CSC413 Homework 1

Name: Yulin Wang

Student #: 1003942326

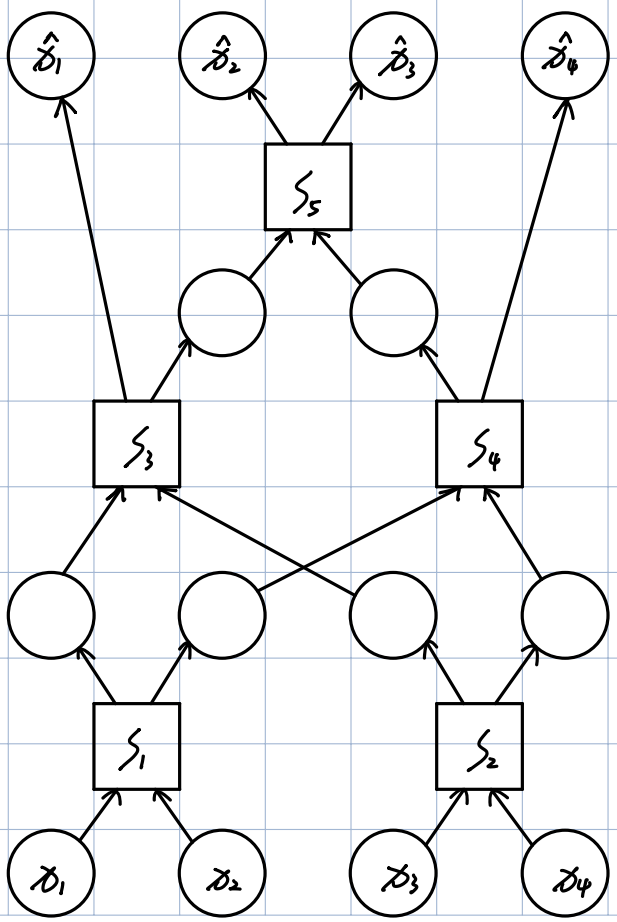
1.1 Sort two numbers

Two weight matrices: $W^{(1)} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} \end{bmatrix}$ $W^{(2)} = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$

Two bias vectors: $b^{(1)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ $b^{(2)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

Two activation functions: $\phi^{(1)}(z) = |z|$ $\phi^{(2)} = z$

1.2 Perform Sort



2.1.2 Backward Pass

$$\bar{J} = 1$$

$$\bar{S} = \bar{J}\frac{\partial J}{\partial S} = -\bar{J} = -1$$

$$\bar{y'} = \bar{S}\frac{\partial S}{\partial y'} = \bar{S}\,[0, \ldots, 0, 1, 0, \ldots, 0]^T \circ [0, \ldots, 0, \frac{1}{y'_k}, 0, \ldots, 0]^T$$

$$= -\bar{J}\,[0, \ldots, 0, \frac{1}{y'_k}, 0, \ldots, 0]^T$$

where $\frac{1}{y'_k}$ is at the k^{th} index

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

$$\bar{g} = \bar{y}\frac{\partial y}{\partial g} = [W^{(3)}]^T\bar{y}$$

$$\bar{h_1} = \bar{g}\frac{\partial g}{\partial h_1} = \bar{g} \circ h_2$$

$$\bar{h_2} = \bar{g}\frac{\partial g}{\partial h_2} = \bar{g} \circ h_1$$

$$\bar{z_1} = \bar{h_1}\frac{\partial h_1}{\partial z_1} = \bar{h_1} \circ ReLU'(z_1) \qquad \text{where } ReLU'(z_1) = \begin{cases} 1, & \text{if } z_1 > 0 \\ 0, & \text{otherwise} \end{cases}$$

$$= \bar{h_1} \circ I(z_1 > 0)$$

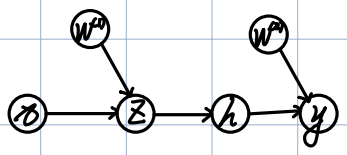
$$\bar{z_2} = \bar{h_2}\frac{\partial h_2}{\partial z_2} = \bar{h_2} \circ \sigma'(z_2) = \bar{h_2} \circ \sigma(z_2) \circ (1 - \sigma(z_2))$$

