Atividade 1 **Yan A. S. Duarte**
PEL 208 Tópicos Especiais em Aprendizagem
Prof. Reinaldo A. C. Bianchi Entrega: 11/10/2017

Introdução

Esse relatório tem como objetivo detalhar a teoria, a implementação, os resultados e a conclusão da primeira atividade do curso. A propósta do exercício é implementar o método dos mínimos quadrados, tanto o básico (linear), quanto o quadrático.

Teoria

Matriz Inversa

Para a execução do método dos mínimos quadrados é necessário obter a matriz inversa. Uma matriz é inversa de outra quando o produto de ambas resulta em uma matriz identidade, ou seja:

$$A.A^{-1} = A^{-1}.A = I$$

Para que isso aconteça, a matriz A deve ser uma matriz quadrada (mesmo número de linhas e colunas).

Podemos encontrar a matriz inversa A^{-1} com a seguinte fórmula:

$$A^{-1} = \frac{1}{det(A)} adj(A)$$

onde:

- $det(A)$: é a determinante da matriz A que representa o valor numérico da matriz;
- $adj(A)$: é a matriz adjunta da matriz A que representa a transposta de sua matriz dos cofatores.

Para definirmos o que é um cofator, é necessário, a princípio, definirmos o que é o menor principal ou menor complementar associado a um elemento qualquer de uma matriz quadrada. Essa definição se dá pela determinante de uma matriz D obtida a partir da eliminação de uma linha i e uma coluna j da matriz A.

Sendo assim, podemos assumir que um cofator $\tilde{a}_{ij}$ associado a um elemento a_{ij} é definido por:

$$\tilde{a}_{ij} = (-1)^{i+j} D_{ij}$$

Logo, a matriz de cofatores C é formada por todos os cofatores da matriz original A.

Mínimos Quadrados

O método dos mínimos quadrado é uma técninca de otimização matemática que possui como objetivo encontrar o melhor ajuste para um conjunto de dados. Seu funcionamento se dá através da minimização da somatória da diferença entre o valor estimado e do dado observado elevado ao quadrado:

$$\min \sum (y_i - \hat{y}_i)^2$$

Podemos expressar sua notação matricial da seguinte forma:

$$\beta = (X^T X)^{-1} X^T y$$

Implementação

Para a elaboração do exercício foi utilizada a linguagem de programação Python. Foram criadas funções separadas para calcular cada etapa da fórmula dos mínimos quadrados que serão descritas a seguir.

check_tamanho_matriz

Função para exibir retorno de erro caso a matriz de entrada tenha dimensão diferente de 2x2 ou 3x3.

```
def check_tamanho_matriz(matriz):
    lin, col = matriz.shape

    if (lin!=col):
        return {
            'success': False,
            'message': 'Matriz deve ser quadrada'
        }

    if (lin < 2 or lin > 3):
        return {
            'success': False,
            'message': 'Apenas matrizes de tamanho 2x2 '+
                       'ou 3x3 sao aceitas'
        }

    return{
        'success': True,
        'message': 'Tamanho da matriz: {}x{}'.format(lin, col)
    }
```

calc_determinante

Função que calcula o determinante de uma matriz de dimensão 2x2 ou 3x3.

```
def calc_determinante(matriz):
    lin, col = matriz.shape
    tamanho = check_tamanho_matriz(matriz)

    if not tamanho.get('success'):
        return {
            'success': False
            , 'message': tamanho.get('message')
            , 'result': None
        }

    if (lin == 2):
        return {
            'success': True
            , 'message': 'Determinante obtida com sucesso.'
            , 'result': ( matriz[0][0] * matriz[1][1] ) -
                        ( matriz[0][1] * matriz[1][0] )
        }

    if (lin == 3):
        det = 0
        for i in range(lin):
            det = det + (
                matriz[0][i] *
                (matriz[1][(i+1)%3] * matriz[2][(i+2)%3] -
                 matriz[1][(i+2)%3] * matriz[2][(i+1)%3])
            );
        return {
            'success': True
            , 'message': 'Determinante obtida com sucesso.'
            , 'result': det
        }
```

