ECE 544- Homework 1

Ahmadreza Eslaminia

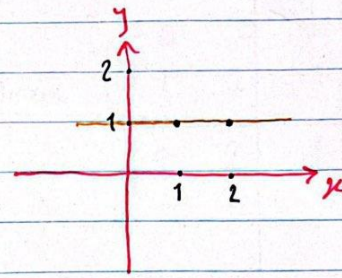
Ae15

Questions:

Ahmadreza Eslaminia ECE 544 HW#1

NetID, AE15

1. [Linear Regression]



a) $w^* = \begin{bmatrix} w_1^* \\ w_2^* \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

b) $X = \begin{bmatrix} x^{(1)} & 1 \\ x^{(2)} & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix}$

X is 2×2 Matrix

y is 2×1 vector

w is a 2×1 Vector

$y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ $w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}_{2\times 1}$

So $\min \frac{1}{2} \sum \left(y - w^T \begin{bmatrix} x \\ 1 \end{bmatrix}\right)^2 = \min \frac{1}{2} \|y - Xw\|_2^2$

c) we want $\min_w \frac{1}{2} \|y - Xw\|_2^2 \xrightarrow[\text{solve}]{} \frac{\partial \frac{1}{2}\|y - Xw\|_2^2}{\partial w} = 0$

$= \frac{1}{2} \frac{\partial (y - Xw)^T (y - Xw)}{\partial w} = \frac{1}{2} \frac{\partial}{\partial w} \left(y^T y - y^T X w - w^T X^T y + w^T X^T X w \right) = 0$

$0 - X^T y - X^T y + (X^T X + (X^T X)^T) w = 0 \Rightarrow 2X^T y = 2X^T X w$

$\Rightarrow w^* = (X^T X)^{-1} X^T y \Rightarrow w^* = \left(\begin{bmatrix} 1 & 2 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix} \right)^{-1} \begin{bmatrix} 1 & 2 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 & -3 \\ -3 & 5 \end{bmatrix} \begin{bmatrix} 3 \\ 2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ ✓

d) code in the attachment.

Solution $S_1 \rightarrow [0, 1]$ ✓

" $S_2 \rightarrow [0, 1]$ ✓

" $S_3 \rightarrow [-4.768 \times 10^{-7}, 1] = [0, 1]$ ✓

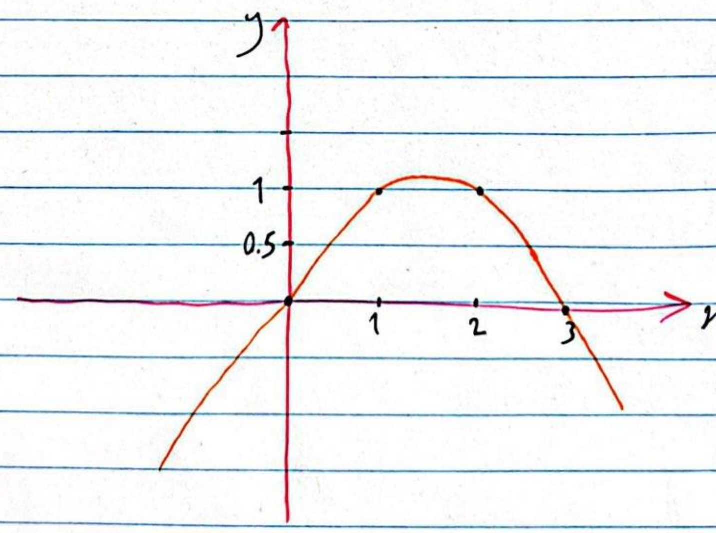
ECE 544

1.e) $X = \begin{bmatrix} (x^{(1)})^2 & x^{(1)} & 1 \\ (x^{(2)})^2 & x^{(2)} & 1 \\ (x^{(3)})^2 & x^{(3)} & 1 \end{bmatrix} \rightarrow$ 3x3 Matrix $= \begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 4 & 2 & 1 \end{bmatrix}$