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

$$= [W^{(1)}]^T\bar{z_1} + [W^{(2)}]^T\bar{z_2} + [W^{(4)}]^T\bar{y}$$

2.2.1 Naive Computation

Draw the computation graph first:



$$z = W^{(1)}x = \begin{bmatrix} 1 & 2 & 1 \\ -2 & 1 & 0 \\ 1 & -2 & -1 \end{bmatrix}\begin{bmatrix} 1 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 8 \\ 1 \\ -6 \end{bmatrix}$$

$$\Rightarrow \quad h = ReLU(z) = \begin{bmatrix} 8 \\ 1 \\ 0 \end{bmatrix} \quad \text{and} \quad ReLU'(z) = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

$$\bar{h} = \bar{y}\frac{dy}{dh} = [W^{(2)}]^T\bar{y} = \begin{bmatrix} -2 & 4 & 1 \\ 1 & -2 & -3 \\ -3 & 4 & 6 \end{bmatrix}^T\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -2 & 1 & 3 \\ 4 & -2 & 4 \\ 1 & -3 & 6 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -4 \\ 6 \\ 4 \end{bmatrix}$$

$$\bar{z} = \bar{h}\frac{dh}{dz} = \bar{h} \circ ReLU'(z) = \begin{bmatrix} -4 \\ 6 \\ 4 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} -4 \\ 6 \\ 0 \end{bmatrix}$$

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

$$\frac{\partial J}{\partial W^{(2)}} = (\bar{y}h^T)^T = \left[\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}\begin{bmatrix} 8 & 1 & 0 \end{bmatrix}\right]^T = \begin{bmatrix} 8 & 1 & 0 \\ 8 & 1 & 0 \\ 8 & 1 & 0 \end{bmatrix}^T = \begin{bmatrix} 8 & 8 & 8 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\Rightarrow \quad \left\|\frac{\partial J}{\partial W^{(1)}}\right\|_F^2 = \sum_{i=1}^{3}\sum_{j=1}^{3}|a_{ij}|^2 = 4^2+6^2+12^2+18^2+4^2+6^2 = 572$$

$$\left\|\frac{\partial J}{\partial W^{(2)}}\right\|_F^2 = \sum_{i=1}^{3}\sum_{j=1}^{3}|a_{ij}|^2 = (8^2+1^2)\times 3 = 195$$

2.2.2 Efficient Computation

From 2.2.1, we have: $\bar{z} = \begin{bmatrix} -4 \\ 6 \\ 0 \end{bmatrix}$

Plug in x and $\bar{z}$:

$$\left\| \frac{\partial J}{\partial W^{(2)}} \right\|_F^2 = \|x\|_2^2 \|\bar{z}\|_2^2 = (1^2+3^2+1^2) \times (4^2+6^2+0^2) = 572$$

Similarly:

$$\left\| \frac{\partial J}{\partial W^{(2)}} \right\|_F^2 = \text{trace}\left(\frac{\partial J}{\partial W^{(2)}}^T \frac{\partial J}{\partial W^{(2)}} \right) \quad \text{(Definition)}$$

$$= \text{trace}(\bar{y} h^T h \bar{y}^T)$$

$$= \text{trace}(h^T h \bar{y}^T \bar{y}) \quad \text{(Cyclic Property of Trace)}$$

$$= (h^T h)(\bar{y}^T \bar{y}) \quad \text{(Scalar Multiplication)}$$

$$= \|h\|_2^2 \|\bar{y}\|_2^2$$

From 2.2.1, we have: $h = \begin{bmatrix} 8 \\ 1 \\ 0 \end{bmatrix}$ $\bar{y} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$

Then:

$$\left\| \frac{\partial J}{\partial W^{(2)}} \right\|_F^2 = (1^2+1^2+1^2) \times (8^2+1^2+0^2) = 195$$

2.2.3 Complexity Analysis

	T (Naive)	T (Efficient)	M (Naive)	M (Efficient)
Forward Pass	NKD^2	NKD^2	$O(KD^2)$	$O(KD^2)$
Backward Pass	$(2K-1)ND^2$	$(K-1)ND^2$	$O(NKD^2)$	$O(KD^2)$
Gradient Norm Computation	NKD^3	$(2D+1)NK$	$O(NKD^2)$	$O(NKD)$

3.2 Underparameterized Model

3.2.1

Assume training converges, then $\frac{\partial J}{\partial \hat{w}} = 0$

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

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

$$\Leftrightarrow \quad \hat{w} = (X^T X)^{-1} X^T t \qquad ; \text{since } X^T X \text{ is invertible when } d < n$$