calc_adjunta

Função que calcula a matriz adjunta de uma matriz de dimensão 2x2 ou 3x3.

```
def calc_adjunta(matriz):
    lin, col = matriz.shape
    tamanho = check_tamanho_matriz(matriz)

    if not tamanho.get('success'):
        return {
            'success': False
            , 'message': tamanho.get('message')
            , 'result': None
        }

    if (lin == 2):
        adj = np.array([
                [matriz[1][1], -matriz[1][0]],
                [-matriz[0][1], matriz[0][0]]
            ])

    if (lin == 3):
        adj = np.zeros((3,3))

        for i in range(lin):
            for j in range(col):
                adj[i][j] = calc_determinante(np.array([
                    [matriz[(i+1)%3][(j+1)%3],
                        matriz[(i+1)%3][(j+2)%3]],
                    [matriz[(i+2)%3][(j+1)%3],
                        matriz[(i+2)%3][(j+2)%3]]
                ])).get('result', None)

    return {
            'success': True
            , 'message': 'Adjunta obtida com sucesso.'
            , 'result': np.transpose(adj)
        }
```

calc_inversa

Função que calcula a matriz inversa de uma matriz de dimensão 2x2 ou 3x3.

```
def calc_inversa(matriz):
    lin, col = matriz.shape
    tamanho = check_tamanho_matriz(matriz)

    if not tamanho.get('success'):
        return {
            'success': False
            , 'message': tamanho.get('message')
            , 'result': None
        }

    return {
        'success': True
        , 'message': 'Inversa obtida com sucesso'
        , 'result':
            (1/calc_determinante(matriz).get('result')) *
            calc_adjunta(matriz).get('result')
    }
```

calc_min_quadrado

Função que calcula a matriz inversa de uma matriz de dimensão 2x2 ou 3x3.

```
def calc_min_quadrado(X, y, quadratico=False):
    lin, col = X.shape

    if (quadratico):
        X = np.array([np.append(i, i[col-1]*i[col-1]) for i in X])

    inv = calc_inversa(np.dot(np.transpose(X), X))

    if not inv.get('success'):
        return {
            'success': False
            , 'message': 'Nao foi possivel calcular o minimo '+
                         'quadrado.\n {}'.format(inv.get('message'))
            , 'result': None
        }

    return {
        'success': True
        , 'message': 'Minimo quadrado {} calculado com '+
                     'sucesso.'.format(
                         '(quadratico)' if quadratico else ''
                     )
        , 'result': np.dot(
                        inv.get('result'),
                        np.dot(np.transpose(X), y)
                    )
    }
```

Testes

Para testar o funcionamento das funções foram utilizados os seguintes conjuntos de dados:

- Height x Shoe size;
- Boiling point at the Alps;
- Books x Grades;
- US Census.

Resultados

Height x Shoe size

Mínimo quadrado linear.

$$\beta = \begin{bmatrix} -25.65123789 \\ 0.51453175 \end{bmatrix}$$

Mínimo quadrado quadrático.

$$\beta = \begin{bmatrix} -4.21942130e+06 \\ -5.40741892e+00 \\ 4.28048409e-02 \end{bmatrix}$$

Boiling point at the Alps

Mínimo quadrado linear.

$$\beta = \begin{bmatrix} -81.06372713 \\ 0.5228924 \end{bmatrix}$$

Mínimo quadrado quadrático.

$$\beta = \begin{bmatrix} 3.88292988e+01 \\ -6.54770851e-01 \\ 2.88971271e-03 \end{bmatrix}$$

Books x Grades

Mínimo quadrado linear.

$$\beta = \begin{bmatrix} 37.3791852 \\ 4.03689261 \\ 1.28347727 \end{bmatrix}$$

Não foi possível obter o mínimo quadrado quadrático desse conjunto de dados tendo em vista que o mesmo possúi duas variáreis de entrada (x_1 e x_2). Isso faz com que, caso eleve uma de suas variáreis ao quadrado, a matriz resultante da multiplicação ($X^T X$) resultará em uma matriz de dimensão maior que 3x3, impossibilitando o algorítmo de funcionar.

US Census

Mínimo quadrado linear.

$$\beta = \begin{bmatrix} -3.78394559e+03 \\ 2.02530273e+00 \end{bmatrix}$$

Mínimo quadrado quadrático.

$$\beta = \begin{bmatrix} -2.44524585e+09 \\ 2.75670665e+09 \\ 6.95532552e+05 \end{bmatrix}$$

Conclusão

Nesse relatório foi apresentado como se obter o mínimo quadrado de um conjunto de dados.

Foi implementado, em linguagem Python, um algorítmo para executar esse método e, para testar o algorítimo, foram utilizados 4 conjuntos de dados diferentes.