\00-CSC413 Homework 1

Name: Kwan Kiu Choy
Student No: 1005005879

1.1

1.1

Assumption: x_1, x_2 are distinct and positive

$W^{(1)} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ $\quad W^{(2)} = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix}$

$\phi^{(1)}(z) =$ absolute value function $\quad \phi^{(2)}(z) = z$ $\quad$ (this is allowed based on @23 on Piazza)

Given input z

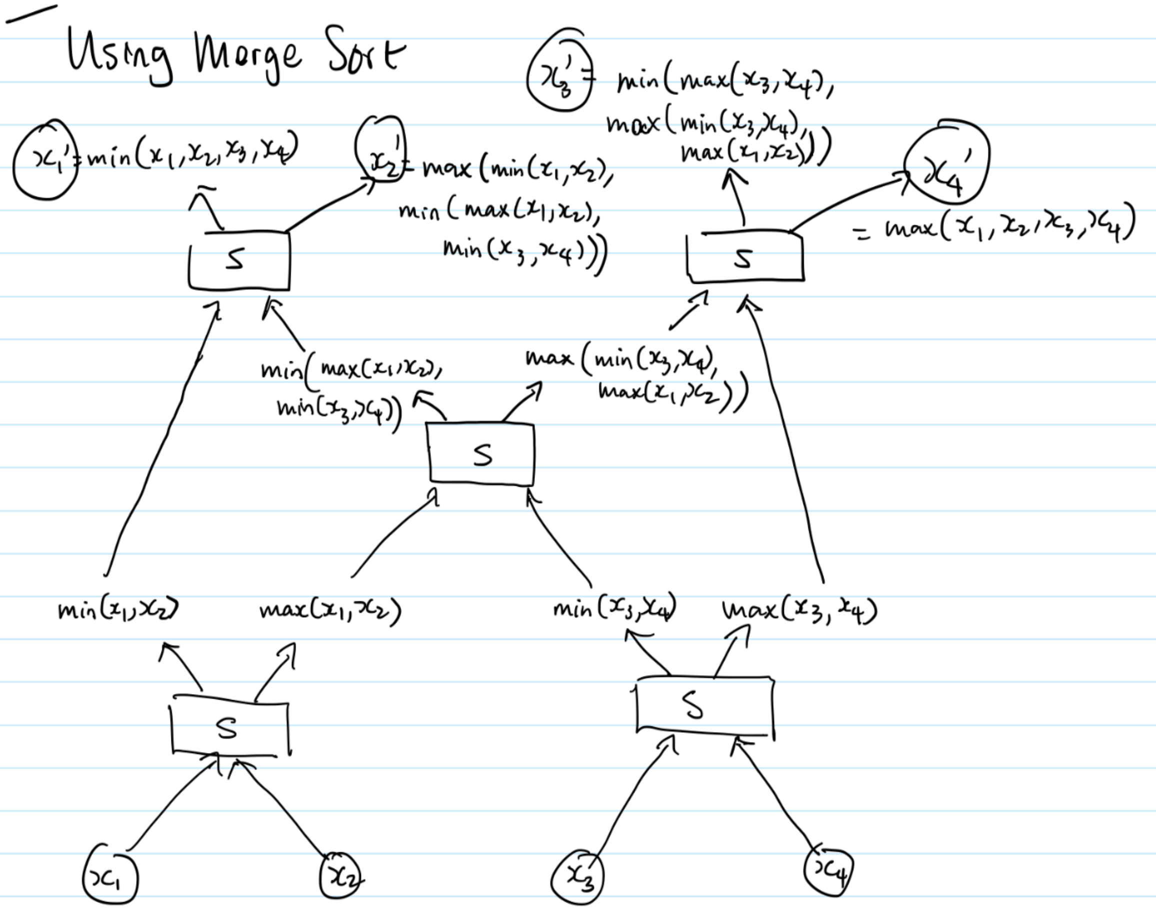
$\phi^{(1)}(z) = |z| \rightarrow$ Note: we don't have to worry $|x_1 + x_2|$ changing the sign of $x_1 + x_2$ since both x_1, x_2 are positive.

