Hwk 5 CSE 250A Andrew Ghafari A59020215.

Problem 1:

$P(Y=1 | x_1, x_2, x \ldots, x_d) = f(\sum_{i=1}^{d} w_i x_i) = f(\vec{w} \cdot \vec{x})$

a) $\frac{\partial L}{\partial w_i} = \frac{\partial}{\partial w_i}\left(\sum_t \log P(y_t | \vec{x_t})\right)$ $\qquad P(y_t|\vec{x_t}) = \begin{cases} p_t & \text{if } y_t = 1 \\ 1-p_t & \text{if } y_t = 0 \end{cases}$

$= \frac{\partial}{\partial w_i} \sum_{t=1}^{T} \log\left[(p_t^{y_t})(1-p_t)^{1-y_t}\right]$ $\Rightarrow p_t^{y_t} \cdot (1-p_t)^{(1-y_t)}$

chain rule

$= \sum_{t=1}^{T} \frac{\partial (y_t \log p_t)}{\partial p_t} \cdot \frac{\partial p_t}{\partial w_i} + \frac{\partial}{\partial p_t}(1-y_t)\log(1-p_t) \cdot \frac{\partial p_t}{\partial w_i}$

$= \sum y_t \times \frac{1}{p_t} \frac{\partial p_t}{\partial w_i} + \frac{(1-y_t) \times -1}{1-p_t} \cdot \frac{\partial p_t}{\partial w_i}$

$= \sum_{t=1}^{T} \left(\frac{y_t}{p_t} - \frac{1-y_t}{1-p_t}\right) \frac{\partial p_t}{\partial w_i}$ w/ $\partial p_t = \partial P(Y=1|\vec{x_t})$

$= \partial g(\vec{w} \cdot \vec{x})$

Scanned with CamScanner

chain rule for derivative

$$= \sum_{t=1}^{T} \frac{y_t - \cancel{y_t p_t} - p_t + \cancel{y_t p_t}}{p_t(1-p_t)} \times \frac{\partial g(\vec{w}.\vec{x_t})}{\partial(\vec{w}.\vec{x_t})} \times \underbrace{\frac{\partial(\vec{w}.\vec{x_t})}{\partial w_i}}_{x_{it}}$$

$$= \sum_{t=1}^{T} \frac{y_t - p_t}{p_t(1-p_t)} \times g'(\vec{w}.\vec{x_t}) \times x_{it}$$

$$= \sum_{t=1}^{T} \frac{g'(\vec{w}.\vec{x_t})}{p_t(1-p_t)} \cdot (y_t - p_t) \cdot x_{it}$$ Proved.

b) $g(z) = \frac{1}{1+e^{-z}}$ sigmoid function.

$$g'(z) = \frac{d}{dz}(1+e^{-z})^{-1} = \frac{- - e^{-z}}{(1+e^{-z})^2} = \frac{e^{-z}}{(1+e^{-z})^2}$$

$$= \frac{e^{-z}}{1+e^{-z}} \times \frac{1}{1+e^{-z}} = (1-g(+z)) \times g(z).$$

$$= (1-g(z)) \cdot g(z).$$

(as proved before.

Scanned with CamScanner

from (a)

$$\frac{\partial L}{\partial w_i} = \sum_{t=1}^{T} \left(\frac{f'(\vec{w} \cdot \vec{x_t})}{p_t(1-p_t)} \cdot (y_t - p_t) x_{it} \right.$$

$$= \sum_{t=1}^{T} \frac{f(\vec{w} \cdot \vec{x_t}) \cdot (1 - f(\vec{w} \cdot \vec{x_t}))}{p_t(1-p_t)} \cdot (y_t - p_t) \cdot x_{it}$$

$$= \sum_{t=1}^{T} \frac{p_t \cdot (1-p_t)}{p_t \cdot (1-p_t)} \cdot (y_t - p_t) \cdot x_{it}$$

$$= \sum_{t=1}^{T} (y_t - p_t) x_{it}.$$

Proved

Problem 5.2

$$P(Y=i \mid X=\vec{x}) = \frac{e^{\vec{w_i} \cdot \vec{x_t}}}{\sum_{j=1}^{c} e^{\vec{w_j} \cdot \vec{x_t}}}$$ w/ $\vec{w_i} \cdot \vec{x_t} = (w_{i1}, w_{i2}, w_{i3}, \dots w_{id}) \cdot (x_{t_1}, x_{t_2}, x_{t_3} \text{ — } x_{td})$.