$y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ y^{(3)} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \rightarrow$ 3x1 vector

from lecture $w^* = (X^T X)^{-1} X^T y = \left(\begin{bmatrix} 0 & 1 & 4 \\ 0 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 4 & 2 & 1 \end{bmatrix} \right)^{-1} \begin{bmatrix} 0 & 1 & 4 \\ 0 & 1 & 2 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$

$= \begin{bmatrix} 17 & 9 & 5 \\ 9 & 5 & 3 \\ 5 & 3 & 3 \end{bmatrix}^{-1} \begin{bmatrix} 5 \\ 3 \\ 2 \end{bmatrix} = \begin{bmatrix} -0.5 \\ 1.5 \\ 0 \end{bmatrix} \Rightarrow \hat{y} = -0.5x^2 + 1.5x$



f) code in attachment

slicing → X (same as above) $X = (x^2, x, 1)$ = torch.tensor([[0,0,1],[1,1,1],[4,2,1]])

P2

2. [Regression] $\{(x^{(i)}, y^{(i)})\}$ $x, y \in \mathbb{R}$ $i = 1, 2, \dots, N$

$$\arg\min \sum_{i=1}^{N} (y^{(i)} - w_1 . x^{(i)} - w_2)^2$$

a) Since we have two unknowns (w_1, w_2) for a unique solution we need at least two observations.

b) for a quadratic model the following Program changes to:

$$\underset{w_1, w_2, w_3}{\arg\min} \sum_{i=1}^{N} (y^{(i)} - w_3 x^{(i)^2} - w_2 x^{(i)} - w_1)^2 \quad \text{I} \quad X = \begin{bmatrix} x^{(1)^2} & x^{(2)^2} & \dots & x^{(N)^2} \\ x^{(1)} & x^{(2)} & & x^{(N)} \\ 1 & 1 & & 1 \end{bmatrix}$$

$$X^T = \begin{bmatrix} x^{(1)^2} & x^{(1)} & 1 \\ \vdots & \vdots & \vdots \\ x^{(N)^2} & x^{(N)} & 1 \end{bmatrix}_{N \times 3} \quad w = \begin{bmatrix} w_3 \\ w_2 \\ w_1 \end{bmatrix}_{3 \times 1} \quad Y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(N)} \end{bmatrix}_{N \times 1}$$

$$\text{I} \Rightarrow \underset{w}{\arg\min} \| Y - X^T W \|_2^2 \longrightarrow \text{For clos form solution:}$$

$$(Y - X^T w)^T (Y - X^T w) = Y^T Y - Y^T X^T w - w^T X Y + w^T X X^T w = A$$

$$0 = \frac{\partial A}{\partial w} = 0 - XY - XY + 2 X X^T w = 0 \Rightarrow X X^T w = XY \Rightarrow w^* = (X X^T)^{-1} X Y$$

c) This would result in over fitting of the model to the train data, where the model captures noise and perform poorly on the new data.
Also using high-degree polynomial would increas the computational complexity and also ~~might~~ might make the model more sensitive to the outliers. ~~So~~ So using high-dimantion polynomial model is not reasonable when we know our data is linear.

P3

2.d) if we know that we have D features, $x^{(i)} \in R^D$ then:

$$\text{equat 3} \rightarrow \underset{w_0, \dots w_D}{\arg\min} \sum_{i=1}^{N} \left(y^{(i)} - w_D x_D^{(i)} - \dots - w_2 x_2^{(i)} - w_1 x_1^{(i)} - w_0\right)^2$$

