g-multiclase-alumnos-avanzadofinal

September 2, 2024

[3]:
```
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.preprocessing import StandardScaler
import numpy as np
import pandas as pd
import math
from sklearn.model_selection import train_test_split
```

[4]:
```
def sigmoid_function(X):
  return 1/(1+math.e**(-X))

def log_regression4(X, y, alpha, epochs):
  y_ = np.reshape(y, (len(y), 1)) # shape (150,1)
  N = len(X)
  theta = np.random.randn(len(X[0]) + 1, 1) #* initialize theta
  X_vect = np.c_[np.ones((len(X), 1)), X] #* Add x0 (column of 1s)
  avg_loss_list = []
  loss_last_epoch = 9999999
  for epoch in range(epochs):
    sigmoid_x_theta = sigmoid_function(X_vect.dot(theta)) # shape: (150,5).
↪(5,1) = (150,1)
    grad = (1/N) * X_vect.T.dot(sigmoid_x_theta - y_) # shapes: (5,150).(150,1)␣
↪= (5, 1)
    best_params = theta
    theta = theta - (alpha * grad)
    hyp = sigmoid_function(X_vect.dot(theta)) # shape (150,5).(5,1) = (150,1)
    avg_loss = -np.sum(np.dot(y_.T, np.log(hyp) + np.dot((1-y_).T, np.
↪log(1-hyp)))) / len(hyp)
    # if epoch % 50 == 0:
    #   print('epoch: {} | avg_loss: {}'.format(epoch, avg_loss))
    #   print('')
    avg_loss_list.append(avg_loss)
    loss_step = abs(loss_last_epoch - avg_loss) #*
    loss_last_epoch = avg_loss #*
    # if loss_step < 0.001: #*
```

```
    #   # print('\nStopping training on epoch {}/{}, as (last epoch loss -␣
 ↪current epoch loss) is less than 0.001 [{}]'.format(epoch, epochs,␣
 ↪loss_step)) #*
    #   break #*
  # plt.plot(np.arange(1, epoch+1), avg_loss_list[1:], color='red')
  # plt.title('Cost function')
  # plt.xlabel('Epochs')
  # plt.ylabel('Cost')
  # plt.show()
  return best_params
```

[5]:
```
# Load Dataset
iris = datasets.load_iris()
# Define X features
X = iris["data"]
# Define binary target 'y' based on iris plant type
y_seto = (iris["target"] == 0).astype(int) # return 1 if Iris Setosa (0 =␣
 ↪setosa, 1 = versicolor, 2 = virginica), and 0 if not
y_vers = (iris["target"] == 1).astype(int)
y_virg = (iris["target"] == 2).astype(int)
# List of ys
y_iris_types = [y_seto, y_vers, y_virg]
y_iris_types = {'Setosa':y_seto,
                'Versicolor':y_vers,
                'Virginica':y_virg}
predicted_probs = {'Setosa':0.0,
                   'Versicolor':0.0,
                   'Virginica':0.0}
actual_y = {'Setosa':0,
            'Versicolor':0,
            'Virginica':0}
for key, y_iris_type in y_iris_types.items():
  # Split dataset (training and testing sets)
  X_train, X_test, y_train, y_test = train_test_split(X, y_iris_type,␣
 ↪test_size=0.2, random_state=0)
  # Scale X
  sc = StandardScaler()
  X_train = sc.fit_transform(X_train)
  X_test = sc.transform(X_test)
  # Train model
  epochs = 1000
  alpha = 1
  best_params = log_regression4(X_train, y_train, alpha, epochs)
  # Make predictions on test set
  index_ = 10
  X_to_predict = [list(X_test[index_])]
  # print(X_to_predict)
```

```
  X_to_predict = np.c_[np.ones((len(X_to_predict), 1)), X_to_predict] # add x0␣
 ↪for bias
  # print(X_to_predict)
  pred_probability = sigmoid_function(X_to_predict.dot(best_params))
  predicted_probs[key] = pred_probability[0][0]
  print('Our model calculated probability of sample being {}, is: {}%'.
 ↪format(key, round(pred_probability[0][0]*100,2)))
  actual_y[key] = y_test[index_]

max_key = max(predicted_probs, key=predicted_probs.get)
print('\n', predicted_probs)
print('\nModel Prediction: {}'.format(max_key))
max_actual_y = max(actual_y, key=actual_y.get)
print('Real value is: {}'.format(max_actual_y))
```

```
Our model calculated probability of sample being Setosa, is: 0.0%
Our model calculated probability of sample being Versicolor, is: 90.64%
Our model calculated probability of sample being Virginica, is: 61.03%

 {'Setosa': 1.6970601192048972e-05, 'Versicolor': 0.9064223992873067,
'Virginica': 0.6103332368086836}

Model Prediction: Versicolor
Real value is: Virginica
```

Discusión: 1. ¿Qué hace este script exactamente?

Este script entrena un modelo de regresión logística para clasificar las flores del conjunto de datos iris en tres tipos diferentes: Setosa, Versicolor, y Virginica. Esto se logra de diferentes formas, empezando por las etiquetas para cada tipo de flor asi llevando al dividir los datos y entrenar el modelo para tener una clase y determinar el resultado.

2. ¿Qué significa que mi modelo generalice bien?

Un modelo que generaliza bien es cuando este no solo se ajusta correctamente a los datos de entrenamiento, sino que también predice correctamente en datos nuevos y no vistos, es decir, datos de prueba.

3. ¿Cómo puedo evaluar la generalización de mi modelo?

Usando el conjunto de datos de prueba que no se utilizó durante el entrenamiento.

4. ¿Qué tipo de metricos tendría que usar?

Accuracy, Precision, Recall y F1 Score.

5. ¿Qué tendría que agregarle al código para lograrlo?

Agregar predicciones para el conjunto de prueba y calcular las métricas usando sklearn.metrics

[ ]:
```
#Aprendizaje:
```

```
#Aprendí a implementar un modelo de regresión logística para clasificar tipos␣
 ↪de Iris, cómo evaluar su rendimiento
#utilizando métricas como accuracy, precision, recall y F1 score, y la␣
 ↪importancia de medir la capacidad de generalización
#del modelo en datos no vistos.
```