Exercício 7

Arthur Felipe Reis Souza

April 29, 2024

1 Introduction

Os conjuntos de dados que serão avaliados são, respectivamente, Breast Cancer e Statlog. Ambos os datasets contém informações e características que irão contribuir parcialmente para predizer se uma pessoa tem ou não tem alguma doença. Para isso, serão utilizados modelos de redes neurais do tipo ELM (Extreme Learning Machine) ou RBF (Radial Basis Function).

1.1 StatLog

O dataset StatLog contém várias características e informações de 270 pacientes, essas características serão utilizadas para predizer se os pacientes de teste tem ou não doença de coração. Primeiramente, o conjunto de dados foi pré processado, ou seja, as colunas que continham características discrepantes e que atrapalhavam a convergência da rede foram transformadas. Foi utilizada uma função da biblioteca skilearn que desloca a função para a origem e a escala para ter um desvio padrão de 1 :

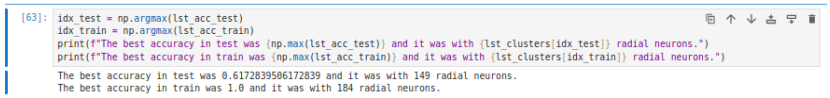
$$x_i = \frac{x - \mu}{\sigma}$$

Com os dados pré processados, o algoritmo KMeans com K = 10 foi utilizado, portanto terá 10 funções radiais na camada intermediária da rede RBF.

```
[6]: from sklearn.cluster import KMeans
     kmeans = KMeans(n_clusters = 10, random_state = 0, n_init = 'auto').fit(X_train)
     cluster_labels = kmeans.labels_
     cluster_centers = kmeans.cluster_centers_

[7]: import train_RBF
     ret_ = train_RBF.trainRBF(X_train, y_train, cluster_centers.shape[0], 0.6)
     H shape : (189, 10)
     Haug shape : (189, 11)
     yin shape : (189,)
     W shape : (11,)
```

Após treinar a rede, um conjunto de 81 dados foram utilizados para teste. Foram criados vários modelos, e cada um foi testado obtendo a acurácia do

mesmo. Esse algoritmo é chamado de Grid Search Cross-Validation, e tem por objetivo, dado um conjunto e possíveis parâmetros, selecionar o modelo que teve o melhor desempenho sobre os dados de teste. Portanto, a ideia do Grid Search aqui foi de encontrar o melhor valor de neurônios radiais para os quais uma boa performance em uma dada métrica de validação é satisfeita.



Analisando o resultado acima, é possível afirmar que 149 neurônios radiais nos darão uma melhor performance dado um conjunto de dados de teste. Outro ponto é, utilizando mais neurônios radiais, o modelo terá uma acurácia de 100% sobre os dados de treino, mas não generalizará bem para os dados de teste. Isso acontecerá porque o modelo neuronal do tipo RBF, com uma grande quantidade de neurônios na camada intermediária, tenderá a sobreajustar (overfitting) aos dados.

1.2 Breast Cancer

O dataset Breast Cancer contém várias características e informações de 699 pacientes, essas características serão utilizadas para predizer se os pacientes de teste tem um tumor maligno ou benigno. Primeiramente, o conjunto de dados foi pré processado, ou seja, as colunas que continham características discrepantes e que atrapalhavam a convergência da rede foram transformadas.

Com os dados pré processados, o algoritmo KMeans com $K = 10$ foi utilizado, portanto terá 10 funções radiais na camada intermediária da rede RBF.

Após treinar a rede, um conjunto de 210 dados foram utilizados para teste. Foram criados vários modelos, e cada um foi testado obtendo a acurácia do mesmo. Esse método é chamado de Grid Search Cross-Validation, e tem por objetivo, dado um conjunto e possíveis parâmetros, selecionar o modelo que teve o melhor desempenho sobre os dados de teste. Portanto, a ideia do Grid Search aqui foi de encontrar o melhor valor de neurônios radiais para os quais uma boa performance em uma dada métrica de validação é satisfeita.

