LIGN 167: Problem Set 3

Name: Chih-Hsuan Kao

Date: November 6, 2018

In [ ]:

```
# coding: utf-8

import numpy as np
import torch
```

In [ ]:

```
def relu(x):
    if x<0:
        return 0
    else:
        return x

def loss(y_predicted, y_observed):
    return (y_predicted - y_observed)**2


def mlp(x,W0,W1,W2):


    r0_0 = x*W0[0]
    r0_1 = x*W0[1]
    r0_2 = x*W0[2]
    r0 = np.array([r0_0,r0_1,r0_2])

    h0_0 = relu(r0_0)
    h0_1 = relu(r0_1)
    h0_2 = relu(r0_2)
    h0 = np.array([h0_0,h0_1,h0_2])



    r1_0 = h0_0*W1[0,0] + h0_1*W1[0,1]+ h0_2*W1[0,2]
    r1_1 = h0_0*W1[1,0] + h0_1*W1[1,1]+ h0_2*W1[1,2]
    r1_2 = h0_0*W1[2,0] + h0_1*W1[2,1]+ h0_2*W1[2,2]
    r1 = np.array([r1_0,r1_1,r1_2])

    h1_0 = relu(r1_0)
    h1_1 = relu(r1_1)
    h1_2 = relu(r1_2)
    h1 = np.array([h1_0,h1_1,h1_2])

    y_predicted = h1_0*W2[0] + h1_1*W2[1]+ h1_2*W2[2]

    variable_dict = {}
    variable_dict['x'] = x
    variable_dict['r0'] = r0
    variable_dict['h0'] = h0
    variable_dict['r1'] = r1
    variable_dict['h1'] = h1
    variable_dict['y_predicted'] = y_predicted

    return variable_dict


x = 10
W0 = np.array([1,2,3])
W1 = np.array([[3,4,5],[-5,4,3],[3,4,1]])
W2 = np.array([1,3,-3])
b = mlp(x,W0,W1,W2)
print(mlp(x,W0,W1,W2))
```

```
### Pytorch Section

def torch_mlp(x,W0,W1,W2):
    m = torch.nn.ReLU()
    h0 = m(torch.mul(W0,x))

    h1 = m(torch.matmul(W1,h0))

    y_predicted = torch.dot(W2,h1)

    return y_predicted

def torch_loss(y_predicted,y_observed):
    return torch.pow(y_predicted-y_observed,2)


x_torch = torch.tensor(x,dtype=torch.float)
W0_torch = torch.tensor(W0,dtype=torch.float,requires_grad=True)
W1_torch = torch.tensor(W1,dtype=torch.float,requires_grad=True)
W2_torch = torch.tensor(W2,dtype=torch.float,requires_grad=True)
output = torch_mlp(x_torch,W0_torch,W1_torch,W2_torch)


#y_observed = 180;
#loss = loss(output, y_observed)
#print(loss)
#loss.backward()
#print('W0:', W0_torch)
#print('W1:', W1_torch)
#print('W2:', W2_torch)
#print('W0.grad:', W0_torch.grad)
#print('W1.grad:', W1_torch.grad)
#print('W2.grad:', W2_torch.grad)

#variable_dict = mlp(x,W0,W1,W2)
#W0_grad = d_loss_d_W0(variable_dict,W1,W2,y_observed)
#W1_grad = d_loss_d_W1(variable_dict,W2,y_observed)
#W2_grad = d_loss_d_W2(variable_dict,y_observed)
#print('W0_grad:', W0_grad)
#print('W1_grad:', W1_grad)
#print('W2_grad:', W2_grad)
```

In [ ]:

```
#PROBLEM 1
def d_loss_d_ypredicted(variable_dict,y_observed):
    return 2*(variable_dict['y_predicted'] - y_observed)
```

In [ ]:

```
#PROBLEM 2
def d_loss_d_W2(variable_dict,y_observed):
    return d_loss_d_ypredicted(variable_dict,y_observed) * variable_dict['h1']
```

In [ ]:

```
#PROBLEM 3
def d_loss_d_h1(variable_dict,W2,y_observed):
    return d_loss_d_ypredicted(variable_dict,y_observed) * W2
```

In [ ]:

```
#PROBLEM 4
def relu_derivative(x):
    if x<0:
        return 0
    else:
        return 1
```

In [ ]:

```
#PROBLEM 5
```

```
def d_loss_d_r1(variable_dict,W2,y_observed):
    dh = np.ones(3)
    for i in range(3):
        dh[i] = relu_derivative(variable_dict['h1'][i])
    return d_loss_d_h1(variable_dict,W2,y_observed) * dh
#print(d_loss_d_r1(variable_dict,W2,y_observed))
```

In [ ]:

```
#PROBLEM 6
def d_loss_d_W1(variable_dict,W2,y_observed):
    return np.outer(d_loss_d_r1(variable_dict,W2,y_observed), variable_dict['h0'])
#print(d_loss_d_W1(b,W2,y_observed))
```

In [ ]:

```
#PROBLEM 7
def d_loss_d_h0(variable_dict,W1,W2,y_observed):
    dh = np.zeros(3);
    dr1 = d_loss_d_r1(variable_dict,W2,y_observed);
    dr1 = np.reshape(dr1,(1,3));
    w0 = np.reshape(W1[:,0],(3,1))
    w1 = np.reshape(W1[:,1],(3,1))
    w2 = np.reshape(W1[:,2],(3,1))
    dh[0] = np.dot(dr1,w0)
    dh[1] = np.dot(dr1,w1)
    dh[2] = np.dot(dr1,w2)
    return dh
#print(d_loss_d_h0(b,W1,W2,y_observed))
```

In [ ]:

```
#PROBLEM 8
def d_loss_d_r0(variable_dict,W1,W2,y_observed):
    dh0 = d_loss_d_h0(variable_dict,W1,W2,y_observed)
    r0 = variable_dict['r0']
    dr0 = np.zeros(3);
    dr0[0] = dh0[0]*relu_derivative(r0[0]);
    dr0[1] = dh0[1]*relu_derivative(r0[1]);
    dr0[2] = dh0[2]*relu_derivative(r0[2]);
    return dr0
#print(d_loss_d_r0(variable_dict,W1,W2,y_observed))
```

In [ ]:

```
#PROBLEM 9
def d_loss_d_W0(variable_dict,W1,W2,y_observed):
    return d_loss_d_r0(variable_dict,W1,W2,y_observed) * variable_dict['x']
```