CENTRO UNIVERSITÁRIO FEI

VICTOR BIAZON
RA: 119.115-4

RELATÓRIO I – TÓPICOS ESPECIAIS DE APRENDIZAGEM
REGRESSÃO LINEAR

SÃO BERNARDO DO CAMPO
2019

Sumário:

1. Objetivo

Implementar o algoritmo regressão linear pelo método dos mínimos quadrados com a implementação básica, quadrática e robusta e aplicando-os para os datasets propostos.

2. Teoria

Regressão Linear

A regressão linear trate-se de uma abordagem simples de aprendizado supervisionado e avalia a relação linear de uma variável com sua variável dependente. Comumente se utiliza o método dos mínimos quadrados para fazer a aproximação dos coeficientes que representam a reta que melhor aproxima a linearidade da variável dependente a partir de valores conhecidos (pontos). [1].

Metodo dos mínimos quadrados

O método dos mínimos quadrados trata-se de utilizar o vetor da variável independente X, juntamente com o vetor correspondente da variável independente Y para calcular os coeficientes B0 e B1 que descrevem a reta que melhor aproxima a linearidade dos dados.

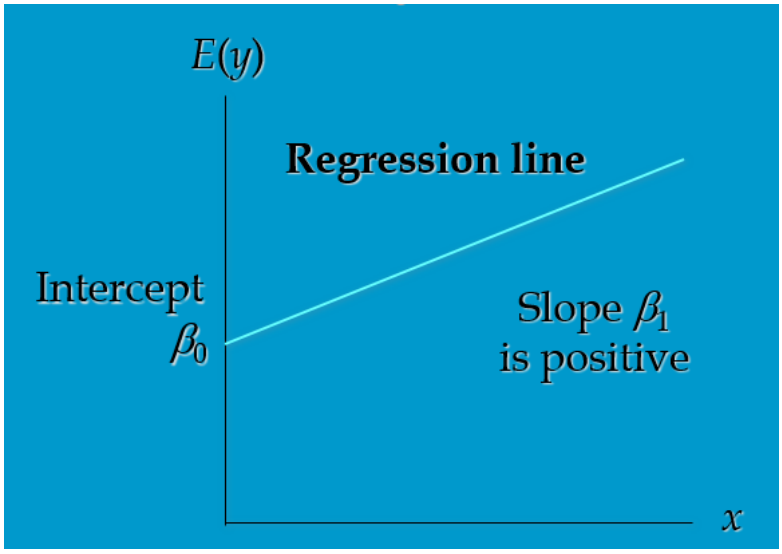


Figura 1: Exemplo de regressão linear simples. [2]

O critério a ser seguido é minimizar a diferença entre o Y real e o Y previsto conforme formula abaixo:

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

E para encontrar os coeficientes B devemos realizar a seguinte conta na forma matricial:

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

Após encontrar os coeficientes B podemos realizar predição de valores utilizando-os da seguinte forma:

$$\hat{y}(x_0) = x_0^T\hat{\beta}$$

Sendo que o Y previsto é a multiplicação matricial do X que deseja ser realizada a predição multiplicado pelos coeficientes B.

A regressão linear simples considera apenas dois coeficientes um que representa uma constante e um que representa a inclinação da reta.

$$\hat{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}$$

Regressão Linear Quadrática

A regressão quadrática segue os mesmos passos da simples, no entanto é considerado um terceiro coeficiente B2, que representa a curvatura da curva aproximada pelos coeficientes sendo este o coeficiente quadrático.

$X = \begin{bmatrix} 1 & x & x^2 \end{bmatrix}$

β_0 = Intercept

β_1 = Linear coefficient

β_2 = Quadratic coefficient

Regressão Linear Robusta

A regressão linear simples é muito influenciada pelos pontos chamados “outliers” que são pontos “fora da curva”, ou seja, pontos que apresentam pouca correlação com o curva de regressão e por estarem longe da “média” deslocam a curva pois tem grande resíduo quadrático entre o valor previsto e o valor real. Pra evitar tal influência negativa a regressão robusta utiliza de pesos atribuídos a cada

ponto para representar sua importância para o cálculo dos coeficientes, sendo que quanto mais distante da média dos valores, menos peso o ponto tem.