$$\Rightarrow X = \begin{bmatrix} x_D^{(1)} & x_D^{(2)} & \dots & x_D^{(N)} \\ \vdots & \vdots & & \vdots \\ x_1^{(1)} & x_1^{(2)} & \dots & x_1^{(N)} \\ 1 & 1 & \dots & 1 \end{bmatrix}_{(D+1)\times N} \quad W = \begin{bmatrix} w_D \\ w_{D-1} \\ \vdots \\ w_1 \\ w_0 \end{bmatrix}_{(D+1)\times 1} \quad Y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{N} \end{bmatrix}_{N\times 1}$$

$$\boxed{W \in R^{D+1}}$$

$$\text{Eq 3} \Rightarrow \text{Argmin} \| Y - X^T W \|^2$$

e) if number of features (D) is greater than number of data point (N) then unique solution is not avalable with ordinary least squar function we can modify the model with some regularization terms such as lasso regualarization term which encorage some coefficient to become zero. if we can consider $(D-N)$ features to become zero then we might be able to find a unique solution. Also we can use other Dimention reduction techniqe on our data to get fewer feature.

$$\text{modified model} \rightarrow \text{argmin} \tfrac{1}{2} \| Y - X^T W \|_2^2 + \frac{C}{2} \| W \|^2$$

f) $$\text{argmin} \tfrac{1}{2} \| Y - X^T W \|_2^2 + \frac{C}{2} \| W \|^2 = \text{argmin} \tfrac{1}{2} (Y - X^T W)^T (Y - X^T W) + \frac{C}{2} \| W \|^2 = A$$

$$\rightarrow \frac{\partial A}{\partial W} = 0 \Rightarrow 0 - XY - XY + 2XX^T W + 2CW = 0$$

$$\Rightarrow (XX^T + CI) W^* = XY \Rightarrow W^* = (XX^T + CI)^{-1} XY$$

as you can see if we define C we can get unique solution for that C but we can chose arbitrary C.

P4

3. [softmax Regresion] $D = \{(x^{(i)}, y^{(i)})\}_{i=1}^{|D|}$ $\quad x^{(i)} \in R^d$ $\quad y^{(i)} \in \{1, \dots, k\}$

$$p(y^{(i)} = k \mid x^{(i)}) = \mu_k(x^{(i)}) = \frac{e^{w_k^T x^{(i)}}}{\sum_{j=1}^{k} e^{w_j^T x^{(i)}}}$$

show: $-\log p(D) = -\log p(\{y^{(i)}\}_{i=1}^{|D|} \mid \{x^{(i)}\}_{i=1}^{|D|}) = -\sum_{i=1}^{|D|}\sum_{k=1}^{k} \mathbb{1}_{\{y^{(i)}=k\}} w_k^T x^{(i)} + \sum_{i=1}^{|D|} \log \left(\sum_{j=1}^{k} e^{w_j^T x^{(i)}}\right)$

if we assum all data points in D iid we can write the $-\log P(D)$ as follow:

$$-\log P(D) = -\log\left(\prod_{(x^i, y^i) \in D} P(\{y^i\} \mid \{x^i\})\right) \quad \text{(I)}$$

Then we know each data point distribution of: $p(y_i = k \mid x^{(i)}) = \dfrac{e^{w_k^T x^{(i)}}}{\sum_{j=1}^{k} e^{w_j^T x^{(i)}}}$

where $y^{(i)} = k$ in as $k \in \{1, \dots, k\}$ so we can write each

$\log P(y^{(i)} \mid x^{(i)}) = \sum_{k=1}^{k} \mathbb{1}(y^{(i)} = k) \log \dfrac{e^{w_k^T x^{(i)}}}{\sum_{j=1}^{k} e^{w_j^T x^{(i)}}}$ So we will have:

$$\text{(I)} \rightarrow -\log P(D) = -\sum_{i=1}^{|D|}\sum_{k=1}^{k} \mathbb{1}(y^{(i)} = k) \log\left(\frac{e^{w_k^T x^{(i)}}}{\sum_{j=1}^{k} e^{w_j^T x^{(i)}}}\right)$$

