for negativo. Nesse caso o paciente não tem a doença, mas o algoritmo identificou incorretamente que ele a possui;

TRUE NEGATIVE se o valor encontrado e o valor do vetor y forem negativos. Nesse caso o paciente não tem a doença e previmos que ele de fato não a possuía;

FALSE NEGATIVE se o valor encontrado for negativo e o valor do vetor y for positivo. Nesse caso o paciente tem a doença, mas o algoritmo identificou incorretamente que ele não a possuía.

Temos a seguinte matriz como resultado:

```
MatrizConfusao

 MatrizConfusao  =
   60.    75.
   0.     125.
```

3.3 Indicadores

Com a Matriz de Confusão criada, podemos agora calcular alguns indicadores a partir dessas informações. Focaremos em 5: a acurácia, a precisão, *recall*, a probabilidade de falso alarme e a probabilidade de falsa omissão de alarme. Vamos definir cada um deles e analisar nossos resultados:

Acurácia é a medida do total de acertos entre todas as previsões realizadas. Note que esse valor já foi calculado anteriormente, quando apontamos o percentual de acertos da previsão;

```
acuracia = (TP+TN)/(TP+TN+FP+FN)

 acuracia  =
   0.7115385
```

Não é um valor muito alto, o que compromete em alguma medida a generalização do modelo. Mesmo assim, o resultado está em linha com o valor que vimos anteriormente.

Precisão é a medida da quantidade de casos realmente positivos dentre todos os casos indicados como positivos (afinal, como vimos, há casos em que o programa classifica incorretamente);

```
precisao = (TP)/(TP+FP)

 precisao  =
   0.4444444
```

Note que a precisão do modelo é baixa, menor que 50%. Veja que o modelo verificou um número alto de False Positives, o que certamente é um problema de classificação que compromete a generalização. Esse é um problema bastante grave — afinal, uma notícia de testagem positiva para câncer pode ser recebida de forma muito negativa por um paciente. E, se ele não tem de fato a doença, será exposto a uma situação desnecessária e pode ter sua integridade comprometida.

Recall é a medida da quantidade de casos positivos entre todos os casos que eram de fato positivos — isto é, considerando também os False Negatives;

```
recall = (TP)/(TP+FN)

 recall  =
   1.
```

De fato, é um resultado bastante positivo e de alguma forma compensa o problema da precisão. Pela Matriz de Confusão vimos que não há False Negatives, um resultado importante na medida em que um paciente não deixará de ser notificado caso teste positivo para a doença.

Probabilidade de falso alarme é a medida que indica o percentual de False Positives dentre todas as previsões realizadas;

```
probFalsoAlarme = (FP)/(TP+TN+FP+FN)

 probFalsoAlarme  =
   0.2884615
```

É uma probabilidade alta e, como vimos, trata-se de um problema que compromete a generalização do modelo. Além disso, apontar para um determinado paciente que ele tem câncer sem que ele de fato tenha a doença pode gerar uma série de situações desagradáveis.

Probabilidade de falsa omissão de alarme é a medida que indica o percentual de False Negatives entre todos os casos classificados como negativos.

```
probFalsaOmissaoAlarme = (FN)/(TN+FN)

 probFalsaOmissaoAlarme  =
   0.
```

Como vimos na Matriz de Confusão, não há casos de False Negative e, logicamente, a probabilidade de omissão de falso alarme é nula.