Os pesos são calculados realizando uma regressão linear simples e logo após calculando cada peso pela seguinte formula:

$$w_i = \left| 1 \middle/ \left(y_i - \hat{y}_i \right) \right.$$

Após realizado o cálculo dos pesos, se recalcula os coeficientes B de forma a readequar a curva pela seguinte formula:

$$b = \hat{\beta} = (X^T \times W \cdot X)^{-1} X^T \times W \cdot y$$

Nesta equação os pesos W multiplicam ponto a ponto seus respectivos X e y correspondentes a sua posição no vetor, ou seja, o Wi multiplica o valor de Xi e Yi, Wi+1 multiplica Xi+1 e Yi+1 e assim por diante. Desta forma os B são recalculados e a curva é adequada para considerar cada ponto com uma relevância diferente para a predição de valores.

3. Implementação

A implementação partiu do seguinte fluxo:

```
def LinearRegression(data): #regressão linear simples
    X = np.asarray(data.iloc[:,:-1]) #separa as variáveis independentes no vetor X
    y = np.asarray(data.iloc[:,-1:]) #separa as variáveis dependentes no vetor Y
    XBeta = np.c_[np.ones((len(X),1), float), X] #adiciona vetor de 1's no vetor X
    XBetat = MatrixTranspose(XBeta) #transpõe o vetor XBeta para executar a multiplicação
    Xtx = MultMatrix(XBetat,XBeta) #multiplica XBetat por XBeta
    Xty = MultMatrix(XBetat,y) #multiplica XBetat pelo vetor y
    Beta = MultMatrix(InverseMatrix(Xtx), Xty) #Multiplica a inversa do vetor Xtx pelo vetor Xty
```

Com estes passos que seguem a formula:

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

Se calcula os coeficientes beta que representam a curva da regressão.

Para a regressão quadrática, os passos são exatamente os mesmos, no entanto se adiciona uma coluna em X que representa o X^2 para calcular o B2 que representa a curvatura da curva de regressão:

```
def QuadLinearRegression(data): # regressão linear quadrática
    X = np.asarray(data.iloc[:,:-1]) #separa as variáveis independentes no vetor X
    y = np.asarray(data.iloc[:,-1:]) #separa as variáveis dependentes no vetor Y
```

```
    XBeta = np.c_[np.ones((len(X),1), float), X, np.square(X)] #adiciona vetor de 1's no vetor X e vetor X² na coluna seguinte
    XBetat = MatrixTranspose(XBeta) #transpõe o vetor XBeta para executar a multiplicação
    Xtx = MultMatrix(XBetat,XBeta) #multiplica XBetat por XBeta
    Xty = MultMatrix(XBetat,y) #multiplica XBetat pelo vetor y
    Beta = MultMatrix(InverseMatrix(Xtx), Xty) #Multiplica a inversa do vetor Xtx pelo vetor Xty
```