$$y_{it} = \begin{cases} 1 & \text{if } y_t = i \\ 0 & \text{otherwise.} \end{cases}$$

$\mathcal{L} = \sum \log P(y_t \mid \vec{x_t})$ $\quad y \in 1 \longrightarrow k.$ (given).

$$\mathcal{L} = \sum_{t} \prod_{i=1}^{K} \log P(Y=i \mid \vec{x_t})^{y_{it}}$$

$$\mathcal{L} = \sum_{t=1}^{T} \sum_{i=1}^{K} y_{it} \log P(Y=i \mid \vec{x_t}).$$

$$\mathcal{L} = \sum_{t=1}^{T} \sum_{i=1}^{K} y_{it} \log p_{it}$$

In given we have; $p_{it} = \dfrac{e^{\vec{w_i} \cdot \vec{x_t}}}{\sum_{j=1}^{K} e^{\vec{w_j} \cdot \vec{x_t}}}$

Substituting that in $\mathcal{L}$, we get:

$$\mathcal{L} = \sum_{t=1}^{T}\sum_{i=1}^{K} y_{it} \log \frac{e^{\vec{w_i}\cdot\vec{x_t}}}{\sum_{j=1}^{K} e^{\vec{w_j}\cdot\vec{x_t}}}$$

$$= \sum_{t=1}^{T}\left(\sum_{i=1}^{K} y_{it} \log e^{\vec{w_i}\cdot\vec{x_t}} - \log \sum_{j=1}^{K} e^{\vec{w_i}\cdot\vec{x_t}}\right)$$

$$= \sum_{t=1}^{T}\left(\sum_{i=1}^{K} y_{it}\, \vec{w_i}\cdot\vec{x_t} - \log \sum_{j=1}^{K} e^{\vec{w_i}\cdot\vec{x_t}}\right)$$

$$\frac{\partial L}{\partial \vec{w_i}} = \sum_{t=1}^{T}\left(y_{it}\cdot\vec{x_t} - \frac{\vec{x_t}\, e^{\vec{w_i}\cdot\vec{x_t}}}{\sum_{j=1}^{K} e^{\vec{w_i}\cdot\vec{x_t}}}\right)$$

$$= \sum_{t=1}^{T}\left(y_{it}\,\vec{x_t} - \vec{x_t}\cdot p_{it}\right)$$

$$= \sum_{t=1}^{T}\left(\vec{x_t}\,(y_{it} - p_{it})\right)$$

Proved

$$= \sum_{t=1}^{T} (y_{it} - p_{it})\cdot\vec{x_t}\,.$$

Problem 5.3

a) $g(x) = \frac{\alpha}{2}(x - x_*)^2$ w/ $\alpha > 0$

$g'(x) = \frac{\alpha}{2} \times 2 \times (x - x_*) = \alpha(x - x_*)$

$\varepsilon_n = |x_n - x_*|$ $\quad \varepsilon_{n+1} = |x_{n+1} - x_*|$

$= x_n - \eta\alpha(x_n - x_*) - x_*$

$= x_n - \eta\alpha x_n + \eta\alpha x_* - x_*$

$= (1 - \eta\alpha)(x_n - x_*)$

$= (1 - \eta\alpha)\varepsilon_n$

$\boxed{\varepsilon_{n+1} = (1 - \eta\alpha)\varepsilon_n} \Rightarrow$ So $\varepsilon_n = (1 - \eta\alpha)\varepsilon_{n-1}$

$= (1 - \eta\alpha)^2 \varepsilon_{n-1} \dots = (1 - \eta\alpha)^n \varepsilon_0$