$$\Rightarrow -\log P(D) = -\sum_{i=1}^{|D|}\sum_{k=1}^{k} \mathbb{1}(y^{(i)} = k)\left(\log e^{w_k^T x^{(i)}} - \log \sum_{j=1}^{k} e^{w_j^T x^{(i)}}\right)$$

$$= -\sum_{i=1}^{|D|}\left(\sum_{k=1}^{k} \mathbb{1}(y^{(i)} = k) w_k^T x^{(i)} - \sum_{k=1}^{k} \mathbb{1}(y^{(i)} = k) \log \sum_{j=1}^{k} e^{w_j^T x^{(i)}}\right)$$

$$= -\sum_{i=1}^{|D|}\sum_{k=1}^{k} \mathbb{1}(y^{(i)} = k) w_k^T x^{(i)} + \sum_{i=1}^{|D|}\left(\log \sum_{j=1}^{k} e^{w_j^T x^{(i)}} \sum_{k=1}^{k} \mathbb{1}(y^{(i)} = k)\right)$$

• we know $\sum_{k=1}^{k} \mathbb{1}(y^{(i)} = k) = 1$

So $$-\log P(D) = -\sum_{i=1}^{|D|}\sum_{k=1}^{k} \mathbb{1}\{y^{(i)} = k\} w_k^T x^{(i)} + \sum_{i=1}^{|D|} \log \sum_{j=1}^{k} e^{w_j^T x^{(i)}}$$

3.b) we have discussed the purpose of adding regularization term in previous question (2.e). As we mentioned we wanto to shrink the weights and get them near or to zero so we can overcome over fitting. in this regard when we want this impact along with minimizing negative log function λ has to larger than zero. since the $\sum \|w_k\|$ is a possitive term and for decreasing it λ has to be positive. if $\lambda = 0$ the we would not using the regularization term.

3.c) $lr(w_1, \dots w_k) = \frac{1}{|D|} L(w_1, \dots, w_k) + \lambda \sum_{k=1}^{k} \|w_k\|^2$

show $\rightarrow \nabla_{w_k} lr(w_1, \dots w_k) = 2\lambda w_k + \frac{1}{|D|} \sum_{i=1}^{|D|} \left(\mu_k(x^{(i)}) - \mathbb{1}\{y^{(i)} = k\}\right) x^{(i)}$

From ~~from previous section we have~~ from definition of $L(w_1, \dots w_k)$ from previous sec we have:

$$\nabla_{w_k} lr(w_1, \dots w_k) = 2\lambda w_k + \frac{1}{|D|} \frac{\partial}{\partial w_k} \left(-\sum_{i=1}^{|D|} \left(\sum_{k=1}^{k} \mathbb{1}\{y^{(i)} = k\} w_k^T x^{(i)} - \log \left(\sum_{j=1}^{k} e^{w_j^T x^{(i)}}\right)\right)\right)$$

$$= 2\lambda w_k + \frac{1}{|D|} \left(-\sum_{i=1}^{|D|} \left(\mathbb{1}\{y^{(i)} = k\} x^{(i)} - \frac{e^{w_k^T x^{(i)}} x^{(i)}}{\sum_{j=1}^{k} e^{w_j^T x^{(i)}}} \right)\right)$$

we have $\mu_k(x^{(i)}) = \dfrac{e^{w_k^T x^{(i)}}}{\sum_{j=1}^{k} e^{w_j^T x^{(i)}}}$

then $$= 2\lambda w_k + \frac{1}{|D|} \left(\sum_{i=1}^{|D|} \left(\mu_k^{(x)} - \mathbb{1}(y^{(i)} = k)\right) x^{(i)} \right)$$ ✓

d) $w_k^{(t+1)} \leftarrow w_k^{(t)} - \alpha \nabla_{w_k} lr$

$$\Rightarrow w_k^{(t+1)} \leftarrow w_k^{(t)} - \alpha \left(2\lambda w_k^{(t)} + \frac{1}{|D|} \sum_{i=1}^{|D|} \left(\mu_k^{(t)}(x^{(i)}) - \mathbb{1}(y^{(i)} = k)\right) x^{(i)} \right)$$

