ECE 544- Homework 3

Ahmadreza Eslaminia (Ae15)

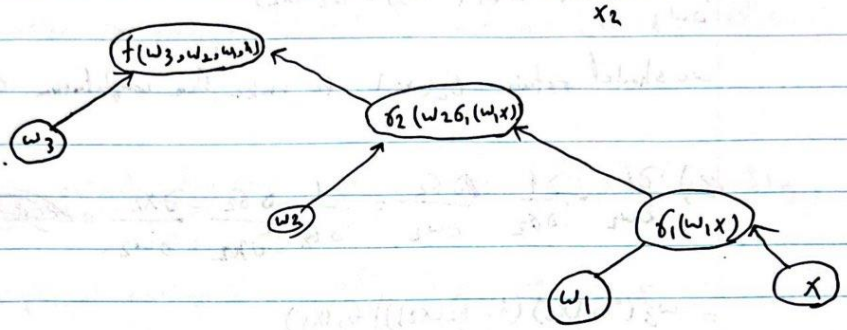
Questions:

Ahmadreza Eslaminia ECE 544. HW 3

AE15

1. [Deep Net] $f(w_3, w_2, w_1, x) = w_3 \sigma_2(w_2 \sigma_1(\underbrace{w_1 x}_{x_1}))$ $\sigma \rightarrow$ sigmoid (x_2)

a)



b) $\frac{\partial \sigma_1(u)}{\partial u} \quad \frac{0-(-\exp(-u))}{(1+\exp(-u))^2} = \frac{\exp(-u)}{(1+\exp(-u))^2} \Rightarrow \frac{\partial \sigma_1(u)}{\partial u} = \frac{\exp(-u)}{(1+\exp(-u))^2}$ (I)

we have $\Rightarrow \frac{1}{\sigma_1(u)} = 1+\exp(-u) \Rightarrow \exp(-u) = \frac{1-\sigma_1(u)}{\sigma_1(u)}$

(I) $\Rightarrow \frac{\partial \sigma_1(u)}{\partial u} = \frac{1-\sigma_1(u)}{\sigma_1(u)} \times \sigma_1(u)^2 = \sigma_1(u)(1-\sigma_1(u)) \Rightarrow \frac{\partial \sigma_1(u)}{\partial u} = \sigma_1(u)(1-\sigma_1(u))$

c) The forward pass refers to calculation process in order to get values of output layers from input data. it's traversing input data through all layers from first to last. / backward pass is counting changes in the weights using gradient decent algorithm which is computed from last layer (loss) backward to the first one.

Scanned with CamScanner

1. d) $f(w_3, w_2, w_1, x) = w_3 \sigma_2(w_2 \sigma_1(w_1 x))$

$$\frac{\partial f}{\partial w_3} = \sigma_2(w_2 \sigma_1(w_1 x)) = \sigma_2(x_2)$$

we should retain $\sigma_2(x_2)$ to make the computation easy.

e) $$\frac{\partial f}{\partial w_2} = \frac{\partial f}{\partial \sigma_2} \frac{\partial \sigma_2}{\partial w_2} = \frac{\partial f}{\partial \sigma_2} \frac{\partial \sigma_2}{\partial x_2} \frac{\partial x_2}{\partial w_2} = [illegible]$$

$$= w_3 (\sigma_2(x_2)(1 - \sigma_2(x_2))) \sigma_1(x_1)$$

$$\Rightarrow \frac{\partial f}{\partial w_2} = w_3 \times \sigma_2'(w_2 \sigma_1(w_1 x)) \times [1 - \sigma_2(w_2 \sigma_1(w_1 x))] \times \sigma_1(w_1 x)$$

For this calculation we should retain $\sigma_1(x_1)$, $\sigma_2(x_2)$ and w_3.

f) $$\frac{\partial f}{\partial w_1} = \frac{\partial f}{\partial x_2} \frac{\partial \sigma_2}{\partial \sigma_1} \frac{\partial \sigma_1}{\partial w_1} = \frac{\partial f}{\partial \sigma_2} \frac{\partial \sigma_2}{\partial x_2} \frac{\partial x_2}{\partial \sigma_1} \frac{\partial \sigma_1}{\partial x_1} \frac{\partial x_1}{\partial w_1}$$

$$\Rightarrow \frac{\partial f}{\partial w_1} = w_3 \sigma_2(x_2)(1 - \sigma_2(x_2)) w_3 \sigma_1(x_1)(1 - \sigma_1(x_1)) x$$

As you can see, we should retain $x, w_3, \sigma_1(x_1), \sigma_2(x_2), w_3$ to make computation easy.