Já para a regressão linear robusta é necessário um passo a mais sendo que primeiramente se calcula a regressão simples e a partir dela se calcula os pesos de cada ponto para recalcular os coeficientes Beta.

```
def RobustLinearRegression(data): #regressão linear robusta
    X = np.asarray(data.iloc[:,:-1]) #separa as variáveis independentes no vetor X
    y = np.asarray(data.iloc[:,-1:]) #separa as variáveis dependentes no vetor Y
    XBeta = np.c_[np.ones((len(X),1), float), X] #adiciona vetor de 1's no vetor X
    XBetat = MatrixTranspose(XBeta) #transpõe o vetor XBeta para executar a multiplicação
    Xtx = MultMatrix(XBetat,XBeta) #multiplica XBetat por XBeta
    Xty = MultMatrix(XBetat,y) #multiplica XBetat pelo vetor y
    Beta = MultMatrix(InverseMatrix(Xtx), Xty) #Multiplica a inversa do vetor Xtx pelo vetor Xty
    Wi = np.ones((len(X),1)) #cria vetor de pesos
    WX = np.zeros_like(X, dtype = float) #cria vetor de X ponderado pelos pesos
    WY = np.zeros_like(y, dtype = float) #cria vetor de Y ponderado pelos pesos
    Y_pred = MultMatrix(XBeta, Beta) #realiza predição inicial para comparar com os valores de Y reais
    for l in range(0,1):
        for k in range(0,len(X)):
```

```
                Wi[k] = abs(1/(y[k] - Y_pred[k])) #calcula o peso Wi do elemento k
                WY[k] = Wi[k] * y[k] #multiplica o elemento y correspondente ao
peso W por este
                for j in range(0, len(X[0])):
                    WX[k,j] = Wi[k] * X[k,j] #multiplica o elemento X correspondente
ao peso W por este
            XBeta = np.c_[Wi, WX] #cria novo vetor XBeta
            XBetat = MatrixTranspose(XBeta) #transpõe o vetor XBeta
            Xtx = MultMatrix(XBetat,XBeta)
            Xty = MultMatrix(XBetat, WY)
            Beta = MultMatrix(InverseMatrix(Xtx), Xty)    #calcula novos
coeficientes Beta
```

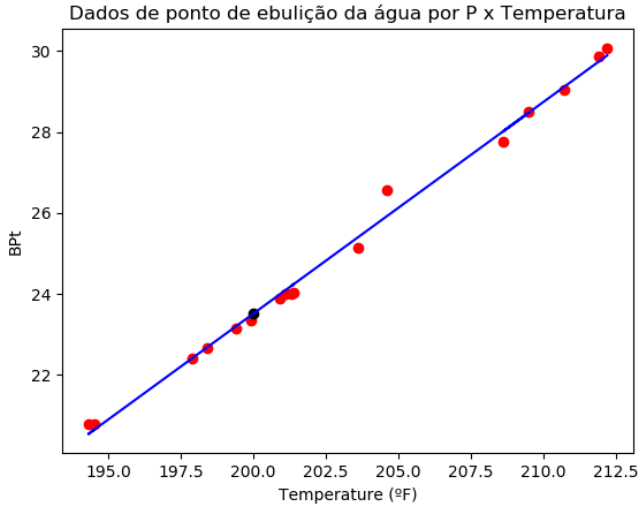
4. Resultados

Para testar os algoritmos foram implementados em três datasets:

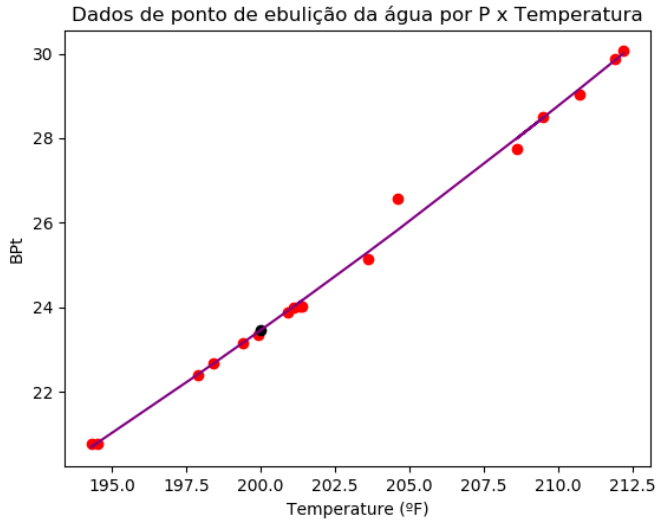
Alpswater, que representa dados de ponto de ebulição da água a diferentes pressões atmosféricas, US Census, que representa o crescimento da população americana ao longo dos anos, e Books Vs Grades, que representa as notas de alunos de uma escola de acordo com o número de empréstimos de livros e presença em sala.