$b^{(1)} = \vec{0}$ (zero vector) $\quad b^{(2)} = \vec{0}$ (zero vector)

1.2

1.2

Using Merge Sort



2.1.2

2.1.2 Backward pass

Goal: Compute $\bar{x} = \frac{\partial J}{\partial x}^T$

$\frac{\partial J}{\partial J} = 1$ //

$\frac{\partial J}{\partial S} = \frac{\partial J}{\partial J}\frac{\partial J}{\partial S} = \bar{J} \times (-1) = -\bar{J}$ //

$\frac{\partial J}{\partial y'} = \frac{\partial J}{\partial J}\frac{\partial J}{\partial S}\frac{\partial S}{\partial y'} = \bar{S} \left(\begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} \frac{1}{y_t} \right)$ one hot vector where only the t^{th} position has a 1 and other position is 0.

$= \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} \frac{1}{y_t} \bar{S}$ //

$\frac{\partial J}{\partial y} = \frac{\partial J}{\partial J}\frac{\partial J}{\partial S}\frac{\partial S}{\partial y'}\frac{\partial y'}{\partial y} = \bar{y'} \, softmax'(y)$

$\frac{\partial J}{\partial g} = \frac{\partial J}{\partial J}\frac{\partial J}{\partial S}\frac{\partial S}{\partial y'}\frac{\partial y'}{\partial y}\frac{\partial y}{\partial g} = \bar{y} W_3$

$\frac{\partial J}{\partial h_1} = \frac{\partial J}{\partial J}\frac{\partial J}{\partial S}\frac{\partial S}{\partial y'}\frac{\partial y'}{\partial y}\frac{\partial y}{\partial g}\frac{\partial g}{\partial h_1} = h_2 \circ \bar{g}$ element-wise multiplication

$\frac{\partial J}{\partial h_2} = \frac{\partial J}{\partial J}\frac{\partial J}{\partial S}\frac{\partial S}{\partial y'}\frac{\partial y'}{\partial y}\frac{\partial y}{\partial g}\frac{\partial g}{\partial h_2} = h_1 \circ \bar{g}$

derivative of RELU can be split into cases $RELU'(z) = \begin{cases} 1 & z > 0 \\ 0 & z < 0 \\ \text{undefined} & z = 0 \end{cases}$

$\frac{\partial J}{\partial z_1} = \frac{\partial J}{\partial J}\frac{\partial J}{\partial S}\frac{\partial S}{\partial y'}\frac{\partial y'}{\partial y}\frac{\partial y}{\partial g}\frac{\partial g}{\partial h_1}\frac{\partial h_1}{\partial z_1} = \bar{h}_1 \frac{\partial h_1}{\partial z_1} = \bar{h}_1$ or 0 or undefined //

$\frac{\partial J}{\partial z_2} = \frac{\partial J}{\partial J}\frac{\partial J}{\partial S}\frac{\partial S}{\partial y'}\frac{\partial y'}{\partial y}\frac{\partial y}{\partial g}\frac{\partial g}{\partial h_2}\frac{\partial h_2}{\partial z_2} = \bar{h}_2 \frac{e^{z_2}}{(1+e^{z_2})^2} = \bar{h}_2 \left(\sigma(z_2)\left(1 - \sigma(z_2)\right)\right)$

$\frac{\partial J}{\partial x} = \frac{\partial J}{\partial z_1}\frac{\partial z_1}{\partial x} + \frac{\partial J}{\partial z_2}\frac{\partial z_2}{\partial x} + \frac{\partial J}{\partial y}\frac{\partial y}{\partial x}$

$= \bar{z}_1 W_1 + \bar{z}_2 W_2 + \bar{y} W_4$

Therefore, $\bar{x} = \frac{\partial J}{\partial x}^T = W_1^T \bar{z}_1^T + W_2^T \bar{z}_2^T + W_4^T \bar{y}^T$ //