We should first compute $\frac{\partial f}{\partial w_3}$ then $\frac{\partial f}{\partial w_2}$ then $\frac{\partial f}{\partial w_1}$ to make it as easy as possible

this order is exactly reverse of the Forward pass order.

Scanned with CamScanner

P2

g) output after convolution: $\frac{28 + 2\times0 - 5}{1} + 1 = 24 \rightarrow (24\times24)$ (20 channel)

(padding, filter, 1 → stride)

output after max pooling: $\frac{24 - 2}{2} + 1 = 12 \rightarrow (12\times12)$ (20 channel)

(padding max pool, stride 2)

h) $\Rightarrow \frac{\text{output after conv} - 2}{2} + 1 = 4$ (output of max pooling)

$\Rightarrow$ output of conv → 8 $\Rightarrow \frac{\text{input } 12 - \text{filter}}{\text{stid}} + 1 = 8$

$\Rightarrow$ filter size (5×5) → stride → 1 $\Rightarrow$ we have 50 filter with 5×5 dim and stide 1 and padding

i)

```
class Net(nn.Module):
    def __init__(self):
        super(Net,self).__init__()
        self.conv1 = nn.Conv2d(1,20,5)
        self.conv2 = nn.Conv2d(20,20,5)
        self.fc1 = nn.Linear(50x4x4, 500)
        self.fc2 = nn.Linear(500, 10)
    def forward(self,x):
        x = F.relu(self.conv1(x))
        x = F.max_pool2d(x,(2,2))
        x = F.relu(self.conv2(x))
        x = F.max_pool2d(x,(2,2))
        x = x.view(-1, 50x4x4))
        x = F.relu(self.fc1(x))
        return self.fc2(x)
```

best accuracy: 99.26%

network parameters:

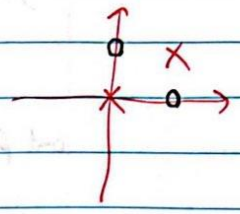
431080

CS Scanned with CamScanner

P3 P5

2. [Presentation of DNN]

i	$x_0^{(i)}$	$x_1^{(i)}$	y
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	0



a) $p(y^{(i)}|x) = \frac{1}{1+\exp(-w^T x^{(i)} - b)} \cdot \frac{1}{1+\exp(-w_0 x_0 - w_1 x_1 - b)}$

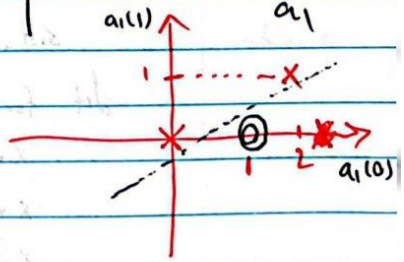
$(x,y) \rightarrow \begin{cases} (0,0) \Rightarrow p = \frac{1}{1+\exp(-b)} \xrightarrow{\text{assume}} p < 0.5 \Rightarrow b < 0 \\ (0,1) \Rightarrow p = \frac{1}{1+\exp(-w_1-b)} > 0.5 \Rightarrow -w_1 - b < 0 \Rightarrow w_1 + b > 0 \Rightarrow w_1 > 0 \\ (1,0) \rightarrow w_0 + b > 0 \Rightarrow w_0 > 0 \\ (1,1) \rightarrow w_1 + w_0 + b > 0 \end{cases}$

with first 3 corrected
$\Rightarrow$ the last one $w_1 + (w_0 + b) > 0$ not correct since $(w_1 > 0, w_0 + b > 0)$
$\Rightarrow$ at most $\frac{3}{4} = 75\%$

b) we compute a_1 as follow

i	y	$a_1^{(i)}(0)$	$a_1^{(i)}(1)$	$\max(a_1, 0)$	$\max_i(a_1^{(i)}(0), 0)$	$\max_i(a_1^{(i)}(1))$
0	0	0	-1		0	0
1	1	1	0		1	0
2	1	1	0		1	0
3	0	2	1		2	1

