1.1

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

using Chain Rule : $\frac{dJ}{d\hat{w}} = \frac{dJ}{dy} \cdot \frac{dy}{d\hat{w}}$

$dJ/dy = \frac{d}{dy}\left(\frac{1}{n} y^T y\right) = \frac{2}{n} y$

$dy/d\hat{w} = \frac{d}{d\hat{w}}(X\hat{w} - t) = X$

$$dJ/d\hat{w} = 2/n\, y \cdot X = \frac{2}{n}(X\hat{w} - t) \cdot X = \frac{2}{n} X^T(X\hat{w} - t)$$

Underparameterized Model

1.2.1

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

Gradient decent $\hat{w} \leftarrow \hat{w} - \alpha \frac{\partial J}{d\hat{w}}$ this Converges once $\frac{dJ}{dw} = 0$

set $dJ/d\hat{w} = 0$ to find the minimum from 1.1 $dJ/d\hat{w} = \frac{2}{n} X^T(X\hat{w} - t)$

$$\frac{2}{n} X^T(X\hat{w} - t) = 0 \longrightarrow X^T(X\hat{w} - t) = 0 \longrightarrow X^T X\hat{w} - X^T t = 0 \longrightarrow X^T X\hat{w} = X^T t \longrightarrow \hat{w} = (X^T X)^{-1} X^T t$$

$X^T X$ is invertable since $d > n$

1.2.2

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

from 1.2.1 $\rightarrow$ $\hat{w} = (X^T X)^{-1} X^T t$ sub into Loss $\rightarrow \frac{1}{n}\|X(X^T X)^{-1} X^T t - t\|_2^2$

$t_i = w^{*T} x_i + \epsilon_i$ $\quad t = \Sigma\, w^{*T} x_i + \epsilon_i = Xw^* + \epsilon$ sub this in for t $\rightarrow$ $Loss = \frac{1}{n}\|X(X^T X)^{-1} X^T(Xw^* + \epsilon) - (Xw^* + \epsilon)\|_2^2$

$$Loss = \frac{1}{n}\|\underbrace{X\underbrace{(X^T X)^{-1} X^T X}_{I} w^*}_{XIw^* = Xw^*} + X(X^T X)^{-1} X^T \epsilon - (Xw^* + \epsilon)\|_2^2 = \frac{1}{n}\|\cancel{Xw^*} + X(X^T X)^{-1} X^T \epsilon - \cancel{Xw^*} - \epsilon)\|_2^2$$

$Loss = \frac{1}{n}\|X(X^T X)^{-1} X^T \epsilon - \epsilon\|$ factor out ϵ $\quad Loss = \frac{1}{n}\|(X(X^T X)^{-1} X^T - I)\epsilon\|_2^2$

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

$$\|(X(X^T X)^{-1} X^T - I)\epsilon\|_2^2 = \left((X(X^T X)^{-1} X^T - I)\epsilon\right)^T \left((X(X^T X)^{-1} X^T - I)\epsilon\right) = \epsilon^T (X(X^T X)^{-1} X^T - I)^T (X(X^T X)^{-1} X^T - I)\epsilon$$

Proof that $(X(X^TX)^{-1}X^T - I)$ is a symmetric matrix.

$(X^TX)^T = X^TX$ hence X^TX is symmetric therefore $(X^TX)^{-1}$ is symmetric.

$(X(X^TX)^{-1}X^T)^T = X(X^TX)^{-1}X^T$ hence $X(X^TX)^{-1}X^T$ is symmetric.

The Identity matrix is also symmetric. The difference of two symmetric matrix is also symmetric.