2.2.1

2.2.1 Naive Computation

$$\frac{\partial J}{\partial W_2} = \left(\frac{\partial J}{\partial J}\frac{\partial J}{\partial y}\frac{\partial y}{\partial W_2}\right)^T$$
$$= \left(\bar{y}\,\frac{\partial y}{\partial W_2}\right)^T$$
$$= \left(\bar{y}\,h^T\right)^T$$
$$= \left(\begin{bmatrix}1\\1\\1\end{bmatrix}\begin{bmatrix}8 & 1 & 0\end{bmatrix}\right)^T$$
$$= \begin{bmatrix}8 & 8 & 8\\1 & 1 & 1\\0 & 0 & 0\end{bmatrix}$$

$$z = \begin{bmatrix}1 & 2 & 1\\-2 & 1 & 0\\1 & -2 & -1\end{bmatrix}\begin{bmatrix}1\\3\\1\end{bmatrix} = \begin{bmatrix}8\\1\\-6\end{bmatrix}$$

$$h = \begin{bmatrix}8\\1\\0\end{bmatrix}$$

$$\frac{\partial J}{\partial h} = W_2^T\bar{y} = \begin{bmatrix}-2 & 1 & -3\\4 & -2 & 4\\1 & -3 & 6\end{bmatrix}\begin{bmatrix}1\\1\\1\end{bmatrix} = \begin{bmatrix}-4\\6\\4\end{bmatrix}$$

$$\frac{\partial J}{\partial z} = \bar{h} \circ \mathrm{RELU}'(z) = \begin{bmatrix}-4\\6\\4\end{bmatrix} \circ \begin{bmatrix}1\\1\\0\end{bmatrix} = \begin{bmatrix}-4\\6\\0\end{bmatrix}$$

$$\frac{\partial J}{\partial W_1} = \left(\bar{z}\,x^T\right)^T = \left(\begin{bmatrix}-4\\6\\0\end{bmatrix}\begin{bmatrix}1 & 3 & 1\end{bmatrix}\right)^T = \begin{bmatrix}-4 & 12 & -4\\6 & 18 & 6\\0 & 0 & 0\end{bmatrix}^T$$

The Jacobian matrix for $\frac{\partial J}{\partial W_1}$ and $\frac{\partial J}{\partial W_2}$ are

$$\frac{\partial J}{\partial W_1} = \begin{bmatrix}-4 & 6 & 0\\12 & 18 & 0\\-4 & 6 & 0\end{bmatrix} \qquad \text{Norm: } \left\|\frac{\partial J}{\partial W_1}\right\|_F^2 = 572$$

$$\frac{\partial J}{\partial W_2} = \begin{bmatrix}8 & 8 & 8\\1 & 1 & 1\\0 & 0 & 0\end{bmatrix} \qquad \text{Norm: } \left\|\frac{\partial J}{\partial W_2}\right\|_F^2 = 195$$

2.2.2

2.2.2

Efficient Computation

$$\left\|\frac{\partial J}{\partial W_1}\right\|_F^2 = (x^T x)(\bar{z}^T \bar{z})$$

$$= \begin{bmatrix} 1 & 3 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \\ 1 \end{bmatrix} \times \begin{bmatrix} 4 & 6 & 0 \end{bmatrix} \begin{bmatrix} 4 \\ 6 \\ 0 \end{bmatrix}$$

$$= 11 \times (16 + 36)$$

$$= 11 \times 52$$

$$= 572$$

$$\left\|\frac{\partial J}{\partial W_2}\right\|_F^2 = (h^T h)(\bar{y}^T \bar{y})$$

$$= \begin{bmatrix} 8 & 1 & 0 \end{bmatrix} \begin{bmatrix} 8 \\ 1 \\ 0 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$$= 65 \times 3$$

$$= 195$$

Both norms computed what we got in 2.1.3 are the same. with what we have.

2.2.3

2.2.3

$\boxed{1}$ Forward pass:

i) T - Naive:

Each layer to layer transition

$\underset{D\times D}{W}\ \underset{D\times 1}{(\text{input})} = \underset{D}{\text{Output}}$

memory needed: $(D^2+D)N$
$= O(ND^2+ND) = O(ND^2)$

For each input, each layer to layer transition we have D^2 scaler multiplications

$T_{\text{Naive Forward}} = KND^2$

ii) T - Efficient:

Each layer to layer transition same forward pass with Naive

so $T_{\text{Efficient Forward}} = KND^2$

memory needed
$(D^2+D)N$
$= O(ND^2+ND) = O(ND^2)$

$\boxed{2}$ Backward pass

i) T-Naive:

Each layer to layer transition

$2\times$ no of scaler multiplications in forward pass

$= 2KND^2$

memory needed weight matrix and

$= O(NKD^2)$

ii) T-Efficient:

Since we don't want to update the weight / bias we don't need to compute the $\frac{\partial J}{\partial W_K}$ Jacobian matrix.