$\Rightarrow a_1$



$c = [0, -1]^T \quad w = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$

we want $a_2 < 0$, [redacted] the we can set

$\theta = [1, -3]^T \quad b = -0.5 \Rightarrow$

i	$a_2^{(i)}$
0	-0.5
1	0.5
2	0.5
3	-1.5

Scanned with CamScanner

P_4

2.c) $\min L_1 = \sum \| y^{(i)} - \tilde{y}^{(i)} \|_2^2$

$\Rightarrow \frac{\partial L}{\partial \tilde{y}^i} = (-1) \times 2 [y^{(i)} - \tilde{y}^{(i)}] = -2 [y^{(i)} - \tilde{y}^{(i)}]$

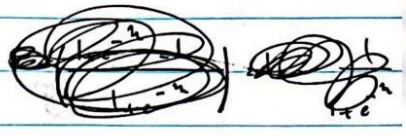
d) $\frac{\partial \tilde{y}^i}{\partial a_2^{(i)}}$? $\tilde{y}^i = \frac{1}{1 + \exp^{(-a_2^{(i)})}} \Rightarrow \frac{\partial \tilde{y}^i}{\partial a_2^i} = \frac{\exp^{(-a_2^i)}}{(1 + \exp^{(a_2^i)})^2}$

e) $\frac{\partial L}{\partial c} = \sum_{(x,y) \in D} \frac{\partial L}{\partial \tilde{y}^i} \frac{\partial \tilde{y}^i}{\partial a_2^{(i)}} \frac{\partial a_2^i}{\partial a_1^{(i)}}$

$= \sum_{(x,y) \in D} \frac{\partial L}{\partial \tilde{y}^i} \cdot \frac{\partial \tilde{y}^i}{\partial a_{i2}^{(i)}} \left(\theta \cdot \mathbb{1}[a_1^i > 0] \right)$

Scanned with CamScanner

3. [Back propation] $z_j = c_j + \sum_i a_i u_{ij}$

a) $g(x) = \sigma(x) = \frac{1}{1+e^{-x}} \Rightarrow g'(x) = \frac{e^{-x}}{(1+e^{-x})^2} =$ 

$= \left(\frac{1}{1+e^{-x}}\right)\left(\frac{e^{-x}+1-1}{1+e^{-x}}\right) = \sigma(x)(1-\sigma(x))$ ✓

b) $E = \frac{1}{2}\sum_k (c_k - t_k)^2 \quad c_k = g(f_k + \sum_j w_{jk} b_j)$

$\Rightarrow \frac{\partial E}{\partial w_{jk}} = \frac{\partial E}{\partial c_k}\frac{\partial c_k}{\partial w_{jk}} = (c_k - t_k) g'(f_k + \sum_j w_{jk} b_j) b_j$

c) $\frac{\partial E}{\partial f_k} = \frac{\partial E}{\partial c_k}\frac{\partial c_k}{\partial f_k} = (c_k - t_k) g'(f_k + \sum_j w_{jk} b_j)$

d) $\frac{\partial E}{\partial u_{ij}} = \frac{\partial E}{\partial c_k}\frac{\partial c_k}{\partial b_j}\frac{\partial b_j}{\partial u_{ij}} = \sum_k (c_k - t_k) g'(f_k + \sum_j w_{jk} b_j) w_{jk} \times g'(z_j) a_i$