$\longrightarrow (X(X^TX)^{-1}X^T - I)$ is symmetric.

$$\varepsilon^T(X(X^TX)^{-1}X^T - I)^2\varepsilon$$

From matrixcookbook $E[x^TAx] = Tr(A\,Var[x]) + E[x]^T A E[x]$ if A is symmetric

Since $Var[\varepsilon] = \sigma^2$ and $E[\varepsilon] = 0$

$$E[\varepsilon^T(X(X^TX)^{-1}X^T - I)^2\varepsilon] = Tr((X(X^TX)^{-1}X^T - I)^2\sigma^2) = \sigma^2\, Tr((X(X^TX)^{-1}X^T - I)^2)$$

$$(X(X^TX)^{-1}X^T - I)^2 = \underbrace{(X(X^TX)^{-1}X^T)^2} - 2(X(X^TX)^{-1}X^T) + \underbrace{\cancel{I^2}}_{I}$$

$$\longrightarrow (X(X^TX)^{-1}X^T)(X(X^TX)^{-1}X^T) = X\underbrace{(X^TX^{-1})X^TX}_{I}(X^TX)^{-1}X^T = X(X^TX)^{-1}X^T$$

$$(X(X^TX)^{-1}X^T - I)^2 = X(X^TX)^{-1}X^T - 2X(X^TX)^{-1}X^T + I = I - X(X^TX)^{-1}X^T$$

$$E[\varepsilon^T(X(X^TX)^{-1})X^T - I)^2\varepsilon] = \sigma^2\, Tr(I - X(X^TX)^{-1}X^T) = \sigma^2\, Tr(I) - Tr(X(X^TX)^{-1}X^T)$$

using cyclic property of trace $\longrightarrow$ $Tr(X(X^TX)^{-1}X^T) = Tr((X^TX)^{-1}X^TX) = Tr(I)$ Since X is $n \times d$ $(X^TX)^{-1}X^TX$ is $d \times d$ $\Big\}$ $Tr(X(X^TX)^{-1}X^T) = d$

Dimension of I is $n \times n$ Since $X(X^TX)^{-1}X^T$ is an $n \times n$ matrix $\longrightarrow Tr(I) = n$

Hence

$$E[\varepsilon^T(X(X^TX)^{-1})X^T - I)^2\varepsilon] = \sigma^2(n-d)$$

$Loss = \frac{1}{n}\varepsilon^T(X(X^TX)^{-1})X^T - I)^2\varepsilon \longrightarrow E[Loss] = \sigma^2(1 - \frac{d}{n})$

Overparameterized Model

1.3.1

$\hat{W} = [w_1, w_2]^T \qquad x_1 = [1, 1] \qquad t = 3$

$\hat{W} x_1 = t \qquad \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} [1, 1] = 3 \qquad w_1 + w_2 = 3 \longleftarrow$ equation of the line

$W = \begin{bmatrix} w_1 \\ 3 - w_2 \end{bmatrix}$ this shows that w_1 and w_2 can be any value

1.3.2

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

$\text{Loss} = \frac{1}{n} \| X \hat{w} - t \|_2^2 \qquad$ gradient decent : $\hat{W} \leftarrow \hat{W} - \alpha \frac{\partial \text{loss}}{d \hat{w}}$

As calculate in 1.1 $\quad \frac{d\text{loss}}{d\hat{w}} = \frac{2}{n} X^T (X\hat{w} - t)$

$$\hat{W}' \longleftarrow \hat{W}(0) - \alpha \frac{2}{n} X^T (X \hat{w} - t)$$

The gradient decent converges at a point where $X^T (X \hat{w} - t) = 0$

$$X^T X W - X^T t \rightarrow X^T X W = X^T t$$

Given that we start from $w(0) = 0$ which is in the row space of X

and

$\alpha \frac{2}{n} X^T (X \hat{w} - t)$ is also in the row space of X (due to X^T multiplication)

Any resultant w from subtracting these tw when updating the $\hat{w}$ is also in the row space of X. There any $\hat{w}$ can be represented via $\hat{w} = X^T \alpha$

$$X^T X w = X^T t \xrightarrow{w = X^T \alpha} X^T X X^T \alpha = X^T t \xrightarrow{X X^T \text{ is invertable when } d > n} X^T \alpha = X^T (X X^T)^{-1} t$$