3.2.2

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

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

$$= \frac{1}{n} \| X(X^T X)^{-1} X^T (Xw^* + \varepsilon) - (Xw^* + \varepsilon) \|_2^2 \qquad ; \text{since } t_i = w^{*T} x_i + \varepsilon_i$$

$$= \frac{1}{n} \| X(X^T X)^{-1} X^T X w^* + X(X^T X)^{-1} X^T \varepsilon - Xw^* - \varepsilon \|_2^2$$

$$= \frac{1}{n} \| Xw^* + X(X^T X)^{-1} X^T \varepsilon - Xw^* - \varepsilon \|_2^2$$

$$= \frac{1}{n} \| X(X^T X)^{-1} X^T \varepsilon - \varepsilon \|_2^2$$

$$= \frac{1}{n} \| (X(X^T X)^{-1} X^T - I) \varepsilon \|_2^2$$

Thus, we've showed that $\text{Error} = \frac{1}{n} \| (X(X^T X)^{-1} X^T - I) \varepsilon \|_2^2$

$$E(\text{Error}) = E\left[\frac{1}{n} \| (X(X^T X)^{-1} X^T - I) \varepsilon \|_2^2 \right]$$

$$= E\left[\frac{1}{n} [(X(X^T X)^{-1} X^T - I) \varepsilon]^T [(X(X^T X)^{-1} X^T - I) \varepsilon] \right]$$

$$= E\left[\frac{1}{n} [\varepsilon^T X(X^T X)^{-1} X^T - \varepsilon^T] [X(X^T X)^{-1} X^T \varepsilon - \varepsilon] \right]$$

$$= \frac{1}{n} E\left[\varepsilon^T X(X^T X)^{-1} X^T X (X^T X)^{-1} X^T \varepsilon - \varepsilon^T X(X^T X)^{-1} X^T \varepsilon - \varepsilon^T X(X^T X)^{-1} X^T \varepsilon + \varepsilon^T \varepsilon \right]$$

$$= \frac{1}{n} E\left[\varepsilon^T X(X^T X)^{-1} X^T \varepsilon - \varepsilon^T X(X^T X)^{-1} X^T \varepsilon - \varepsilon^T X(X^T X)^{-1} X^T \varepsilon + \varepsilon^T \varepsilon \right]$$

$$= \frac{1}{n} E\left[\varepsilon^T \varepsilon - \varepsilon^T X(X^T X)^{-1} X^T \varepsilon \right]$$

$$= \frac{1}{n} E[\varepsilon^T \varepsilon] - \frac{1}{n} E\left[\varepsilon^T X(X^T X)^{-1} X^T \varepsilon \right]$$

3.2.2 Continued

$$E[\varepsilon^T\varepsilon] = E(\sum_{i=1}^{n}\varepsilon_i^2) = \sum_{i=1}^{n}E(\varepsilon_i^2) = n\sigma^2$$

$$E[\varepsilon^T X(X^TX)^{-1}X^T\varepsilon] = E(\text{trace}[\varepsilon\varepsilon^T X(X^TX)^{-1}X^T])$$

$$= E(\text{trace}[X^T\varepsilon\varepsilon^T X(X^TX)^{-1}])$$

$$= E(\text{trace}[X^T\sigma^2 I_n X(X^TX)^{-1}])$$

$$= E(\text{trace}[\sigma^2 X^TX(X^TX)^{-1}])$$

$$= \sigma^2 E(\text{trace}[X^TX(X^TX)^{-1}])$$

$$= \sigma^2 \text{trace}(I_d)$$

$$= \sigma^2 \cdot d$$

$$\Rightarrow E(\text{Error}) = \frac{1}{n}E[\varepsilon^T\varepsilon] - \frac{1}{n}E[\varepsilon^T X(X^TX)^{-1}X^T\varepsilon]$$

$$= \frac{1}{n}\cdot n\sigma^2 - \frac{1}{n}\sigma^2\cdot d$$

$$= \sigma^2 - \frac{d}{n}\sigma^2$$

$$= \left(\frac{n-d}{n}\right)\sigma^2$$

3.3 Overparameterized Model

3.3.2

Since we assume that the gradient is spanned by the rows of X, starting from zero initialization, each iterate and the final solution of the gradient descent can be written as a linear combination of rows of X.