P6

ECE544

4) [Binary logistic Regression] $P(y|x) = \dfrac{1}{1+\exp\left(-y w^T \begin{bmatrix} x \\ 1 \end{bmatrix}\right)}$

a) $P(D) = \prod_{(x^{(i)}, y^{(i)}) \in D} P(y^{(i)} | x^{(i)})$

aim: $\max_w P(D) = \max \prod_{(x^{(i)}, y^{(i)}) \in D} P(y^{(i)} | x^{(i)}) = \max_w \prod_{(x,y) \in D} \log P(y^{(i)} | x^{(i)})$

$= \max_w \sum_{(x,y) \in D} \log P(y^{(i)} | x^{(i)}) = \min_w - \sum \log P(y^{(i)} | x^{(i)})$

$= \min \sum_{x^i, y^i \in D} \log\left(1 + \exp\left(-y^{(i)} w^T \begin{bmatrix} x^i \\ 1 \end{bmatrix}\right)\right)$

b) $\min_w f(w) = \sum_{(x^{(i)}, y^{(i)}) \in D} \log\left(1 + \exp\left(-y^{(i)} w^T \begin{bmatrix} x^{(i)} \\ 1 \end{bmatrix}\right)\right)$

$g = \nabla_w f(w) = \sum_{(x^{(i)}, y^{(i)}) \in D} \dfrac{-y^{(i)} \exp\left(-y^{(i)} w^T \begin{bmatrix} x^{(i)} \\ 1 \end{bmatrix}\right)}{1 + \exp\left(-y^{(i)} w^T \begin{bmatrix} x^{(i)} \\ 1 \end{bmatrix}\right)} \begin{bmatrix} x^{(i)} \\ 1 \end{bmatrix}$ I

initialize: $t = 0$ w_{t_0}, step size $= \alpha$

go through $k = 1, 2, 3, \ldots$

compute $g(t) = \nabla_w f(w_t)$ from I

update $w_{t+1} \leftarrow w_t - \alpha g(t)$

$t \leftarrow t+1$

check stop criteria ($\| g_t \| < \varepsilon$ or $k \geq$ stop criteria)

P8.7

4) c) code attached.

f = torch.mean (torch. log (torch. ones_like (tmp) + tmp)

g = torch.mean (((-y))* tmp), ((torch. ones_like (tmp)) + tmp))* X , 1)

$$w^* = \begin{bmatrix} 4.2385 \\ 0.0408 \end{bmatrix}$$

d) from the formulation we can calcute bias for both options as follow

$$w_2^* \rightarrow \begin{cases} (2,1) \rightarrow w_2^* = \text{bias} = 0.0408 \\ (10,1) \rightarrow w_2^* = \text{bias} = 0.0104 \end{cases}$$

This shows if we increas the x or going to right the bias decreases.
however we can find out we still have the same correct classification for previous points so we can consider it is not so much influenc.

if we use linear regression we have:

$$w^* = \begin{cases} (2,1) \rightarrow \begin{bmatrix} 0.7143 \\ -0.1429 \end{bmatrix} \\ (10,1) \rightarrow \begin{bmatrix} 0.1262 \\ -0.0874 \end{bmatrix} \end{cases}$$

as you can see changing of bias (w_2^*) is more in linear regression in comparison to logistic reg. also we can see w_1 change dramatically.

Decison boundry shifterd more in linear regression compare to logistic reg.

that's might be becuse linear regression is more sensetive than logistic to out liers.