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

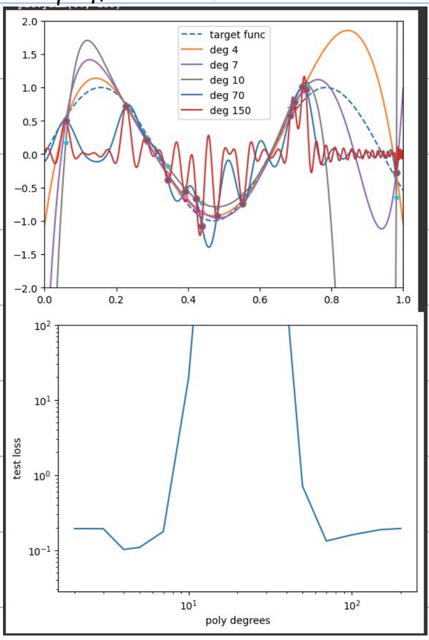
1.3.4

Code snippet :

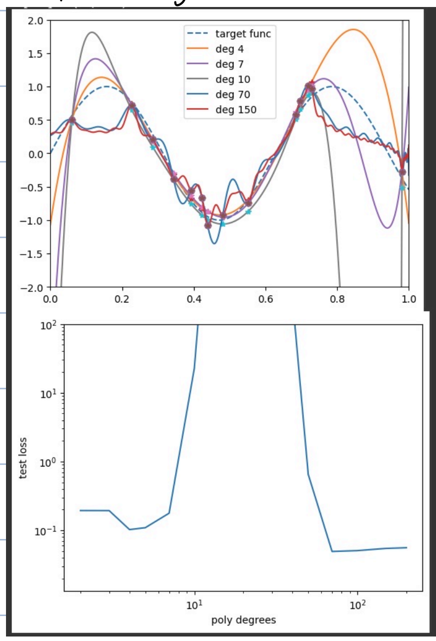
```
[6] # to be implemented; fill in the derived solution for the underparameterized (d<n) and overparameterized (d>n) problem

    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 = X_expand.T @ np.linalg.inv(X_expand @ X_expand.T) @ t
        ## W = ... (Your solution for Part 1.3.2)
      else:
        W = np.linalg.inv(X_expand.T @ X_expand) @ X_expand.T @ t
        ## W = ... (Your solution for Part 1.2.1)
      return W
```

Poly type = Chebyshev



Poly type = legendre



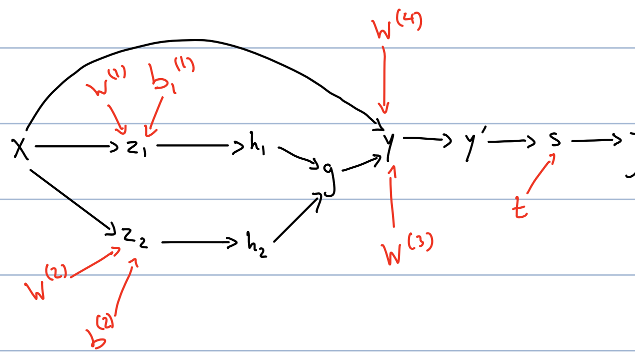
Actual n = 14

It can be seen that overparamaterization does not always lead to over fitting. From the test error vs poly degree it can be seen that as the poly degree gets closer to the target n of 14 it the test error increases. However, around a poly degree of 70 the test error drops. Showing that increase in poly degree does not always cause overfitting.

