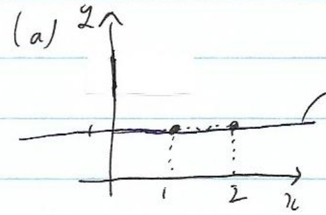
ML Assn 2

20160963 성해빈

1. (a)



optimal $w_1 = 0$, $w_2 = 1$

$w = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ model $y = 1$

(b) when $(x_1, y_1) = (1,1)$ and $(x_2, y_2) = (2,1)$

$$\min_w \frac{1}{2} \sum_{(x,y) \in D} \left(y - w^T \begin{bmatrix} x \\ 1 \end{bmatrix}\right)^2 = \min_w \frac{1}{2} \left[(y_1 - (w_1 x_1 + w_2))^2 + (y_2 - (w_1 x_2 + w_2))^2 \right]$$

$$||$$

when $y : \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$ and $X : \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \end{bmatrix}$

$$\min_w \frac{1}{2} \|y - Xw\|_2^2 = \min_w \frac{1}{2} \left\| \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} - \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \right\|_2^2 = \min_w \frac{1}{2} \left[(y_1 - (w_1 x_1 + w_2))^2 + (y_2 - (w_1 x_2 + w_2))^2 \right]$$

(c) since $\frac{1}{2}\|y - Xw\|_2^2$ is convex regarding w, we find w that $\frac{d}{dw} \frac{1}{2} \|y - Xw\|_2^2 = 0$ to find w that minimize $\frac{1}{2}\|y - Xw\|_2^2$.

note that for vector a, $\frac{d}{dw} w^T a = a$, and for matrix A, $\frac{d}{dw} w^T A w = 2Aw$

$$\frac{d}{dw} \frac{1}{2} \|y - Xw\|_2^2 = \frac{d}{dw} \frac{1}{2} (y - Xw)^T (y - Xw) = \frac{1}{2} \frac{d}{dw} \left(y^T y - \underbrace{w^T X^T y - y^T X w}_{\text{both are } 1\times 1 \text{ scalar, transpose is the same}} + w^T X^T X w\right)$$

$$= \frac{1}{2} \frac{d}{dw} \left(y^T y - 2 w^T X^T y + w^T X^T X w\right)$$

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

$$\therefore w = (X^T X)^{-1} X^T y$$

Let's plug in $y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ and $X = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix}$

$$w = \left(\begin{bmatrix} 1 & 2 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix} \right)^{-1} \begin{bmatrix} 1 & 2 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 5 & 3 \\ 3 & 2 \end{bmatrix}^{-1} \begin{bmatrix} 3 \\ 2 \end{bmatrix} = \begin{bmatrix} 2 & -3 \\ -3 & 5 \end{bmatrix} \begin{bmatrix} 3 \\ 2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

(d)
```
res1 = torch.lstsq(y, X)  # res1[0] : tensor([[-0.], [1.]])
l = torch.matmul(torch.transpose(X, 0, 1), X)   : XᵀX
r = torch.matmul(torch.transpose(X, 0, 1), y)   : Xᵀy
res2 = torch.solve(r, l)  # res2[0] : tensor([[0.], [1.]])
res3 = torch.matmul(torch.inverse(l), r)  # res3: tensor([[-4.7684e-07], [1.0000e+00]])
```

every method seems to be ok, note that the 3rd approach calculates approx values because of floating point arithmetic

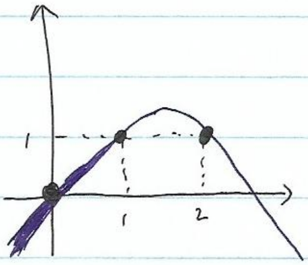
(e) when $(x_1, y_1) = (0,0)$, $(x_2, y_2) = (1,1)$, $(x_3, y_3) = (2,1)$

$$y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} : 1 \times 3 \qquad X = \begin{bmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ x_3^2 & x_3 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 4 & 2 & 1 \end{bmatrix} : 3 \times 3$$

As we did in (c), $w = (X^T X)^{-1} X^T y$

$$w = \left(\begin{bmatrix} 0 & 1 & 4 \\ 0 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 4 & 2 & 1 \end{bmatrix} \right)^{-1} \begin{bmatrix} 0 & 1 & 4 \\ 0 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 17 & 9 & 5 \\ 9 & 5 & 3 \\ 5 & 3 & 3 \end{bmatrix}^{-1} \begin{bmatrix} 5 \\ 3 \\ 2 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{3}{2} & -3 & \frac{1}{2} \\ -3 & \frac{13}{2} & -\frac{3}{2} \\ \frac{1}{2} & -\frac{3}{2} & 1 \end{bmatrix} \begin{bmatrix} 5 \\ 3 \\ 2 \end{bmatrix} = \begin{bmatrix} -\frac{1}{2} \\ \frac{3}{2} \\ 0 \end{bmatrix}$$



model $y = -\frac{1}{2}x^2 + \frac{3}{2}$

(f)
```
X = torch.Tensor([[0,0,1],[1,1,1],[4,2,1]])
y = torch.Tensor([[0],[1],[1]])
res = torch.lstsq(y, X)   : tensor([[[-5.0000e-01][1.5e+00],[-3.5763e-07]]])
```
approximately same value with $\begin{bmatrix} -0.5 \\ 1.5 \\ 0 \end{bmatrix}$