$\Rightarrow \varepsilon_n = (1 - \eta\alpha)^n \varepsilon_0.$

b) $\varepsilon_n \to 0 \quad = (1 - \eta\alpha)^n \varepsilon_0 \to 0$

To converge to a minimum which mean $(1 - \eta\alpha)^n \to 0.$

$-1 < (1 - \eta\alpha) < 1$

$-1 < (1 - \eta\alpha) < 1$

Scanned with CamScanner

$$2 < -\eta\alpha < 0$$

$$2 > \eta\alpha > 0$$

$$\frac{2}{\alpha} > \eta > 0 \rightarrow \eta \in [0; 2/\alpha]$$

To get fastest: $(1-\eta\alpha) = 0$

$$\Rightarrow \alpha = \frac{1}{\eta} = \frac{1}{g''(x_n)}.$$ Proved

c) $\varepsilon_n = x_n - x_*$

$$\varepsilon_{n+1} = x_{n+1} - x_*$$

$$= x_n - \eta\alpha x_n + \eta\alpha x_* + \beta(x_n - x_{n-1}) - x_*$$

$$= (x_n - x_*)(1 - \eta\alpha + \beta) - \beta(x_{n-1} - x_*)$$

$$= \varepsilon_n(1 - \eta\alpha + \beta) - \beta \cdot \varepsilon_{n-1}$$

$$= \varepsilon_n(1 - \eta\alpha + \beta) - \varepsilon_{n-1} \cdot \beta.$$ Proved

d) $\alpha = 1 \quad \eta = \frac{4}{9} \quad \beta = \frac{1}{9}$.

from (c) : $\varepsilon_{n+1} = (1 - \eta\alpha + \beta)\varepsilon_n - \beta\varepsilon_{n-1}$

$$= \frac{2}{3}\varepsilon_n - \frac{1}{9}\varepsilon_{n-1}.$$

if $\varepsilon_n = \lambda^n \varepsilon_0$.

$$\lambda^{n+1}\varepsilon_0 = \frac{2}{3}\lambda^n\varepsilon_0 - \frac{1}{9}\lambda^{n-1}\varepsilon_{n-1}.$$

let $t = \lambda^{n-1}$

$$t \cdot \lambda^2 = \frac{2}{3} t \cdot \lambda - \frac{1}{9} t\, \varepsilon_{n-1}.$$

$$\Rightarrow \lambda^2 = \frac{2}{3}\lambda - \frac{1}{9}$$

$$\Rightarrow \lambda = \frac{1}{3} \rightarrow \text{exists}.$$

$$\varepsilon_n = \left(\frac{1}{3}\right)^n \varepsilon_0$$

if $\beta = 0$ $\quad \varepsilon_{n+1} = \frac{5}{9}\varepsilon_n$.

$$\varepsilon_n = \left(\frac{5}{9}\right)^n \varepsilon_0$$

when $\beta > 0$ converges faster than when $\beta = 0$

$\left(\frac{1}{3}\right)^n \rightarrow 0$ faster than $\left(\frac{5}{9}\right)^n$.

CSE250_HW5

November 4, 2022

[146]:

```
import numpy as np
import matplotlib.pyplot as plt
```

[147]:

```
train3 = 'train3.txt'
test3 = 'test3.txt'
train5 = 'train5.txt'
test5 = 'test5.txt'

train3 = np.loadtxt(train3, dtype=int)
test3 = np.loadtxt(test3, dtype=int)
train5 = np.loadtxt(train5, dtype=int)
test5 = np.loadtxt(test5, dtype=int)
```

[148]:

```
Ltrain3 = np.array([0]*(train3.shape[0])).reshape(-1,1)
Ltrain5 = np.array([1]*(train5.shape[0])).reshape(-1,1)

train3 = np.hstack((train3,Ltrain3))
train5 = np.hstack((train5,Ltrain5))
```

[149]:

```
data = np.concatenate((train3,train5))
np.random.shuffle(data)

X_Train = np.array(data[:,:-1])
Y_Train = np.array(data[:,-1])

print(X_Train.shape,Y_Train.shape)
w = np.random.randn(64,1) / 100
# define sigmoid function

def sigmoid(w,x):
    z = np.dot(x,w)
    return(1/(1+np.exp(-z)))
```

```
(1400, 64) (1400,)
```

1 I am using gradient ascent for this problem

[150]:
```
alpha = 0.2 / X_Train.shape[0]
print("Learning rate is :", 0.2 / X_Train.shape[0] )
max_iter = 20000

loss_list = []
error_list = []

best_w = np.zeros((64,1))
best_error = 100

for i in range(max_iter):
    prob = sigmoid(w,X_Train)
    temp = np.log(prob) * Y_Train[:,np.newaxis] + np.log(1-prob) * (1-Y_Train)[:
 ↪,np.newaxis]
    loss = np.sum(temp,axis = 0)
    loss_list.append(loss)
    prob_cur = sigmoid(w,X_Train)
    y_cur = np.where(prob_cur > 0.5,1,0)
    error_rate = np.sum(np.absolute(Y_Train[:,np.newaxis] - y_cur),axis = 0) /␣
 ↪X_Train.shape[0]
    error_list.append(error_rate)
    if error_rate[0] < best_error:
        best_error = error_rate[0]
        best_w = w
    if i%500 == 0:
        print('after ',str(i),' iterations, the log likelihood is␣
 ↪',str(loss[0]))
    temp1 = (Y_Train[:,np.newaxis] - prob) * X_Train
    gradient = np.sum(temp1,axis = 0)
    w = w + alpha * gradient[:,np.newaxis]
loss_list = np.array(loss_list)
error_list = np.array(error_list)
```