$\Rightarrow \frac{\partial E}{\partial u_{ij}} = \sum_k (c_k - t_k) g'(f_k + \sum_j w_{jk} b_j) w_{jk} \times g'(z_j) a_i$

e) $\frac{\partial E}{\partial c_j} = \frac{\partial E}{\partial c_k}\frac{\partial c_k}{\partial b_j}\frac{\partial b_j}{\partial e_j} = \sum_k (c_k - t_k) g'(f_k + \sum_j w_{jk} b_j) w_{jk} g'(z_j)$

Scanned with CamScanner

4. [Struc. Predic.] $p(y) = \frac{1}{Z}\exp(F(y))$

a) since $\sum p = 0 \Rightarrow$ we need Z as ("normalization constant") to make sure $\sum p = 0$ stands true. we have $(F(0,0), F(0,1), F(1,0), F(1,1))$

$$\sum p = 0 \Rightarrow \sum_{(y_1,y_2)\in Y_i} \frac{1}{Z}\exp^{F(y_1,y_2)} = 0 \Rightarrow \boxed{Z = \sum_{(y_1,y_2)\in Y_i} \exp^{F(y_1,y_2)}}$$

b) solve $\max_{\hat{p}\in\Delta_Y} \sum_{y\in Y} \hat{p}(y)F(y) - \sum_{y\in Y} \varepsilon\,\hat{p}(y)\log\hat{p}(y)$

$$L(\hat{p}) = \sum_{y\in Y} \hat{p}(y)F(y) + \sum_{y\in Y} \varepsilon\,\hat{p}(y)\log\hat{p}(y) + \lambda\left(\sum \hat{p}(y) - 1\right)$$

$$\frac{\partial L}{\partial \hat{p}(y)} = -F(y) + \varepsilon\log\hat{p}(y) + \varepsilon + \lambda = 0 \Rightarrow \hat{p}(y) = \exp\left(\frac{1}{\varepsilon}(F(y)-\lambda)-1\right)$$

$$g(\lambda) = \sum_{y\in Y} \exp\left(\frac{1}{\varepsilon}(F(y)-\lambda)-1\right)\left(-F(y) + \varepsilon\left(\frac{1}{\varepsilon}(F(y)-\lambda)-1\right) + \lambda\right) - \lambda$$

$$= \sum \exp\left(\frac{1}{\varepsilon}(F(y)-\lambda)-1\right)\left(-F(y) + F(y) - \lambda - \varepsilon\right) - \lambda$$

$$= \sum \exp\left(\frac{1}{\varepsilon}(F(y)-\lambda)-1\right)(-\varepsilon) - \lambda$$

$$\Rightarrow \frac{\partial g(\lambda)}{\partial \lambda} = 0 \Rightarrow \exp\left(\frac{1}{\varepsilon}(F(y)-\lambda)-1\right)\cdot\frac{-1}{\varepsilon}\cdot(-\varepsilon) - 1 = 0 \Rightarrow \exp\left(\frac{1}{\varepsilon}(F(y)-\lambda)-1\right) = 1$$

$$\Rightarrow \frac{1}{\varepsilon}(F(y)-\lambda)-1 = 0 \Rightarrow \lambda = F(y)-\varepsilon$$

$$\Rightarrow \hat{p}^*(y) = \exp\left(\frac{1}{\varepsilon}(F(y)-F(y)+\varepsilon)-1\right) = \exp(0) = 1$$

$$\Rightarrow \text{cost function} = \sum 1\times F(y) - \sum \varepsilon\times 1\times\log 1 = \sum F(y)$$

Scanned with CamScanner

P2

c) $f(y) = f_1(y_1) + f_2(y_2) + f_{1,2}(y_1, y_2)$ $\quad y^* = \max F(y)$

$f_1 \rightarrow f_1(0), f_1(1) \rightarrow$ 2 values

$f_2 \rightarrow f_2(0), f_2(1) \rightarrow$ 2 values

$f_{1,2} \rightarrow F(0,0), F(0,1), F(1,0), F(1,1) \rightarrow$ 4 values

d) $\max_{r, y_r} \sum b_r(y_r) f_r(y_r)$ s.t $\begin{cases} b_r(y_r) \in \{0,1\} \\ \sum_{y_r} b_r(y_r) = 1 \\ \sum_{p \in P_r} b_p y_p = b_r(y_r) \end{cases}$