Back propagation

Automatic Differentiation

2.1.1



2.1.2

- $\bar{\mathcal{J}} = -1$
- $\bar{S} = \frac{\partial \mathcal{J}}{\partial s}^T \bar{\mathcal{J}} = -S$
- $\bar{y'} = \frac{ds}{dy'}^T \bar{S}$

Since $\mathbb{I}(t=k) = 1$ only when $k=t$

and $\frac{d\log(y'_k)}{dy'} = 1/\log(y'_k)$

$\bar{y} = [0 \dots, 1/y'_t \dots 0]^T \bar{S}$

- $\bar{y} = \frac{dy'}{dy}^T \bar{y'} = (\mathrm{softmax}'(y))^T \bar{y'}$
- $\bar{g} = \frac{dy}{dg}^T \bar{y} = W^{(3)T} \bar{y}$

- $\bar{h}_2 = \frac{\partial g^T}{\partial h_2}\bar{g} = \frac{\partial g_i^T}{\partial h_{2i}} = \begin{bmatrix} h_{1,1} & 0 & 0 & \cdots \\ 0 & h_{1,2} & 0 & \cdots \\ \vdots & & \ddots & \\ 0 & 0 & & h_{1,N} \end{bmatrix}$ when multiplied by $\bar{g} = h_1 \circ \bar{g}$ $\quad \bar{h}_2 = h_1 \circ \bar{g}$

- $\bar{h}_1 = \frac{\partial g^T}{\partial h_1}\bar{g} = h_2 \circ \bar{g}$ Same reasoning as above

- $\bar{z}_2 = \frac{\partial h_2^T}{\partial z_2}\bar{h}_2 = \sigma'(z_2) \circ \bar{h}_2$

$\sigma'(z_2) = d(1+e^{-z})^{-1}/dz = -(1+e^{-z})^{-2} \times \frac{d}{dz}(\cancel{1}+e^{-z})$

$-e^{-z}$

$\bar{z}_2 = \left(-(1+e^{-z})^{2} \cdot e^{-z}\right) \circ \bar{h}_2$

- $\bar{z}_1 = \frac{dh_1^T}{dz_1}\bar{h}_1 \quad \frac{\mathrm{ReLU}(z_1)^T}{dz_1} = \mathrm{ReLU}'(z_1) \circ \bar{h}_1 \quad \mathrm{ReLU}'(z_1) = \begin{cases} 1 & \text{if } z_1 > 0 \\ 0 & \text{if } z_1 < 0 \end{cases}$

- $\bar{x} = \frac{\partial z_1^T}{\partial x}\bar{z}_1 + \frac{\partial z_2^T}{\partial x}\bar{z}_2 + \frac{\partial y^T}{\partial x}\bar{y} = W^{(1)T}\bar{z}_1 + W^{(2)T}\bar{z}_2 + W^{(3)T} t$

Gradient Norm Computation

2.2.1

$$\begin{tikzcd} & W^{(1)} \arrow[d] & & \\ x \arrow[r] & z \arrow[r] & h \arrow[r] & y \\ & & W^{(2)} \arrow[ur] & \end{tikzcd}$$

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

- $\bar{z} = \frac{dh^T}{dz}\bar{h} = \mathrm{Relu}'(z) \circ \bar{h} \rightarrow h = \mathrm{Relu}(z) = \begin{bmatrix} 8 \\ 1 \\ 0 \end{bmatrix} \quad \mathrm{Relu}'(z) = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \rightarrow \bar{z} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}\begin{bmatrix} -4 \\ 6 \\ 4 \end{bmatrix} = \begin{bmatrix} -4 \\ 6 \\ 0 \end{bmatrix}$

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

- $\frac{\partial J}{\partial W^{(1)}} = \bar{z}\frac{dz^T}{dW^{(1)}} = \bar{z}x^T = \begin{bmatrix} -4 \\ 6 \\ 0 \end{bmatrix}\begin{bmatrix} 1 & 3 & 1 \end{bmatrix} = \begin{bmatrix} -4 & -12 & -4 \\ 6 & 18 & 6 \\ 0 & 0 & 0 \end{bmatrix}$
- $\left\| \frac{\partial J}{\partial W^{(1)}} \right\|_F^2 = \sum_{i=1}^{m}\sum_{j=1}^{n} |a_{ij}|^2 = 4^2+12^2+4^2+6^2+18^2+6^2 = 572$