2\. (a) $\arg\min_w f = \arg\min_w \sum_{n=1}^{3} \log(1+\exp(-y_n w^T \begin{bmatrix} x_n \\ 1 \end{bmatrix}))$

when $(x_1, y_1) = (-1,-1)$, $(x_2, y_2) = (1,1)$, $(x_3, y_3) = (2,1)$

(b) $g = \nabla_w f = \nabla_w \sum_{n=1}^{3} \log(1+\exp(-y_n w^T \begin{bmatrix} x_n \\ 1 \end{bmatrix}))$

$$= \sum_{n=1}^{3} \frac{-y_n \exp(-y_n w^T \begin{bmatrix} x_n \\ 1 \end{bmatrix})}{1+\exp(-y_n w^T \begin{bmatrix} x_n \\ 1 \end{bmatrix})} \begin{bmatrix} x_n \\ 1 \end{bmatrix}$$

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

(c) f = torch.mean(torch.log(1 + tmp))

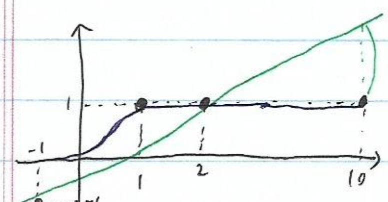
g = torch.mean((((-y) * tmp) / (1 + tmp)) * X, dim = 1)

tmp : $\begin{bmatrix} \exp(-y_1(w_1 x_1 + w_2)) & \exp(-y_2(w_1 x_2 + w_2)) & \exp(-y_3(w_1 x_3 + w_2)) \end{bmatrix}$

f and g are basically the same with (a) and (b), except they use mean rather than sum

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

(d) it would not influence w_2 much. if we used linear regression, it would influence w_2 more than logistic regression.



this is because data points like (10, 1) would make higher errors in linear regression. this higher error would influence the regression results more.

(e) loss = torch.mean(torch.log(1 + tmp))

optimizer.step() optimizer.zero_grad()

I could realize that the results are exactly the same.

(f) $D = \{(-1,-1), (1,1), (2,1)\} \rightarrow D' = \{(-1,0), (1,1), (2,1)\}$

$p(y=1|x) = \frac{1}{1+\exp(-w^T \begin{bmatrix} x \\ 1 \end{bmatrix})}$ $p(y=0|x) = \frac{1}{1+\exp(w^T \begin{bmatrix} x \\ 1 \end{bmatrix})}$ $p(y|x) = \frac{1}{1+\exp((-2y+1) w^T \begin{bmatrix} x \\ 1 \end{bmatrix})}$

loss = Criterion(netOutput, y)

I could see that the results are same.

3. (a) there are 4 constraints since $i = 1, 2, 3, 4$.

$y^{(1)}(w^T x^{(1)} + b) \geq 1 \quad : \quad w_1 + b \geq 1$

$y^{(2)}(w^T x^{(2)} + b) \geq 1 \quad : \quad w_2 + b \geq 1$

$y^{(3)}(w^T x^{(3)} + b) \geq 1 \quad : \quad b \leq -1$

$y^{(4)}(w^T x^{(4)} + b) \geq 1 \quad : \quad w_1 + w_2 - b \geq 1$

(b)

	$b = 0$	$b = -1$	$b = -2$
constraints	$w_1 \geq 1$	$w_1 \geq 2$	$w_1 \geq 3$
	$w_2 \geq 1$	$w_2 \geq 2$	$w_2 \geq 3$
	$0 \leq -1$ (x)	$-1 \leq -1$	$-2 \leq -1$
	$w_1 + w_2 \geq 1$	$w_1 + w_2 \geq 0$	$w_1 + w_2 \geq -1$
feasible w	ϕ	$\{w_1, w_2 \mid w_1 \geq 2, w_2 \geq 2\}$	$\{w_1, w_2 \mid w_1 \geq 3, w_2 \geq 3\}$
optimal w^*		$w^* = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$	$w^* = \begin{bmatrix} 3 \\ 3 \end{bmatrix}$

optimal w^* minimizes $\|w\|_2^2$, so it can be easily infered from these straight forward constraints like $w_1 \geq a$, $w_2 \geq b$.

from $b \in \{0, -1, -2\}$, $w^* = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$ $\left(\|\begin{bmatrix} 2 \\ 2 \end{bmatrix}\|_2^2 < \|\begin{bmatrix} 3 \\ 3 \end{bmatrix}\|_2^2 \right)$

if we found an optimal solution, we can check it by checking if there exists points that $w^T x + b = 1$ and $w^T x + b = -1$.