AlpsWater:

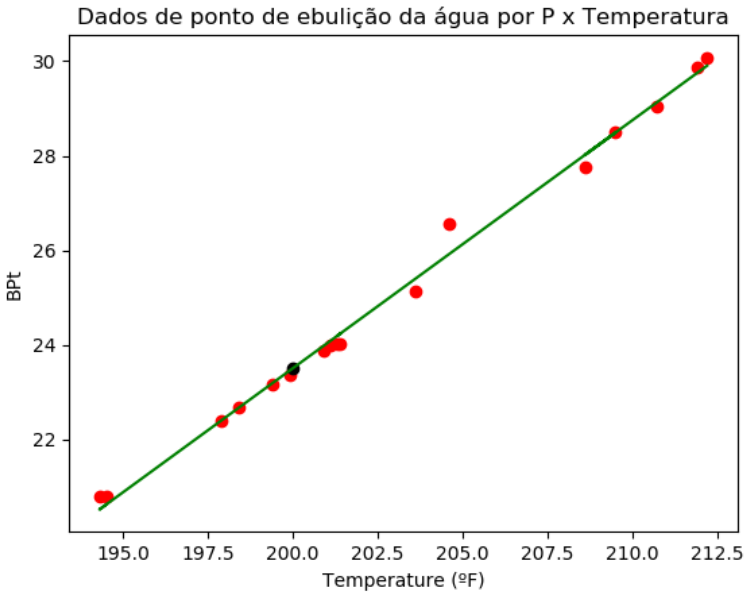
Regressão Linear Simples: Teve como predição para a temperatura de 200 ºF a pressão de 23.51 BPt.



Regressão Linear Quadrática: Teve como predição para a temperatura de 200 ºF a pressão de 23.46 BPt.

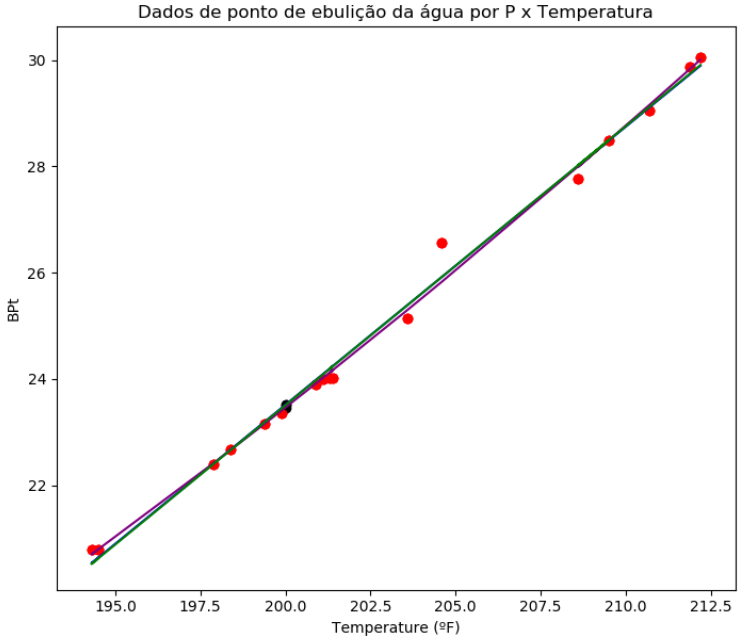


Regressão Linear Robusta: Teve como predição para a temperatura de 200 ºF a pressão de 24.5 BPt.



Como pode-se notar a diferença entre o robusto e o linear simples não foi expressiva devido a haver poucos ou inexpressivos “outliers”.

A imagem a seguir mostra todas a curvas de predição em um só gráfico:



