

## Trabalho 2: Aprendizado de Máquina

Augusto César Muniz Lopes

### Tema Tipo 8: RBF aplicada à Regressão (Algoritmo totalmente supervisionado).

#### Dados Utilizados:

Utilizei da biblioteca *sklearn* para gerar aleatoriamente um *dataset*, utilizando a função *make\_regression*, com 200 amostras, 4 features e um ruído de 0.1, como demonstrado na imagem a seguir:

```
# Gerar um dataset
X, y = make_regression(n_samples=200, n_features=4, noise=0.1, random_state=109)
```

Em sequência, usei uma função para normalizar os dados (*from sklearn.preprocessing import StandardScaler*), visto que em meus testes iniciais o erro estava resultando em alguns valores altos, o uso da normalização é demonstrado na imagem abaixo:

```
# Normalizar X e y
scaler_X = StandardScaler()
X = scaler_X.fit_transform(X)
scaler_y = StandardScaler()
y = scaler_y.fit_transform(y.reshape(-1, 1)).flatten()
```

#### Valores Iniciais:

\*Observação: Vi em algumas literaturas, que uma das formas de determinar o melhor centro (*m*) seria utilizando o *K-Means*. Porém, como o enunciado do trabalho especifica claramente a restrição de não utilizar algoritmos como “*K-Means, clusterdata, ksdensity, k-NN, PCA*”, optei por não aplicá-lo, mesmo que o foco do meu trabalho não seja sobre esse algoritmo.

Dada a observação, para os centros (*m*), utilizei uma função para gerar os valores aleatórios dentro das observações do *dataset* e escolhi de forma estática aquela que apresentou o menor erro ao final das iterações. Então, com os 12 neurônios (*H*) na camada oculta, *m* apresentou uma lista de 12 listas e 4 colunas (quatro features).

Para os pesos (*w*) escolhi uma abordagem semelhante a *m*, mas apenas gerando aleatoriamente valores próximos aos vistos no *dataset*.

Para o *bias* (*w0*) iniciei seu valor com zero.

Para o espalhamento (*s*) comecei com 0.5;

A taxa de aprendizado (*n*) com 0.1;

E o número de épocas (definida como *epoca* dentro do primeiro laço) como 100 (resultados melhores foram obtidos com um número maior de épocas, mas para evitar tempo de espera optei por deixar em 100 mesmo).

#### Configuração ideal

Com certeza existem configurações melhores para resultados mais precisos com o *dataset*, seja aplicando o *kmeans* para descobrir os centro ideais, ou utilizando da distância média para um melhor espalhamento, ou aumentando o número de neurônios na camada oculta, ou treinando com um número maior de épocas. No entanto, a seguir vou demonstrar algumas imagens comparando os valores dos *erros* encontrados no meu resultado ideal (encontrado manualmente em alguns testes), com alguns outros valores iniciais.

Meu Resultado ideal (dados e resultado nas duas imagens abaixo):

```
# Neuronios da camada oculta (H)
H = 12

# Centros (m):
m = [
    [0.69321602, -0.83645056, 0.46356806, 1.0588505],
    [0.83322947, 0.6626997, -0.14475505, 0.15257985],
    [-1.16554823, 0.20852695, -0.0820158, -0.5065535],
    [-1.03920593, 0.48854546, 0.75655635, -2.54265183],
    [-1.02496614, -0.85763018, -0.45610099, 0.33885959],
    [1.60826194, -0.42876896, 1.21270681, -0.01765857],
    [0.87782818, 0.0217352, 1.44171054, 0.23357005],
    [-0.70060563, 0.64368139, 1.24567399, -0.70215599],
    [-0.30198368, -0.80411294, 1.40516094, 0.38811918],
    [1.66519542, -1.00305709, -1.33229093, 0.81620114],
    [-0.59933721, -0.88871029, 2.70523491, -1.03686115],
    [0.22861962, -0.87257873, 0.25379815, -0.09882513]
]
#m = X[np.random.choice(len(X), H, replace=False)]
#print("m=>", m)

# Estpalhamento (sh):
s = 0.5

# Pesos (w):
w = [-1.59088316, 0.5591758, 1.12198116, 0.62340975, 0.18093274, -0.70016163,
      0.83453995, 1.36510077, 0.66412175, 0.89119888, 0.61414482, 0.61751515]
#w = np.random.uniform(-2, 2, H)
#print("w=>", w)

# Bias (w0)
bias = 0.01

# Taxa de aprendizado (n)
n = 0.1

# Erro em batelada
erro_b = []

#### TREINAMENTO
for epoca in range(100):
    predic = [] # Saida dos meus neuronios
```

```
#####
Erro em batelada 3.820112355451071
#####
#####
Erro quadratico medio: 0.12398757943743285
Erro quadratico relativo: 0.4244418704180922
#####
```