[77]:
```
from sklearn.cluster import KMeans
import train_RBF
from sklearn.metrics import accuracy_score

lst_acc_test = list()
lst_acc_train = list()
lst_clusters = list()

for i in range(1, 300, 1):
    kmeans = KMeans(n_clusters = i, random_state = 0, n_init = 'auto').fit(X_train)
    cluster_labels = kmeans.labels_
    cluster_centers = kmeans.cluster_centers_
    ret_ = train_RBF.trainRBF(X_train, y_train, cluster_centers.shape[0], 0.6)

    yhat_test = train_RBF.y_RBF(X_test, ret_)
    yhat_test[yhat_test <= -0.5] = -1
    yhat_test[yhat_test > -0.5] = 1
    yhat_train = train_RBF.y_RBF(X_train, ret_)
    yhat_train[yhat_train <= -0.5] = -1
    yhat_train[yhat_train > -0.5] = 1

    acc_test = accuracy_score(y_test, yhat_test)
    acc_train = accuracy_score(y_train, yhat_train)
    lst_acc_test.append(acc_test)
    lst_acc_train.append(acc_train)
    lst_clusters.append(i)
```

```
W shape : (10,)
H shape : (489, 10)
Haug shape : (489, 11)
yin shape : (489,)
W shape : (11,)
H shape : (489, 11)
Haug shape : (489, 12)
yin shape : (489,)
W shape : (12,)
H shape : (489, 12)
Haug shape : (489, 13)
yin shape : (489,)
W shape : (13,)
H shape : (489, 13)
Haug shape : (489, 14)
yin shape : (489,)
W shape : (14,)
H shape : (489, 14)
```

[78]:
```
idx_test = np.argmax(lst_acc_test)
idx_train = np.argmax(lst_acc_train)
print(f"The best accuracy in test was {np.max(lst_acc_test)} and it was with {lst_clusters[idx_test]} radial neurons.")
print(f"The best accuracy in train was {np.max(lst_acc_train)} and it was with {lst_clusters[idx_train]} radial neurons.")
```

```
The best accuracy in test was 0.9476190476190476 and it was with 50 radial neurons.
The best accuracy in train was 1.0 and it was with 215 radial neurons.
```

Analisando o resultado acima, é possível afirmar que 50 neurônios radiais nos darão uma melhor performance dado um conjunto de dados de teste. Usando 215 neuronios na camada intermediária, a acurácia nos dados de treino é de 100%. O segundo resultado é esperado, considerando-se que um grande número de neurônios na camada intermediária tenderá o modelo à um overfitting.

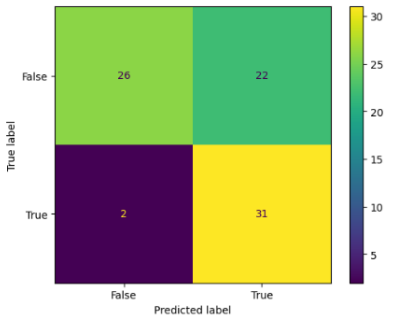
2 Centros aleatórios

A lógica para escolher os centros e os raios aleatórios são : os pontos centrais das funções radiais serão exatamente o primeiro e o terceiro quartile. O primeiro quartile é a mediana do conjunto abaixo da mediana, enquanto o terceiro quartile é a mediana do conjunto acima da mediana. O raio será a metade da distância entre os quartiles. Após realizar o treinamento da rede neuronal RBF com esses valores de centro, foram obtidos os resultados abaixo :

2.1 StatLog

Para o dataset statlog utilizando os 2 centros sendo, respectivamente o primeiro e o terceiro quartile, obtemos a acurácia média e o erro (desvio padrão):

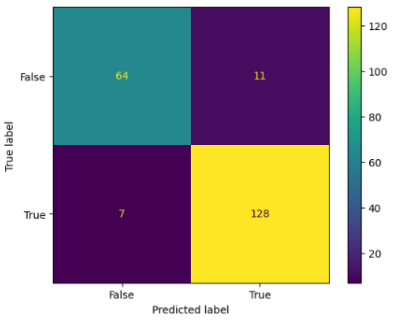
```
[63]: print(f"The mean accuracy in test was ({mean_acc} +- {stand_dev}).")
      The mean accuracy in test was (0.7037037037037039 +- 2.465190328815662e-33).

[60]: from sklearn import metrics
      confusion_matrix = metrics.confusion_matrix(y_test, yhat_test_rand_)
      cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = [False, True])
      #plt.title(' TRAIN DATA RESULT')
      cm_display.plot()
      plt.show()
```



2.2 Breast Cancer

Para o Breast statlog utilizando os 2 centros sendo, respectivamente o primeiro e o terceiro quartile, obtemos a acurácia média e o erro (desvio padrão):

```
[100]: print(f"The mean accuracy in test was ({mean_acc} +- {stand_dev}).")
       The mean accuracy in test was (0.9142857142857144 +- 6.162975822039155e-34).

[94]: from sklearn import metrics
      confusion_matrix = metrics.confusion_matrix(y_test, yhat_test_rand_)
      cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = [False, True])
      #plt.title(' TRAIN DATA RESULT')
      cm_display.plot()
      plt.show()
```



3 Statlog

Por fim, após realizar em um Grid Search o teste de N valores de K clusters, foram obtidos as seguintes aproximações em relação ao modelo com funções radiais arbitrárias :

3.1 Statlog

Resultado semelhante, com N neuronios :