$i=1 : [2 \ 2] \begin{bmatrix} 1 \\ 0 \end{bmatrix} - 1 = 1$

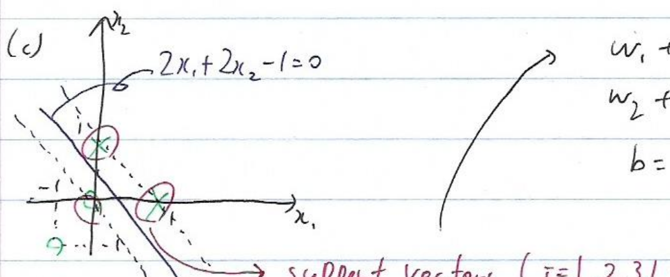
$i=2 : [2 \ 2] \begin{bmatrix} 0 \\ 1 \end{bmatrix} - 1 = 1$

$i=3 : [2 \ 2] \begin{bmatrix} 0 \\ 0 \end{bmatrix} - 1 = -1$

$i=4 : [2 \ 2] \begin{bmatrix} -1 \\ -1 \end{bmatrix} - 1 = -5$

$\rightarrow$ we have both, so $w^* = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$ and $b^* = -1$ seems to be the best solution.

(c)



$w_1 + b = 1$

$w_2 + b = 1$

$b = -1$

$\rightarrow w = \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \ b = -1$

draw hyperplane $2x_1 + 2x_2 - 1 = 0$

(d) the datapoints should be linearly separable.

(e) soft-margin problem with L1 penalty

$$\min_{w,b} \frac{C}{2} \|w\|_2^2 + \sum_{i=1}^{4} \xi_i$$

such that $y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i$ $(\forall i = 1, 2, 3, 4)$

and $\xi_i \geq 0$ $(\forall i = 1, 2, 3, 4)$

reformulate to unconstrained program

$$\underbrace{\min_{w,b} \frac{C}{2} \|w\|_2^2 + \sum_{i=1}^{4} \max\{0, 1 - y^{(i)}(w^T x^{(i)} + b)\}}_{J}$$

gradient of J (when $1 - y^{(i)}(w^T x^{(i)} + b) > 0$)

$$\nabla_w J = Cw + \sum_{i=1}^{4} (-y^{(i)} x^{(i)}) = Cw + \begin{bmatrix} -2 \\ -2 \end{bmatrix} = \begin{bmatrix} Cw_1 - 2 \\ Cw_2 - 2 \end{bmatrix}$$

$$\nabla_b J = \sum_{i=1}^{4} (-y^{(i)}) = 0$$

compute gradient at $w_1 = 2$, $w_2 = 2$, $b = -1$

$$\nabla_w J \big|_{w_1=2, w_2=2} = \begin{bmatrix} 2C - 2 \\ 2C - 2 \end{bmatrix}$$

$$\nabla_b J \big|_{b=-1} = 0$$

when $C = 1$, we get $\nabla J = 0$, as it is the optimal solution.

when C is small, $2C - 2 < 0$ and w will move in a way that increases $\|w\|$ in gradient descent. this means when C is small, loss function focuses in minimizing $\sum_{i=1}^{4} \xi_i$ more than $\frac{C}{2} \|w\|_2^2$, and vice versa.

thus when C is small, soft-margin problem gets more like hard-margin, just like what we could see in $C = 1$ in this case.

(f) at first, I did loss = torch.sum(torch.square(list(net.parameters())[0].data))*C/2)
+ torch.sum(torch.clamp(1 - y*net(torch.transpose(X, 0, 1)).squeeze(), min=0)).

but using list(net.parameters()) makes pytorch not calculate the gradient in regularizer.

So I did iteration in net.parameters(), and added L2 weights in my loss function. it works fine now.

the gradient should be 0 according to (e). but it wasn't. my program found $w = \begin{bmatrix} 0.6674 \\ 0.6674 \end{bmatrix}$ $b = 0.333$ with loss 1.778835. this obviously violates the constraint, and I could see the gradient is not 0.

This problem is because $\max(0, x)$ is not differentiable in $x = 0$, and we are putting values that are 0. getting gradient in this value is actually invalid. To fix this problem, we should specify gradient functions like we did in LogisticRegression.py, not using autograd of pytorch.