P8

ECE 544

4.)e) code is attached.

```
Loss = torch.mean(torch.log(torch.ones_like(tmp)+tmp))

optimizer.step()
optimizer.zero_grad()
```

Solution: $\begin{bmatrix} 4.2940 \\ 0.0341 \end{bmatrix}$ loss: 0.0093 | loss reg 1 → 0.0098

in this new method w_2^* is smaller and w_1^* is larger. by comparing the loss amount between these two approach we can find that second one is more accurate.

$x_i \in R^d$ $\quad w \in R^d$

5. [binary Classification] $D = \{(x_i, y_i)\}$ $\quad i \in \{1, \dots N\}$ $\quad y_i \in \{0,1\}$

$\hat{y}_i = g(w^T x_i) = \frac{1}{1+e^{-w^T x_i}}$ $\qquad P[Y = y_i \mid X = x_i] = (\hat{y}_i)^{y_i} \cdot (1-\hat{y}_i)^{(1-y_i)}$

$w^* = \underset{w}{\arg\min} \left(-\sum_{i=1}^{N} (y_i \cdot \log^{\hat{y}_i}) + (1-y_i) \cdot \log^{1-\hat{y}_i} \right) \equiv \arg\min L(y, X, w)$

we had $\hat{y}_i = \frac{1}{1+e^{-w^T x_i}} \Rightarrow L(y, X, w) = -\sum_{i=1}^{N} \left(y_i \log^{\left(\frac{1}{1+e^{-w^T x_i}}\right)} + (1-y_i) \log^{\left(1 - \frac{1}{1+e^{-w^T x_i}}\right)} \right)$

$= -\sum_{i=1}^{N} \cancel{-y_i \log^{1+e^{-w^T x_i}}} + (1-y_i)(\log^{-w^T x_i} + \cancel{y_i \log^{1+e^{-w^T x_i}}} - \log^{1+e^{-w^T x_i}} = \sum_{i=1}^{N} (1-y_i)(-w^T x_i) - \log^{1+e^{-w^T x_i}} = A$

$\Rightarrow \frac{\partial A}{\partial w} = -\sum_{i=1}^{N} \left((1-y_i) x_i - \frac{e^{-w^T x_i} x_i}{1+e^{-w^T x_i}} \right) = \sum_{i=1}^{N} \left(\frac{x_i + \cancel{x_i e^{-w^T x_i}} - x_i y_i (1+e^{-w^T x_i}) \cancel{- e^{-w^T x_i} x_i}}{1+e^{-w^T x_i}} \right)$

$= \sum_{i=1}^{N} -x_i y_i + \frac{x_i}{1+e^{-w^T x_i}} \Rightarrow \frac{\partial^2 A}{\partial w^2} = \sum_{i=1}^{N} 0 + \frac{x_i^2 e^{-w^T x_i}}{(1+e^{-w^T x_i})^2}$

we know $\frac{x_i^2 e^{-w^T x_i}}{(1+e^{-w^T x_i})^2} \geq 0 \Rightarrow$ convex respect to $\underline{w}$

b) we have $\frac{\partial A}{\partial w} = 0 \Rightarrow \sum_{i=1}^{N} \left(\frac{x_i}{1+e^{-w^T x_i}} - x_i y_i \right) = 0 \rightarrow$ Since we have exponential non linearity we cannot solve it. so there is no closed form solution (the expression cannot ge simpler.)

So we are using gradient descent for this problem as follows

1. chose some initial w, α_k $\rightarrow$ for example $\alpha_k = \frac{1}{L}$ which is constant (we also can use adaptive α_k if needed)
2. for each k: compute d_k as $\nabla f_w = \sum_{i=1}^{N} -x_i y_i + \frac{x_i}{1+e^{-w^T x_i}}$
3. $w^{t+1} \leftarrow w^t + \alpha_k d_k$ (we want $\alpha_k d_k < 0$)
4. check convergence criteria ($\nabla f_w = 0$)

~~[illegible] (N is big) we can use [illegible])~~

P10

Appendix:

A1_LinearRegression

A1_LinearRegression.py × | A1_LinearRegression2.py | A2_LogisticRegression.