$\Rightarrow$ linear program $\rightarrow \max \big(b_1(y_1=0) f_1(y_1=0) + b_1(y_1=1) f_1(y_1=1)$
$+ b_2(y_2=0) f_2(y_1=0) + b_2(y_2=1) f_2(y_2=2)$
$+ b_{12}(y_1, y_2=0,0) f_2(y_1, y_2=0,0) + b_{12}(y_1, y_2=0,1) f_{12}(y_1, y_2=0,1)$
$+ b_{12}(y_1, y_2=1,0) f_{12}(y_1, y_2=10) + b_{12}(y_1, y_2=1,1) f_{12}(y_1, y_2=11) \big)$

s.t $\begin{cases} b_1(y_1), b_2(y_2), b_{12}(y_{12}) \in [0,1] \\ b_1(y_1=0) + b_1(y_1=1) = 1 \\ b_2(y_2=0) + b_2(y_2=1) = 1 \\ b_{12}(y_1, y_2=0,0) + b_{12}(y_1, y_2=0,1) + b_{12}(y_1, y_2=1,0) + b_{12}(y_1, y_2=1) = 1 \\ b_1(y_1) = b_{12}(y_1, y_2=0,0) + b_{12}(y_1, y_2=1,1) \\ b_2(y_2) = b_{12}(y_1=0, y_2) + b_{12}(y_1=1, y_2) \end{cases}$

Scanned with CamScanner

4) c → matrix form.

we can write $A_{8\times 8}$ =
$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & -1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & -1 \\ 0 & 0 & 1 & 0 & -1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & -1 & -1 \end{bmatrix} \quad c_2 \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

f) we input 1 since its a section of the optimistic problem.

$b = [0, 1, 1, 0, 0, 1, 0, 0]^T$

cost Function = $[1, 0, 0, 1, 1, 0, 1, 1]$

$x = [0, 1, 1, 0, 0, 1, 0, 0]$

Scanned with CamScanner

A5-Deepnet:

```
        return (self.d[idx,:,:], self.l[idx])

class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        ##############################
        ## declare the layers of the network which have parameters
        ##############################
        self.conv1 = nn.Conv2d(1,20,5)
        self.conv2 = nn.Conv2d(20, 50, 5)
        self.fc1 = nn.Linear(50*4*4,500)
        self.fc2 = nn.Linear(500,10)

    def forward(self, x):
        ##############################
        ## combine the layers; don't forget the relu and pooling operations
        ##############################
        x = F.relu(self.conv1(x))
        x = F.max_pool2d(x, (2, 2))
        x = F.relu(self.conv2(x))
        x = F.max_pool2d(x, (2, 2))
        x = x.view(-1,50*4*4)
        x = F.relu(self.fc1(x))
        return self.fc2(x)

testData = OurDataset('MNIST/t10k-images-idx3-ubyte','MNIST/t10k-labels-idx1-ubyte',transform=transforms.Compose([
                        transforms.Normalize((255*0.1307,), (255*0.3081,))
                    ]))
trainData = OurDataset('MNIST/train-images-idx3-ubyte','MNIST/train-labels-idx1-ubyte',transform=transforms.Compose([
                        transforms.Normalize((255*0.1307,), (255*0.3081,))
                    ]))
print(testData.__len__())
print(trainData.__len__())
```

```
PS C:\DriveA\UIUCcourses\Fall 2
& 'C:\Users\Ahmadreza\anaconda3
r' '58649' '--' 'C:\DriveA\UIUC
10000
60000
torch.Size([20, 1, 5, 5])
torch.Size([20])
Test Accuracy: 98.970000
Test Accuracy: 98.910000
Test Accuracy: 99.020000
Test Accuracy: 99.100000
Test Accuracy: 99.050000
Test Accuracy: 99.260000
Test Accuracy: 98.960000
Test Accuracy: 99.040000
PS C:\DriveA\UIUCcourses\Fall 2
```

