ajuste-de-redes-neuronales

September 12, 2023

#Ajuste de redes neuronales

Ernesto Reynoso Lizárraga A01639915

[61]:
```
import pandas as pd
import numpy as np
from sklearn.neural_network import MLPRegressor, MLPClassifier
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import GridSearchCV, cross_val_predict,␣
 ↪StratifiedKFold, KFold
from sklearn.metrics import classification_report
from sklearn.metrics import mean_squared_error, mean_absolute_error
```

0.1 Ejercicio 1

[63]:
```
df = pd.read_csv('/content/drive/MyDrive/Inteligencia Artificial/crime_data.
 ↪csv')
df.head()
```

[63]:

	State	VR	MR	M	W	H	P	S
0	AK	761	9.0	41.8	75.2	86.6	9.1	14.3
1	AL	780	11.6	67.4	73.5	66.9	17.4	11.5
2	AR	593	10.2	44.7	82.9	66.3	20.0	10.7
3	AZ	715	8.6	84.7	88.6	78.7	15.4	12.1
4	CA	1078	13.1	96.7	79.3	76.2	18.2	12.5

[64]:
```
df = df.drop(['State','VR','P'],axis=1)
df.head()
```

[64]:

	MR	M	W	H	S
0	9.0	41.8	75.2	86.6	14.3
1	11.6	67.4	73.5	66.9	11.5
2	10.2	44.7	82.9	66.3	10.7
3	8.6	84.7	88.6	78.7	12.1
4	13.1	96.7	79.3	76.2	12.5

[65]:
```
#Variables regresoras
x = np.array(df[['M','W','H','S']])
```

```
y = np.array(df['MR'])
scaler = StandardScaler()
x = scaler.fit_transform(x)
```

0.1.1 1. Evalúa con validación cruzada un modelo pereceptrón multicapa para las variables que se te asignaron para este ejercicio.

[66]:
```
# Train classifier with all the available observations
clf = MLPRegressor(hidden_layer_sizes=(10, 10), max_iter=10000)
clf.fit(x, y)
# 5-fold cross-validation
kf = KFold(n_splits=5, shuffle=True)
cv_mse = []
cv_mae = []
for train_index, test_index in kf.split(x):
    x_train, x_test = x[train_index], x[test_index]
    y_train, y_test = y[train_index], y[test_index]

    regressor_cv = MLPRegressor(hidden_layer_sizes=(10, 10), max_iter=10000)
    regressor_cv.fit(x_train, y_train)

    y_pred = regressor_cv.predict(x_test)
    mse = mean_squared_error(y_test, y_pred)
    cv_mse.append(mse)
    mae = mean_absolute_error(y_test, y_pred)
    cv_mae.append(mae)

# Calcular el error cuadrático medio promedio
mse = np.mean(cv_mse)
mae = np.mean(cv_mae)
print("MSE:", mse, "\nMAE:", mae)
```

```
MSE: 56.767396573002614
MAE: 3.105378588600903
```

0.1.2 2. Agrega al conjunto de datos columnas que representen los cuadrados de las variables predictoras (por ejemplo, M2, W2), así como los productos entre pares de variables (por ejemplo, PxS, MxW). Evalúa un modelo perceptrón multicapa para este nuevo conjunto de datos.

[67]:
```
df['M2'] = df['M']**2
df['W2'] = df['W']**2
df['H2'] = df['H']**2
df['S2'] = df['S']**2

df['MW'] = df['M'] * df['W']
df['MH'] = df['M'] * df['H']
```

```
df['MS'] = df['M'] * df['S']
df['WH'] = df['W'] * df['H']
df['WS'] = df['W'] * df['S']
df['HS'] = df['H'] * df['S']
df.head()
```

[67]:

	MR	M	W	H	S	M2	W2	H2	S2	MW	\
0	9.0	41.8	75.2	86.6	14.3	1747.24	5655.04	7499.56	204.49	3143.36	
1	11.6	67.4	73.5	66.9	11.5	4542.76	5402.25	4475.61	132.25	4953.90	
2	10.2	44.7	82.9	66.3	10.7	1998.09	6872.41	4395.69	114.49	3705.63	
3	8.6	84.7	88.6	78.7	12.1	7174.09	7849.96	6193.69	146.41	7504.42	
4	13.1	96.7	79.3	76.2	12.5	9350.89	6288.49	5806.44	156.25	7668.31	

	MH	MS	WH	WS	HS
0	3619.88	597.74	6512.32	1075.36	1238.38
1	4509.06	775.10	4917.15	845.25	769.35
2	2963.61	478.29	5496.27	887.03	709.41
3	6665.89	1024.87	6972.82	1072.06	952.27
4	7368.54	1208.75	6042.66	991.25	952.50

[68]:
```
x = np.
  ↪array(df[['M','W','H','S','M2','W2','H2','S2','MW','MH','MS','WH','WS','HS']])
scaler = StandardScaler()
x = scaler.fit_transform(x)
```

[69]:
```
# Train classifier with all the available observations
clf = MLPRegressor(hidden_layer_sizes=(10, 10), max_iter=10000)
clf.fit(x, y)
# Predict one new sample
kf = KFold(n_splits=5, shuffle=True)
cv_mse = []
cv_mae = []
for train_index, test_index in kf.split(x):
    x_train, x_test = x[train_index], x[test_index]
    y_train, y_test = y[train_index], y[test_index]

    regressor_cv = MLPRegressor(hidden_layer_sizes=(10, 10), max_iter=10000)
    regressor_cv.fit(x_train, y_train)

    y_pred = regressor_cv.predict(x_test)
    mse = mean_squared_error(y_test, y_pred)
    cv_mse.append(mse)
    mae = mean_absolute_error(y_test, y_pred)
    cv_mae.append(mae)

# Calcular el error cuadrático medio promedio
mse = np.mean(cv_mse)
```

```
mae = np.mean(cv_mae)
print("MSE:", mse, "\nMAE:", mae)
```

```
MSE: 39.593297615630156
MAE: 3.553553720456468
```

0.1.3 3. Viendo los resultados de regresión, desarrolla una conclusión sobre los siguientes puntos:

¿Consideras que el modelo perceptrón multicapa es efectivo para modelar los datos del problema? ¿Por qué?