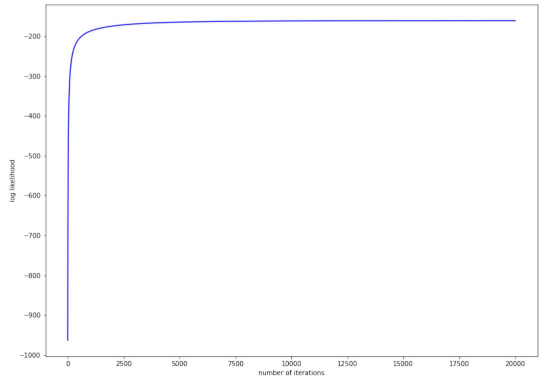
```
Learning rate is : 0.00014285714285714287
after  0  iterations, the log likelihood is  -962.9701411618304
after  500  iterations, the log likelihood is  -206.38451221329063
after  1000  iterations, the log likelihood is  -187.17860053332834
after  1500  iterations, the log likelihood is  -179.0340127070183
after  2000  iterations, the log likelihood is  -174.32373556394415
after  2500  iterations, the log likelihood is  -171.2229424937031
after  3000  iterations, the log likelihood is  -169.03412960490095
after  3500  iterations, the log likelihood is  -167.41958794954405
after  4000  iterations, the log likelihood is  -166.19153145920853
after  4500  iterations, the log likelihood is  -165.23586875924153
after  5000  iterations, the log likelihood is  -164.4788838746091
```

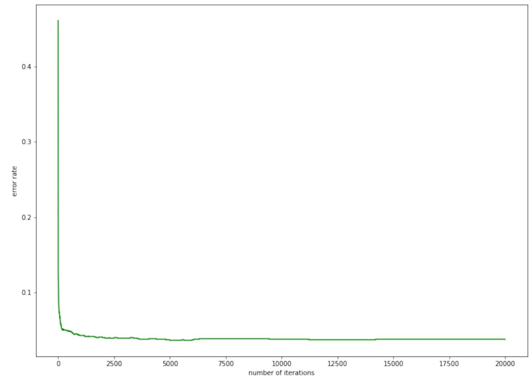
```
after  5500  iterations, the log likelihood is  -163.87071771874446
after  6000  iterations, the log likelihood is  -163.37642032208743
after  6500  iterations, the log likelihood is  -162.9707765672874
after  7000  iterations, the log likelihood is  -162.63515804497362
after  7500  iterations, the log likelihood is  -162.35552695948832
after  8000  iterations, the log likelihood is  -162.1211260284511
after  8500  iterations, the log likelihood is  -161.9235929125052
after  9000  iterations, the log likelihood is  -161.75634618533013
after  9500  iterations, the log likelihood is  -161.61415005017807
after  10000  iterations, the log likelihood is  -161.49279972866918
after  10500  iterations, the log likelihood is  -161.38889015550342
after  11000  iterations, the log likelihood is  -161.2996433354684
after  11500  iterations, the log likelihood is  -161.22277774617604
after  12000  iterations, the log likelihood is  -161.15640835713876
after  12500  iterations, the log likelihood is  -161.09896926134428
after  13000  iterations, the log likelihood is  -161.04915322272026
after  13500  iterations, the log likelihood is  -161.00586402501742
after  14000  iterations, the log likelihood is  -160.96817861048373
after  14500  iterations, the log likelihood is  -160.93531677700759
after  15000  iterations, the log likelihood is  -160.90661676213006
after  15500  iterations, the log likelihood is  -160.88151544885858
after  16000  iterations, the log likelihood is  -160.85953222693846
after  16500  iterations, the log likelihood is  -160.84025576504015
after  17000  iterations, the log likelihood is  -160.8233331156568
after  17500  iterations, the log likelihood is  -160.80846070036915
after  18000  iterations, the log likelihood is  -160.79537681916293
after  18500  iterations, the log likelihood is  -160.78385540134434
after  19000  iterations, the log likelihood is  -160.77370077280204
after  19500  iterations, the log likelihood is  -160.76474325898056
```