Diminuindo a quantidade de neurônios para 10, há um pequeno aumento nos erros:

```
#####  
Erro em batelada 3.961978965599881  
#####  
#####  
Erro quadratico medio: 0.1524989329877839  
Erro quadratico relativo: 0.4775024397121143  
#####
```

Trocando os valores dos centros (para 10 neurônios) também pode haver uma aumento nos erros como demonstrado na imagem a seguir:

```
m=> [[ 0.09783608  0.37484693 -1.00556513 -0.73321804]  
[ 0.25925735  0.51621824  0.44837574 -1.58008644]  
[ 1.85063781 -0.80043425 -0.21245015  0.9812979 ]  
[-0.85295381 -2.21746724  0.86160896 -0.84490425]  
[ 0.94792779  0.61603402 -0.29988726 -0.67531751]  
[-0.35986204  0.52611955 -0.16128356  0.88484957]  
[ 0.05609479 -1.2295616  -0.48443025  0.53709501]  
[ 1.08923179  0.86322806  0.0921232  -1.69038105]  
[ 0.77440525 -1.23373475 -0.18349778 -0.37237444]  
[ 0.16748128 -1.59106335  0.83002801 -0.20790154]]  
#####  
Erro em batelada 3.9842562429500576  
#####  
#####  
Erro quadratico medio: 0.16888759303669681  
Erro quadratico relativo: 0.504206209270942  
#####
```

Trocando somente os pesos (12 neurônios e os pesos são aqueles apresentado inicialmente) também é possível notar um aumento nos valores dos erros.

```
w=> [-0.68958    -0.12283264  1.68212971 -1.91420791 -0.17501713 -0.02523364  
-1.48283342  1.51273089 -0.13648878  1.44422297 -0.27190792 -0.46892045]  
#####  
Erro em batelada 4.3167678968743965  
#####  
#####  
Erro quadratico medio: 0.16871263949460594  
Erro quadratico relativo: 0.5051097382029193
```

Para um espalhamento menor com  $s = 0.2$

```
#####  
Erro em batelada 4.907804980260524  
#####  
#####  
Erro quadratico medio: 0.18148430998013723  
Erro quadratico relativo: 0.5213462485127585  
#####
```

Com uma taxa de aprendizado bem menor em 0.01 e o valor da época em 1000 temos (s continuou como 0.2 nesse teste):

```
#####  
Erro em batelada 11.41252658889888  
#####  
#####  
Erro quadratico medio: 0.1556362180122778  
Erro quadratico relativo: 0.4058520201927062  
#####
```

Com os mesmos valores iniciais, modificando somente a época para 1000, temos uma melhoria significativa como observado abaixo:

```
#####  
Erro em batelada 1.1400197110978914  
#####  
#####  
Erro quadratico medio: 0.014607744899849819  
Erro quadratico relativo: 0.12930670717071013  
#####
```

\*Valores usados para testar outros centros e pesos foram descobertos usando funções de escolha aleatória, ambas funções estão comentadas no código.

\*Para comparar o valor previsto pelo modelo e o valor real, também foi comentado no código os prints que demonstram tais resultados como o da imagem abaixo

```
<<<<<( 55 )>>>>>  
y_pred=> 0.5376557815753529  
Resultado real (r)=> 0.5525407008255385
```