C: > DriveA > UIUCcourses > Fall 2023 > ECE 544 pattern recognition > HWS > Hw1 > homew

```python
################################
## Fill in the arguments
################################
res1 = torch.linalg.lstsq(X,y)
print('Solution 1: {}'.format(res1.solution))

# Solution 2
XTX = torch.matmul(torch.transpose(X, 0, 1), X)
XTy = torch.matmul(torch.transpose(X, 0, 1), y)

print('X^TX: {}'.format(XTX))
print('X^Ty: {}'.format(XTy))

################################
## How to compute l and r?
## Dimensions: l (2x2); r (2x1)
################################
l = XTX
r = XTy
res2 = torch.linalg.solve(l,r)
print('Solution 2: {}'.format(res2))

# Solution 3
################################
## What is l and r?
## Dimensions: l (2x2); r (2x1)
################################
l = XTX
r = XTy
res3 = torch.matmul(torch.linalg.inv(l),r)
print("Solution 3: {}".format(res3))
```

```
\Hw1\homework1 (2)'; & 'C:\Users\Ahmadreza\anaconda3\python.exe' 'c:\Users\Ahmadreza\.vscode\exter
py\adapter/../..\debugpy\launcher' '53828' '--' 'C:\DriveA\UIUCcourses\Fall 2023\ECE 544 pattern r
x: tensor([[1.],
        [2.]])
X: tensor([[1., 1.],
        [2., 1.]])
Solution 1: tensor([[-0.],
        [1.]])
X^TX: tensor([[5., 3.],
        [3., 2.]])
X^Ty: tensor([[3.],
        [2.]])
Solution 2: tensor([[0.],
        [1.]])
Solution 3: tensor([[-4.7684e-07],
        [ 1.0000e+00]])
PS C:\DriveA\UIUCcourses\Fall 2023\ECE 544 pattern recognition\HWS\Hw1\homework1 (2)>
```

A1_LinearRegression2

```
A1_LinearRegression.py    A1_LinearRegression2.py ×    A2_LogisticRegression.py    A2_LogisticRegression2.py
C: > DriveA > UIUCcourses > Fall 2023 > ECE 544 pattern recognition > HWS > Hw1 > homework1 (2) > A1_LinearRegression2.py > ...
import torch

################################
## Specify the matrix X
## Dimensions: X (3x3)
################################
X = torch.Tensor([[0, 0, 1], [1, 1, 1], [4, 2, 1]])  # X is [x^2, x, 1]
y = torch.Tensor([[0], [1], [1]])  # y
print(X)
print(y)

# Solution
################################
## Use one of the ways to compute the result
################################
res1 = torch.linalg.lstsq(X, y)
print('Solution : {}'.format(res1.solution))
```

```
        [ 1.0000e+00]])
PS C:\DriveA\UIUCcourses\Fall 2023\ECE 544 pattern recognition\HWS\Hw1\homework1
\Hw1\homework1 (2)'; & 'C:\Users\Ahmadreza\anaconda3\python.exe' 'c:\Users\Ahmadr
py\adapter/../..\debugpy\launcher' '53846' '--' 'C:\DriveA\UIUCcourses\Fall 2023\
tensor([[0., 0., 1.],
        [1., 1., 1.],
        [4., 2., 1.]])
tensor([[0.],
        [1.],
        [1.]])
Solution : tensor([[-5.0000e-01],
        [ 1.5000e+00],
        [-3.9940e-07]])
PS C:\DriveA\UIUCcourses\Fall 2023\ECE 544 pattern recognition\HWS\Hw1\homework1
```