[151]:
```
x = np.linspace(0,20000,20000)
```

[152]:
```
fig=plt.figure(figsize=(30,10))
fig.add_subplot(1,2,1)
print(x.shape,loss_list.shape)
plt.plot(x,loss_list,'b')
plt.xlabel('number of iterations')
plt.ylabel('log likelihood')
fig.add_subplot(1,2,2)
plt.plot(x,error_list,'g')
plt.xlabel('number of iterations')
plt.ylabel('error rate')
plt.show()
```

```
(20000,) (20000, 1)
```





[153]:

```
print("Weights")
for i in range(best_w.shape[0]):
    if i % 8 == 0:
        print()
    print('w%d = %.3f'%(i,best_w[i][0]),end = '\t')
```

```
Weights

w0 = -0.890     w1 = -1.391     w2 = -1.152     w3 = -1.098     w4 = -0.745
w5 = -0.772     w6 = 0.815      w7 = 1.700
w8 = 0.065      w9 = -0.092     w10 = 0.202     w11 = -0.069    w12 = -0.335
w13 = 0.684     w14 = -1.218    w15 = -1.275
w16 = 3.220     w17 = 1.361     w18 = 1.352     w19 = 0.221     w20 = 0.625
w21 = -1.912    w22 = -2.384    w23 = -2.427
w24 = 0.785     w25 = 0.408     w26 = 0.552     w27 = -0.267    w28 = -0.488
w29 = -2.154    w30 = 0.352     w31 = -0.029
w32 = 0.472     w33 = 1.042     w34 = 0.046     w35 = -0.318    w36 = -0.625
w37 = -0.207    w38 = -0.396    w39 = -0.322
w40 = 1.119     w41 = -0.189    w42 = -0.311    w43 = -0.064    w44 = 0.103
w45 = -0.806    w46 = 0.761     w47 = -1.411
w48 = 1.364     w49 = -0.594    w50 = 1.240     w51 = 0.555     w52 = 0.399
w53 = -0.291    w54 = 0.215     w55 = -1.130
w56 = 0.524     w57 = 0.276     w58 = 0.866     w59 = 1.669     w60 = 0.475
w61 = 0.630     w62 = 0.518     w63 = -0.461
```

[154]:

```
# test data
label_test3 = np.zeros(test3.shape[0])
label_test5 = np.ones(test5.shape[0])
test3 = np.concatenate((test3,label_test3[:,np.newaxis]),axis = 1)
test5 = np.concatenate((test5,label_test5[:,np.newaxis]),axis = 1)
test = np.concatenate((test3,test5))
np.random.shuffle(test)
x_test = test[:,:-1]
y_test = test[:,-1]
```

```
print(x_test.shape,y_test.shape)
```

```
(800, 64) (800,)
```

#Test 3 and 5 combined

[155]:
```
prob_test = sigmoid(best_w,x_test)
pred_test = np.where(prob_test > 0.5,1,0)
error_rate = np.sum(np.absolute(y_test[:,np.newaxis] - pred_test),axis = 0) /␣
↪x_test.shape[0]
print("testing error rate of 3 and 5 combined: ",str(error_rate[0]))
```

```
testing error rate of 3 and 5 combined:  0.06
```

#Test 5

[156]:
```
prob_test = sigmoid(best_w , test5[:,:-1])
pred_test = np.where(prob_test > 0.5,1,0)
error_rate = np.sum(np.absolute(label_test5[:,np.newaxis] - pred_test),axis =␣
↪0) / test5[:,:-1].shape[0]
print("testing error rate of 5: ",str(error_rate[0]))
```

```
testing error rate of 5:  0.0525
```

#Test 3

[157]:
```
prob_test = sigmoid(best_w , test3[:,:-1])
pred_test = np.where(prob_test > 0.5,1,0)
error_rate = np.sum(np.absolute(label_test3[:,np.newaxis] - pred_test),axis =␣
↪0) / test3[:,:-1].shape[0]
print("testing error rate of 3: ",str(error_rate[0]))
```

```
testing error rate of 3:  0.0675
```

[157]:
```

```