Regresión logística
multi-clase y redes neuronales

Código

Regresión logística multi-clase

```
from scipy.io import loadmat
import numpy as np
import matplotlib.pyplot as plt
import scipy.optimize as opt

data = loadmat('p3/ex3data1.mat')
y = data['y']
X = data['X']

X_new = np.ones((5000, 401))
X_new[:, 1:] = X
size = len(y)
lambda_reg = 0.1

sample = np.random.choice(X.shape[0], 10)
plt.imshow(X[sample, :].reshape(-1, 20).T)
plt.axis('off')
plt.show()

def sigmoid(x):
    return 1.0 / (1.0 + np.exp(-x))

def cost_function(theta, x, y, lambda_reg):
    m = x.shape[0]
    J = (-np.log(sigmoid(x.dot(theta)).T).dot(y) - np.log(1 -
sigmoid(x.dot(theta)).T).dot(1 - y))/m + lambda_reg*np.sum(np.square(theta))/(2*m)
    return J

def gradient_function(theta, x, y, lambda_reg):
    m = x.shape[0]
    h = sigmoid(x.dot(theta)).reshape(-1, 1)
    y = y.reshape(m, 1)
    gradient = np.zeros((theta.shape[0], 1))
    gradient = x.T.dot(h - y)/m
    theta = theta.reshape((theta.shape[0], 1))
    gradient[1:] = gradient[1:] + (lambda_reg/m)*theta[1:]
    return gradient

def oneVsAll(X, y, num_labels, lambda_reg):
    thetas_set = np.zeros((num_labels, 401))
    for i in range(num_labels):
        theta = np.zeros((401, 1))
        example = (y == i + 1) * 1
        result = opt.fmin_tnc(func=cost_function , x0=theta ,
fprime=gradient_function , args=(X, example, lambda_reg))
        theta_opt = result[0]
        idx = i + 1
        if(i == 9):
            idx = 0
```

```
        thetas_set[idx,:] = theta_opt
    return thetas_set

thetas = oneVsAll(X_new, y, 10, lambda_reg)

activations = X_new.dot(thetas.T)
predictions = np.zeros(len(activations))
predictions = predictions.reshape(X.shape[0], 1)

for i in range(len(activations)):
    idx = np.argmax(activations[i])
    if(idx == 0):
        predictions[i] = 10
    else:
        predictions[i] = idx

number = np.sum(predictions == y)
accuracy = (float(number)/X.shape[0])*100
print("Accuracy = " + str(accuracy) + "%")
```

Redes neuronales

```
from scipy.io import loadmat
import numpy as np

data = loadmat('p3/ex3data1.mat')
y = data['y']
X = data['X']

weights = loadmat('p3/ex3weights.mat')
theta1, theta2 = weights['Theta1'], weights['Theta2']

X_new = np.ones((5000, 401))
X_new[:, 1:] = X
size = len(y)

def sigmoid(x):
    return 1.0 / (1.0 + np.exp(-x))

def neuralNetwork(X, theta1, theta2):
    a2 = sigmoid(X.dot(theta1.T))
    a2_new = np.ones((a2.shape[0], a2.shape[1]+1))
    a2_new[:, 1:] = a2
    a3 = sigmoid(a2_new.dot(theta2.T))
    predictions = np.zeros(len(a3))
    predictions = predictions.reshape(X.shape[0], 1)
    for i in range(len(a3)):
        idx = np.argmax(a3[i])
        idx = (idx + 1) % 10
        if(idx == 0):
            predictions[i] = 10
        else:
            predictions[i] = idx
    return predictions

predictions = neuralNetwork(X_new, theta1, theta2)

number = np.sum(predictions == y)
accuracy = (float(number)/X.shape[0])*100
```

```
print("Accuracy = " + str(accuracy) + "%")
```

Resultados

Regresión logística multi-clase

```
Accuracy = 96.46%
```

Redes neuronales

```
Accuracy = 97.52%
```

Comentarios

Para este conjunto de datos la red neuronal tiene el valor de accuracy más grande que la regresión logística multi-clase (aproximadamente 1% más grande).