Considero que el modelo si es efectivo debido a que las medidas de error son bastante bajas

¿Qué modelo es mejor para los datos de criminalidad, el lineal o el perceptrón multicapa? ¿Por qué?

Considero que el modelo perceptrón es mejor porque aunque en el error medio absoluto sea similar (teniendo una mejora en el primer caso y empeora levemente en el segundo) el error medio cuadratico tiene una mejora significativa en ambos casos

0.2 Ejercicio 2

[72]:
```
df2 = np.loadtxt('/content/drive/MyDrive/Inteligencia Artificial/M_1.txt')
df2
```

[72]:
```
array([[ 1.        ,  1.        ,  0.6819565 , …,  1.69262835,
         1.34553809,  1.81638713],
       [ 1.        ,  1.        ,  0.56855303, …,  0.64268369,
         0.38791499,  1.59719973],
       [ 1.        ,  1.        ,  1.43149784, …,  0.53153428,
         1.10834576,  2.14520145],
       …,
       [ 7.        ,  1.        , -4.63831072, …, -1.9786276 ,
        -4.04741071, -5.17131175],
       [ 7.        ,  1.        , -5.2325368 , …, -1.31486405,
        -4.31667728, -4.56499901],
       [ 7.        ,  1.        , -4.95990009, …, -1.47060583,
        -4.8555384 , -5.13386256]])
```

[73]:
```
# Variables predictoras
x = df2[:,2:]
# Variable de respuesta
y = df2[:,0]
```

0.2.1 1. Evalúa un modelo perceptrón multicapa con validación cruzada utilizando al menos 5 capas de 20 neuronas.

[74]:
```
clf = MLPClassifier(hidden_layer_sizes=(20,20,20,20,20), max_iter=10000)
clf.fit(x,y)

y_pred = cross_val_predict(MLPClassifier(hidden_layer_sizes=(20,20,20,20,20),␣
 ↪max_iter=10000), x, y)
print(classification_report(y, y_pred))
```

```
              precision    recall  f1-score   support

         1.0       0.97      0.98      0.97        90
         2.0       0.93      0.96      0.95        90
         3.0       0.93      0.89      0.91        90
         4.0       1.00      1.00      1.00        90
         5.0       0.95      0.97      0.96        90
         6.0       0.91      0.89      0.90        90
         7.0       0.99      1.00      0.99        90

    accuracy                           0.95       630
   macro avg       0.95      0.95      0.95       630
weighted avg       0.95      0.95      0.95       630
```

0.2.2 2. Evalúa un modelo perceptrón multicapa con validación cruzada, pero encontrando el número óptimo de capas y neuronas de la red.

[75]:
```
# Optimal number of layers and neurons
num_layers = np.arange(1, 20, 5)
num_neurons = np.arange(10, 110, 20)

layers = []

for l in num_layers:
    for n in num_neurons:
        layers.append(l*[n])

clf = GridSearchCV(MLPClassifier(max_iter=10000), {'hidden_layer_sizes':␣
 ↪layers}, cv = 5)
clf.fit(x, y)
print(clf.best_estimator_)
```

```
MLPClassifier(hidden_layer_sizes=[30], max_iter=10000)
```

0.2.3 3. Prepara el modelo perceptrón multicapa:

Opten los hiperparámetros óptimos de capas y neuronas de la red.

Con los hiperparámetros óptimos, ajusta el modelo con todos los datos.

[76]:
```
clf = GridSearchCV(MLPClassifier(max_iter=10000), {'hidden_layer_sizes':␣
↪layers}, cv = 5)
y_pred = cross_val_predict(clf, x, y, cv = 5)
print(classification_report(y, y_pred))
```

```
              precision    recall  f1-score   support

         1.0       0.98      0.99      0.98        90
         2.0       0.95      0.97      0.96        90
         3.0       0.96      0.97      0.96        90
         4.0       0.99      0.99      0.99        90
         5.0       0.94      0.98      0.96        90
         6.0       0.96      0.91      0.94        90
         7.0       0.99      0.96      0.97        90

    accuracy                           0.97       630
   macro avg       0.97      0.97      0.97       630
weighted avg       0.97      0.97      0.97       630
```

0.2.4 4. Contesta lo siguientes:

¿Observas alguna mejora importante al optimizar el tamaño de la red? ¿Es el resultado que esperabas? Argumenta tu respuesta.

Comparandolo con los modelos de clasificación de la actividad anterior, no existe una mejora entre ambos modelos sin embargo sigue arrojando muy buenos resultados, siendo estos una precisión del 97% en general

¿Qué inconvenientes hay al encontrar el tamaño óptimo de la red? ¿Por qué?

Los inconvenientes de este proceso es que puede tener un costo computacional elevado, ademas de que el modelo se puede sobreajustar a los datos, por lo que puede llegar a fallar con otros datos que se le agreguen.