A2_LogistcRegression

DriveA > UIUCcourses > Fall 2023 > ECE 544 pattern recognition > HWS > Hw1 > homework1 (2) > A2_Logistic

```
import torch

torch.manual_seed(1)
X = torch.Tensor([[-1, 1, 2],[1, 1, 1]])
y = torch.Tensor([-1, 1, 1])
w = torch.Tensor([[0.1],[0.1]]) #initialization
alpha = 1

for iter in range(100): # play with the number of iterations

    tmp = torch.exp(torch.matmul(torch.transpose(w,0,1),X)*(-y))
    ###############################
    ## Use tmp to compute f and g. Instead of summing we average the result, i.e.,
    ## complete only inside torch.mean(...) and don't remove this function
    ## Dimensions: f (scalar); g (2)
    ###############################
    f = torch.mean(torch.log(torch.ones_like(tmp))+tmp)
    g = torch.mean(((-y)*tmp)/((torch.ones_like(tmp))+tmp)*X,1)

    print("Loss: {:.6f}; ||g||: {:.6f}".format(f, torch.norm(g)))
    g = g.view(-1,1)
    w = w - alpha*g

print('Solution: {}'.format(w))
```

```
Loss: 0.010858; ||g||: 0.010777
Loss: 0.010741; ||g||: 0.010661
Loss: 0.010626; ||g||: 0.010548
Loss: 0.010514; ||g||: 0.010437
Loss: 0.010404; ||g||: 0.010329
Loss: 0.010296; ||g||: 0.010222
Loss: 0.010191; ||g||: 0.010118
Loss: 0.010087; ||g||: 0.010016
Loss: 0.009986; ||g||: 0.009916
Loss: 0.009887; ||g||: 0.009818
Loss: 0.009789; ||g||: 0.009722
Solution: tensor([[4.2385],
        [0.0408]])
```

A2_LogisticRegression2

A1_LinearRegression.py | A1_LinearRegression2.py | A2_LogisticRegression.py | A2_LogisticRegression2.py

> DriveA > UIUCcourses > Fall 2023 > ECE 544 pattern recognition > HWS > Hw1 > homework1 (2) > A2_LogisticRegression2.py > ...

```python
import torch
import torch.optim as optim

torch.manual_seed(1)
X = torch.Tensor([[-1, 1, 2],[1, 1, 1]])
y = torch.Tensor([-1, 1, 1])
w = torch.Tensor([[0.1],[0.1]]) #initialization
w.requires_grad = True
alpha = 1

optimizer = optim.SGD([w], lr=alpha)
optimizer.zero_grad()

for iter in range(100): # play with the number of iterations

    tmp = torch.exp(torch.matmul(torch.transpose(w,0,1),X)*(-y))
    ################################
    ## loss is the same as f in A2_LogisticRegression.py
    ## Dimensions: loss (scalar)
    ################################
    loss = torch.mean(torch.log(torch.ones_like(tmp))+tmp)

    loss.backward()
    print("Loss: {:.6f}; ||g||: {:.6f}".format(loss, torch.norm(w.grad)))

    ################################
    ## Use two functions within the optimizer instance to perform the update step
    ################################
    optimizer.step()
    optimizer.zero_grad()

print('Solution: {}'.format(w))

```

```
Loss: 0.010434; ||g||: 0.010516
Loss: 0.010324; ||g||: 0.010404
Loss: 0.010216; ||g||: 0.010295
Loss: 0.010111; ||g||: 0.010188
Loss: 0.010008; ||g||: 0.010083
Loss: 0.009906; ||g||: 0.009980
Loss: 0.009807; ||g||: 0.009880
Loss: 0.009710; ||g||: 0.009781
Loss: 0.009615; ||g||: 0.009685
Loss: 0.009522; ||g||: 0.009590
Loss: 0.009430; ||g||: 0.009497
Loss: 0.009340; ||g||: 0.009406
Loss: 0.009252; ||g||: 0.009317
Solution: tensor([[4.2940],
        [0.0341]], requires_grad=True)
PS C:\DriveA\UIUCcourses\Fall 2023\ECE 544 pattern recognition\HWS\Hw1\homework1 (2)>
```