A6-Structure:

Run Terminal Help A6_Structure.py - Visual Studio Code

A6_Structure.py × Workspace Trust

C: > DriveA > UIUCcourses > Fall 2023 > ECE 544 pattern recognition > HWS > h3 > homework3 > A6_Structure.py > ...

```python
import numpy as np
from scipy.optimize import linprog

#potentials
f = np.array([1., 0., 0., 1., 0., 5., 0., 0.])

#local probability constraints
##############################
## specify the constraints which have a value of one as their right hand side
## Dimensions: A_eq1 (list of three lists each with 8 entries)
##############################
A_eq1 = [[1,1 ,0 ,0,0 ,0 ,0 ,0 ],[0, 0, 1,1 ,0 ,0 ,0 ,0 ],[0,0 ,0 ,0 ,1 ,1 ,1 , 1]]
b_eq1 = [[1],]*3

#marginalization constraints
##############################
## specify the constraints which have a value of zero as their right hand side
## Dimensions: A_eq2 (list of three lists each with 8 entries)
##############################
A_eq2 = [[1,0 ,0 ,0 ,-1 ,0 ,-1 ,0 ],[0, 1, 0, 0,0 ,-1 ,0 ,-1 ],[0,0 ,1 ,0 , -1, -1, 0, 0],[0, 0, 0, 0,0 ,0 ,-1 ,-1 ]]
b_eq2 = [[0],]*4

#bounds
bounds = [(0, 1),]*8

res = linprog(-f, A_eq=np.concatenate((A_eq1,A_eq2)), b_eq=np.concatenate((b_eq1,b_eq2)), bounds=bounds)
print(res)

print(np.matmul(A_eq1, res.x))
print(np.matmul(A_eq2, res.x))
```

Result:

```
PS C:\DriveA\UIUCcourses\Fall 2023\ECE 544 pattern recognition\HWS\h3\homework3> c:; cd 'c
& 'C:\Users\Ahmadreza\anaconda3\python.exe' 'c:\Users\Ahmadreza\.vscode\extensions\ms-pytho
r' '63262' '--' 'C:\DriveA\UIUCcourses\Fall 2023\ECE 544 pattern recognition\HWS\h3\homewor
       message: Optimization terminated successfully. (HiGHS Status 7: Optimal)
       success: True
        status: 0
           fun: -5.0
             x: [-0.000e+00  1.000e+00  1.000e+00 -0.000e+00  0.000e+00
                  1.000e+00 -0.000e+00  0.000e+00]
           nit: 0
         lower:  residual: [-0.000e+00  1.000e+00  1.000e+00 -0.000e+00
                            0.000e+00  1.000e+00 -0.000e+00  0.000e+00]
                marginals: [ 0.000e+00  0.000e+00  0.000e+00  0.000e+00
                             4.000e+00  0.000e+00  0.000e+00  1.000e+00]
         upper:  residual: [ 1.000e+00  0.000e+00  0.000e+00  1.000e+00
                             1.000e+00  0.000e+00  1.000e+00  1.000e+00]
                marginals: [ 0.000e+00  0.000e+00  0.000e+00  0.000e+00
                             0.000e+00  0.000e+00  0.000e+00  0.000e+00]
         eqlin:  residual: [ 0.000e+00  0.000e+00  0.000e+00  0.000e+00
                             0.000e+00  0.000e+00  0.000e+00]
                marginals: [-4.000e+00 -1.000e+00 -0.000e+00  3.000e+00
                             4.000e+00  1.000e+00 -3.000e+00]
       ineqlin:  residual: []
                marginals: []
 mip_node_count: 0
 mip_dual_bound: 0.0
        mip_gap: 0.0
[1. 1. 1.]
[0. 0. 0. 0.]
PS C:\DriveA\UIUCcourses\Fall 2023\ECE 544 pattern recognition\HWS\h3\homework3>
```