memory needed $= O(NKD^2)$

Hence T-Efficient $= NKD^2$

$\boxed{3}$ Compute Norm

T-Naive

$\frac{\partial J}{\partial W_K}$ → will be a $D\times D$ matrix

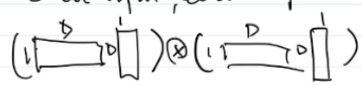
$\frac{\partial J}{\partial W_K}^T \frac{\partial J}{\partial W_K}$ → D^3 scaler multiplications
→ D^2 effective scaler multiplication

Total: KND^3 scaler multiplication,

We have
N $D\times D^2$ matrices
Memory:
$O(ND^2)$

T-Efficient

Each input, each weight matrix



$= 2D + 1$

Total: $NK(2D+1) = 2NKD + NK$

Memory Space:
$2ND$
$= O(ND)$

Summary	T-Naive	T-Efficient	M-naive	M-efficient
Forward Pass	KND^2	KND^2	$O(ND^2)$	$O(ND^2)$
Backward pass	$2KND^2$	KND^2	$O(KND^2)$	$O(KND^2)$
Compute Norm	KND^2	$2NKD + NK$	$O(ND^2)$	$O(ND)$

3.2.1

3.2.1

$$J = \frac{1}{n} \| X\hat{w} - t \|_2^2$$

$$\frac{\partial J}{\partial \hat{w}} = \frac{2}{n} X^T (X\hat{w} - t)$$

For the gradient descent process to termniate, $\frac{\partial J}{\partial \hat{w}} = 0$

$$\frac{\partial J}{\partial \hat{w}} = 0$$

$$\frac{2}{n} X^T (X\hat{w} - t) = 0$$

$$X^T X \hat{w} - X^T t = 0$$

$$X^T X \hat{w} = X^T t$$

$$\hat{w} = (X^T X)^{-1} X^T t$$

Since $d < n$, we know that $X^T X$ is invertible and hence we get $\hat{w} = (X^T X)^{-1} X^T t$.

3.2.2

3.2.2

$$\text{Error} = \frac{1}{n}\|X\vec{w} - t\|_2^2$$
$$= \frac{1}{n}\|X(X^TX)^{-1}X^Tt - t\|_2^2$$
$$= \frac{1}{n}\|(X(X^TX)^{-1}X^T - I)t\|_2^2$$
$$= \frac{1}{n}\|X(X^TX)^{-1}X^T - I)(Xw^* + \epsilon)\|_2^2 \longrightarrow \text{By Associative property of matrix multiplication}$$
$$= \frac{1}{n}\|X\cancel{(X^TX)^{-1}X^TX}w^* - Xw^*) + (X(X^TX)^{-1}X^T - I)\epsilon\|_2^2$$
$$= \frac{1}{n}\|\cancel{Xw^*} - \cancel{Xw^*} + (X(X^TX)^{-1}X^T - I)\epsilon\|_2^2$$
$$= \frac{1}{n}\|(X(X^TX)^{-1}X^T - I)\epsilon\|_2^2$$

Expected Error:

$$\text{Error} = tr(\text{Error})$$
$$\mathbb{E}(\text{Error}) = tr(\mathbb{E}(\text{Error}))$$

→ Let $v = (X(X^TX)^{-1}X^T - I)\epsilon$

$$\mathbb{E}(\text{Error}) = tr(\mathbb{E}(\tfrac{1}{n}\|v\|_2^2))$$
$$= \frac{1}{n}tr(\mathbb{E}(v^Tv))$$
$$= \frac{1}{n}tr(\mathbb{E}(\epsilon^T(X(X^TX)^{-1}X^T - I)^T(X(X^TX)^{-1}X^T - I)\epsilon))$$

Making use of the cyclic property of trace

$$= \frac{1}{n}tr((X(X^TX)^{-1}X^T - I)^T(X(X^TX)^{-1}X^T - I)\,\mathbb{E}(\epsilon^T\epsilon))$$

→ Let $A = X(X^TX)^{-1}X^T - I$

$$= \frac{1}{n}tr(A^TA\sigma^2)$$
$$= \frac{\sigma^2}{n}tr(A^TA) \longrightarrow (*)$$

