COS 598 – Machine Learning - Homework #5

Due: 5:00PM 04/23/2021

Homework Submission: Homeworks must be submitted via Brightspace as pdf files. This includes your code when appropriate. Please use a high quality scanner if possible, as found at the library or your departmental copy room. If you must use your phone, please don't just take photos (if possible), at least use an app like CamScanner that provides some correction for shading and projective transformations.

1) **Neural Network (20 pts).**

Consider that you have a three layer neural network as shown in the figure 1. let x be the input, W_1, W_2, W_3 be the weights of the these layers, b_1, b_2, b_3 be the corresponding biases and y be the output.

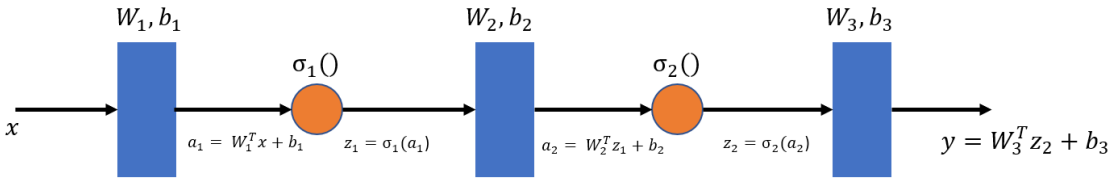

Figure 1: Neural Network

Load the body fat data:

```
#Load the data
bodyfat = sio.loadmat('bodyfat_data.mat')
X = bodyfat['X']
Y = bodyfat['y']
n,d = X.shape
```

Let's try to fit the data using neural network. Use the first 150 examples for training, and the remainder for estimating the mean squared error. Assume that all activation functions are rectified linear unit (ReLU).

a. Implement the forward pass and the backward pass. Please submit your code.

b. Report the mean squared error on the training data

c. Report the mean squared error on the test inputs

Additional comments:

- Initialize weights of all layers using normal distribution (Please add rng(0) or np.random.seed(0))
- Initialize biases with zeros.
- Size of the first layer is 64 and the second layer is 16.

- Try to vectorize the code to reduce running time.
- Use gradient descent for training and use reasonable stopping criteria to terminate the gradient updates.
- Use the learning rate $1e-7$ for gradient descent.

Hint: Forward pass and backward pass functions are attached below.

```
#forward pass function that returns all activations in middle layers:

def forward_pass(X,W1,W2,W3,b1,b2,b3):
    z1=np.dot(X,W1)+b1
    a1=relu(z1)
    z2=np.dot(a1,W2)+b2
    a2=relu(z2)
    y_hat =np.dot(a2,W3)+b3

    return z1,z2,y_hat,a1,a2


#backward pass function that returns all weights and biases:

def backward_pass(X,ytr,y_hat,W1,W2,W3,b1,b2,b3,z1,z2,a1,a2):
    #loss function
    loss=np.mean((y_hat-ytr)**2)
    print(loss)

    #backpropogation code
    l3_error=(y_hat-ytr)/float(ytr.shape[0])
    l2_error=l3_error.dot(W3.T)
    l2_delta=l2_error*relu(z2,derivative=True)
    l1_error=l2_delta.dot(W2.T)
    l1_delta=l1_error*relu(z1,derivative=True)

    #update parameters using GD
    W3-= learning_rate*a2.T.dot(l3_error)
    W2-= learning_rate*a1.T.dot(l2_delta)
    W1-= learning_rate*Xtr.T.dot(l1_delta)
    b3-= learning_rate*np.sum(l3_error)
    b2-= learning_rate*np.sum(l2_delta,axis = 0)
    b1-= learning_rate*np.sum(l1_delta,axis = 0)

    return W1,W2,W3,b1,b2,b3
```