By writing $\hat{w} = X^T a$ for some $a \in R^n$:

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

$\Leftrightarrow XX^T(X\hat{w} - t) = 0$

$\Leftrightarrow X\hat{w} - t = 0$

$\Leftrightarrow XX^T a - t = 0$

$\Leftrightarrow a = (XX^T)^{-1} t$

$\Leftrightarrow \hat{w} = X^T(XX^T)^{-1} t$; where we utilized the invertibility of XX^T when $d > n$.

3.3.4

Code snippets for the fit_poly function:

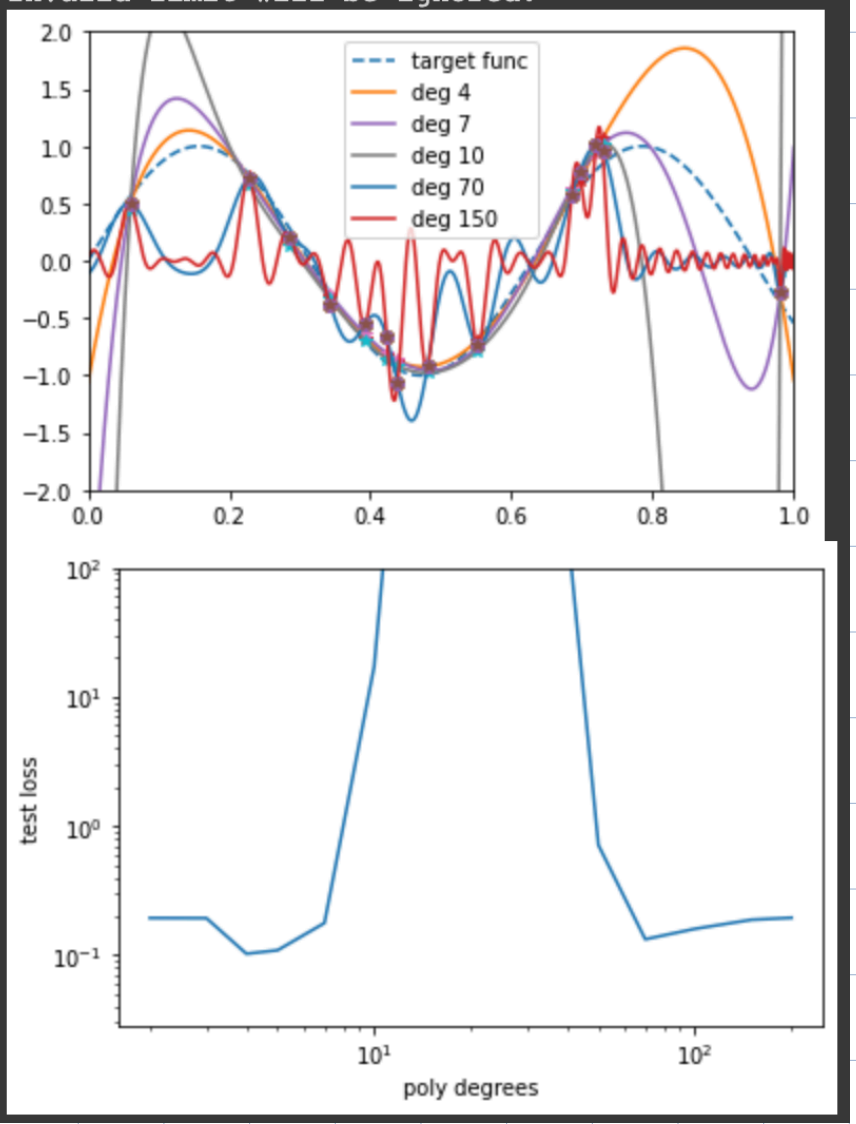
```
def fit_poly(X, d,t):
  X_expand = poly_expand(X, d=d, poly_type = poly_type)
  n = X.shape[0]
  if d > n:
    ## W = ... (Your solution for Part 3.3.2)
    W = X_expand.T.dot(np.linalg.inv(X_expand.dot(X_expand.T))).dot(t)
  else:
    ## W = ... (Your solution for Part 3.2)
    W = np.linalg.inv(X_expand.T.dot(X_expand)).dot(X_expand.T).dot(t)
  return W
```

3.3.4 Continued

Plot the loss values and polynomial degrees:

```
plot_val_loss(poly_degrees, loss_val_list)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel
Invalid limit will be ignored.
```



Overparameterization does NOT always lead to overfitting (larger test error).

As the above plot shows, the test error peaks between polynomial degrees of 10^1 and 10^2.