→ Let $B = X(X^TX)^{-1}X^T$

$$A = B - I$$
$$A^TA = (B - I)^T(B - I)$$
$$= (B^T - I^T)(B - I)$$
$$= B^TB - B - B^T + I$$
$$= X\cancel{(X^TX)^{-1}X^T}X(X^TX)^{-1}X^T - X(X^TX)^{-1}X^T - X(X^TX)^{-1}X^T + I$$
$$= I - B^T$$

$$tr(A^TA) = tr(I - B^T)$$
$$= tr(I) - tr(B^T)$$
$$= rank(I) - rank(B^T)$$
$$= n - d \longrightarrow (1)$$

Substituting (1) into (*), we get

$$\mathbb{E}(\text{Error}) = \frac{\sigma^2}{n}(tr(A^TA))$$
$$= \frac{\sigma^2}{n}(n - d)$$

3.3.2

<u>3.3.2</u>

Let $\hat{w} = X^T A$ where $A \in \mathbb{R}^w$

$$J = \frac{1}{n} \| X\hat{w} - t \|_2^2$$

$$\frac{\partial J}{\delta w} = \frac{2}{w} X^T \| X\hat{w} - t \|_2^2$$

$$\frac{2}{n} X^T \| X\hat{w} - t \|_2^2 = 0$$

$$\frac{2}{n} X^T (XX^T A - t) = 0$$

$$\frac{2}{n} XX^T (XX^T A - t) = 0$$

$$XX^T A - t = 0$$

$$XX^T A = t$$

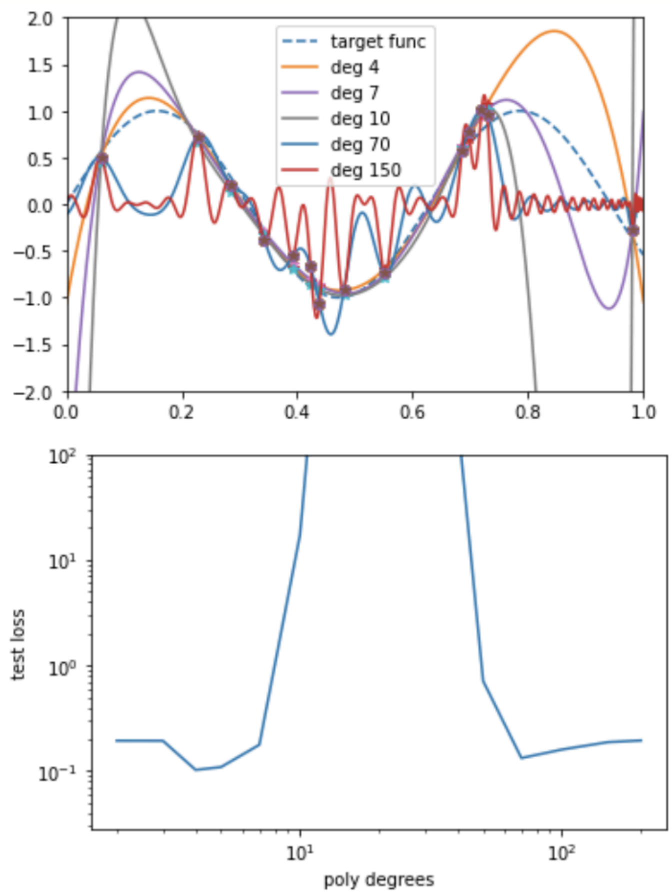
$$A = (XX^T)^{-1} t$$

$$\hat{w} = X^T (XX^T)^{-1} t$$

Since $d > n$, we know that XX^T is invertible and hence $\hat{w} = X^T (XX^T)^{-1} t$ //

3.3.4

```
def fit_poly(X, d,t):
 X_expand = poly_expand(X, d=d, poly_type = poly_type)
 n = X.shape[0]
 if d > n:
   inv = np.linalg.inv(np.dot(X_expand, X_expand.T))
   head = np.dot(X_expand.T, inv)
   W = np.dot(head, t)
 else:
   inv = np.linalg.inv(np.dot(X_expand.T, X_expand))
   head = np.dot(inv, X_expand.T)
   W = np.dot(head, t)
 return W
```



Analysis:
From this diagram above, we know that high degrees don’t always lead to test error. Based on the test loss vs poly degrees diagram, we can see a decrease in test loss after degrees exceed a certain value.