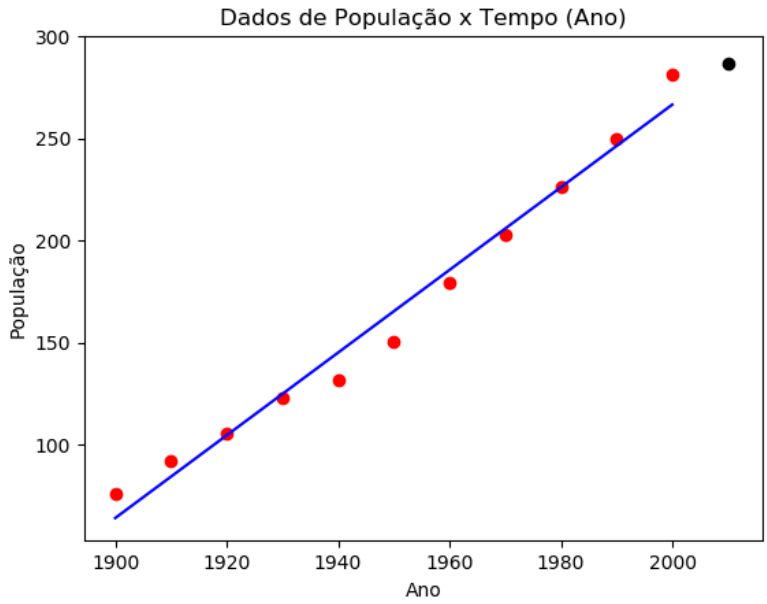
Como pode ser visto, os valores divergem pouco para este dataset embora a regressão linear quadrática tenha leve curvatura.

Para o dataset US Census os seguintes gráficos e predições foram gerados para o ano 2010.

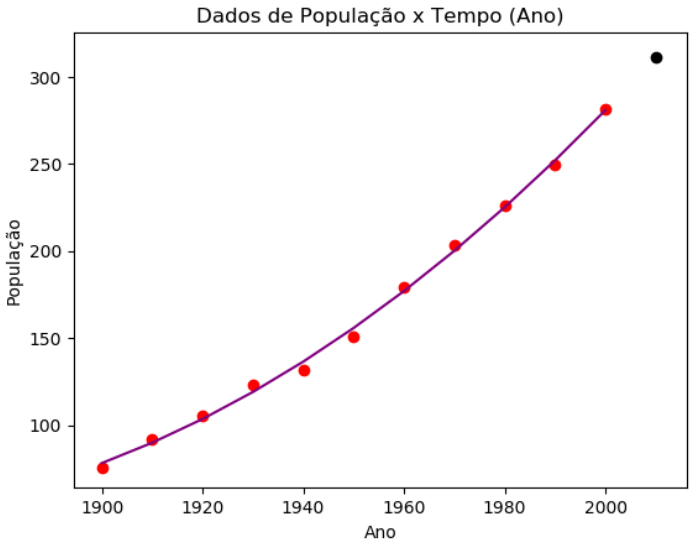
```
X0(2010) =  [[286.91289091]]
X0(2010) =  [[311.58807136]]
X0(2010) =  [[286.902]]
```

Os dados acima são as predições a partir dos modelos gerados respectivamente para o linear simples, quadrático e o robusto. Sendo que o valor real é de 308,74. Ou seja, o quadrático foi o que teve melhor performance neste caso.

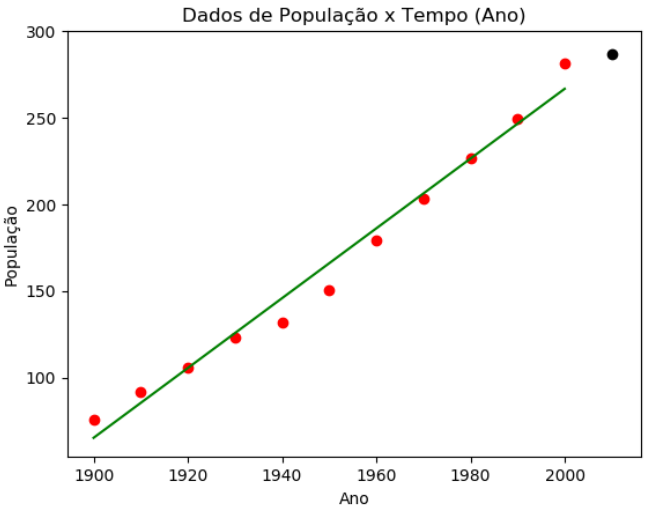
Regressão Linear Simples:



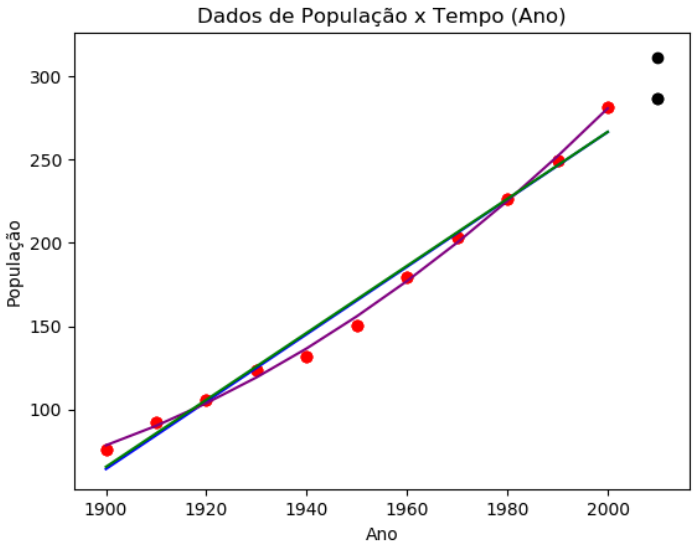
Regressão Linear Quadrática:



Regressão Linear Robusta:



Gráficos sobrepostos:



Neste caso pode-se notar clara curvatura dos dados amostrais sendo que a regressão linear quadrática neste caso é a mais indicada para os dados.

Para o dataset Books_Attend_Grades não foi possível plotar claramente gráficos pois havia mais de uma variável inpedente, portanto será mostrado abaixo a comparação dos valores reais que foram utilizados para treinar os modelos e suas predições baseadas nas variáveis independentes.

Notas			
Real	Reg. Simples	Reg. Quad.	Reg. Robusta
45	48,9305	49,1054	47,9122
57	60,6682	58,1309	61,2182
45	50,214	48,8327	49,4776
51	65,9886	64,5496	66,6968
65	66,3615	65,6623	65,1302
88	79,1963	81,2259	80,785
44	55,5343	54,5659	54,9563
87	79,1963	81,2259	80,785
89	68,742	66,5457	69,0444
59	56,6313	52,4577	57,305
66	55,7208	60,0799	54,173
65	58,1013	55,6833	58,0872
56	76,6293	75,4528	77,654
47	54,2509	54,506	53,3908
66	47,647	49,7106	46,3467
41	54,2509	54,506	53,3908
56	70,0255	68,2683	70,6099
37	51,4974	48,8926	51,0431
45	65,8021	67,0167	67,4801
58	68,9285	66,1147	68,2611
47	67,645	65,7222	66,6957
64	61,7653	61,3435	63,5669
97	64,7051	62,8269	65,1313
55	68,742	66,5457	69,0444
51	67,0856	70,0696	69,0456
61	45,08	51,9187	43,2157

69	68,742	66,5457	69,0444
79	73,8759	75,4316	75,3063
71	63,4217	61,4368	63,5658
62	62,1382	60,3793	62,0003
87	71,309	70,3235	72,1754
54	71,1225	74,7656	72,9587
43	59,5712	59,2619	58,8694
92	75,1594	78,4844	76,8718
83	79,1963	81,2259	80,785
94	79,1963	81,2259	80,785
60	61,0412	63,1934	59,6516
56	51,6839	55,3839	50,2598
88	65,9886	64,5496	66,6968
62	50,214	48,8327	49,4776

Como pode-se notar a que tiveram os valore mais próximos dos reais foi a quadrática.

5. Conclusão

Com estes experimentos pode-se verificar que existe uma necessidade de pré avaliação dos dados a serem ajustados na regressão a fim de determinar qual o melhor regressor para cada caso, sendo que nem sempre os dados seguem uma distribuição linear simples, pode ser necessário utilizar a regressão linear quadrática ou mesmo a robusta para eliminar outliers das considerações do gráfico e do regressor.

6. Referências bibliográficas

[1] SELTMANN, Howard; **Experimental Design for Behavioral and Social Sciences, Disponível em:** http://www.stat.cmu.edu/~hseltman/309/Book/chapter9.pdf Acesso: 25/10/2019 21:34

[2] Hastie, T.; Tibshirani, R.; Friedman, J. **The Elements of Statistical Learning: Data Mining, Inference, and Prediction.** Berlin: Springer, 2001.