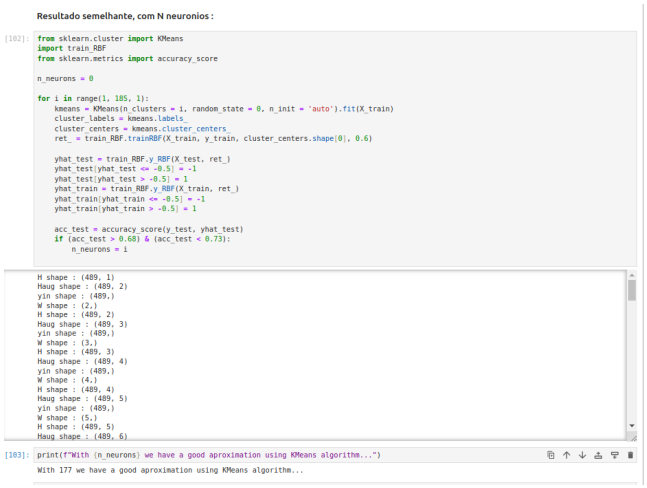
```
[64]: from sklearn.cluster import KMeans
      import train_RBF
      from sklearn.metrics import accuracy_score

      n_neurons = 0

      for i in range(1, 185, 1):
          kmeans = KMeans(n_clusters = i, random_state = 0, n_init = 'auto').fit(X_train)
          cluster_labels = kmeans.labels_
          cluster_centers = kmeans.cluster_centers_
          ret_ = train_RBF.trainRBF(X_train, y_train, cluster_centers.shape[0], 0.6)

          yhat_test = train_RBF.y_RBF(X_test, ret_)
          yhat_test[yhat_test <= -0.5] = -1
          yhat_test[yhat_test > -0.5] = 1
          yhat_train = train_RBF.y_RBF(X_train, ret_)
          yhat_train[yhat_train <= -0.5] = -1
          yhat_train[yhat_train > -0.5] = 1

          acc_test = accuracy_score(y_test, yhat_test)
          if (acc_test > 0.68) & (acc_test < 0.73):
              n_neurons = i

      print(f"With {n_neurons} we have a good aproximation using KMeans algorithm...")
```

```
H shape : (189, 2)
Haug_shape : (189, 3)
yin shape : (189,)
W shape : (3,)
H shape : (189, 3)
Haug_shape : (189, 4)
yin shape : (189,)
W shape : (4,)
H shape : (189, 4)
Haug_shape : (189, 5)
yin shape : (189,)
W shape : (5,)
H shape : (189, 5)
Haug_shape : (189, 6)
yin shape : (189,)
W shape : (6,)
H shape : (189, 6)
Haug_shape : (189, 7)
```

```
[65]: print(f"With {n_neurons} we have a good aproximation using KMeans algorithm...")
      With 183 we have a good aproximation using KMeans algorithm...
```

Como pode ser observado acima, serão necessários 183 neurônios na camada intermediária de uma rede RBF usando o algoritmo KMeans.

3.2 Breast Cancer



Resultado semelhante, com N neuronios :

[102]:
```
from sklearn.cluster import KMeans
import train_RBF
from sklearn.metrics import accuracy_score

n_neurons = 0

for i in range(1, 185, 1):
    kmeans = KMeans(n_clusters = i, random_state = 0, n_init = 'auto').fit(X_train)
    cluster_labels = kmeans.labels_
    cluster_centers = kmeans.cluster_centers_
    ret_ = train_RBF.trainRBF(X_train, y_train, cluster_centers.shape[0], 0.6)

    yhat_test = train_RBF.y_RBF(X_test, ret_)
    yhat_test[yhat_test <= -0.5] = -1
    yhat_test[yhat_test > -0.5] = 1
    yhat_train = train_RBF.y_RBF(X_train, ret_)
    yhat_train[yhat_train <= -0.5] = -1
    yhat_train[yhat_train > -0.5] = 1

    acc_test = accuracy_score(y_test, yhat_test)
    if (acc_test > 0.68) & (acc_test < 0.73):
        n_neurons = i
```

```
H shape : (489, 1)
Haug shape : (489, 2)
yin shape : (489,)
W shape : (2,)
H shape : (489, 2)
Haug shape : (489, 3)
yin shape : (489,)
W shape : (3,)
H shape : (489, 3)
Haug shape : (489, 4)
yin shape : (489,)
W shape : (4,)
H shape : (489, 4)
Haug shape : (489, 5)
yin shape : (489,)
W shape : (5,)
H shape : (489, 5)
Haug shape : (489, 6)
```

[103]:
```
print(f"With {n_neurons} we have a good aproximation using KMeans algorithm...")
```

```
With 177 we have a good aproximation using KMeans algorithm...
```

Como pode ser observado acima, serão necessários 177 neurônios na camada intermediária de uma rede RBF usando o algoritmo KMeans.