- $\frac{\partial J}{\partial W^{(2)}} = \bar{y}\frac{dy^T}{dW^{(2)}} = \bar{y}h^T = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}\begin{bmatrix} 8 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 8 & 1 & 0 \\ 8 & 1 & 0 \\ 8 & 1 & 0 \end{bmatrix}$
- $\left\| \frac{\partial J}{\partial W^{(2)}} \right\|_F^2 = 8^2+8^2+8^2+1^2+1^2+1^2 = 195$

2.2.2

$$\left\| \frac{\partial J}{dw^{(1)}} \right\|_F^2 = \|x\|_2^2 \, \|\bar{z}\|_2^2 = (1^2 + 3^2 + 1^2) \times ((-4)^2 + 6^2 + 0^2) = 11 \times 52 = 572$$

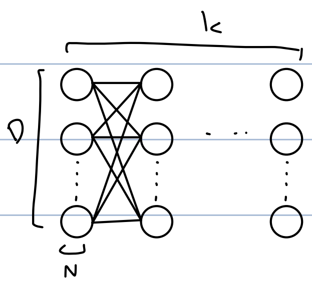
Same as the answer in the prev. question

$$\left\| \frac{\partial J}{dw^{(2)}} \right\|_F^2 = \|h\|_2^2 \, \|\bar{y}\|_2^2 = (8^2 + 1^2 + 0^2) \times (1^2 + 1^2 + 1^2) = 65 \times 3 = 195$$

Same as the answer in the prev. question

2.2.3

	T(Naive)	T(Efficient)	M (Naive)	M (Efficient)
Forward Pass	NkD^2	NkD^2	$O(kD^2 + kND)$	$O(kD^2 + kND)$
Backward Pass	$2NkD^2$	NkD^2	$O(kND^2)$	$O(kD^2 + NkD)$
Gradient Norm Computation	NkD^2	$(2D+1)Nk$	$O(kND^2)$	$O(kND)$



$W \in R^{D \times D}$

Forward pass:

T(Naive) : k layer with $D \times D$ matrix each D nuerons and N training Sample NkD^2

T(Efficient): same as Naive

M(Naive): For the k, $D \times D$ matrecies kD^2, Also need to save value of nuerons (Values needed for back propagation)
So k layer with N training Sample and D Nuerons kND. $\rightarrow$ $O(kD^2 + kND)$

M(Efficient): Same as Naive

Backward Pass:

T(Naive) : Error vector Computation $D \times D$ matrix multiplied by D dimension error vector. (For N training Sample and k layers)
Gradient $D \times D$ gradient matrix and D nuerons for k layers and N sample. $\rightarrow$ $2NkD^2$

T(Efficient): Error vector Computation same as Naive. For Gradient we don't store all of the $D \times D$ gradient matrcies just add them after each step. $\rightarrow$ NkD^2

M(Naive): We need to store Error vector $[O(NkD)]$, parameters/weights $[O(kD^2)]$, and gradient $[O(kND^2)]$
$O(kND^2)$ dominates $\rightarrow$ $O(kND^2)$

M(Efficient): Sames as Naive but gradient is just $O(kD^2)$ as we are updating the sum each time.

$\rightarrow$ $O(kD^2 + NkD)$

Gradient Norm Computation:

T(Naive): Square all the gradient D^2 scaler computation for k layers and N training sample $\rightarrow NkD^2$

T(Efficient): $\|x\|_2^2 \|\bar{y}\|_2^2$ $\rightarrow$ Norm of a D dimension vector has D scalar computation. This is done twice.

Then they are multiplied by each other. $(2D+1)$ scalar computation for k layers and N samples $\rightarrow (2D+1)Nk$

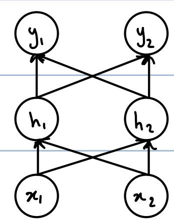
M(Naive): Storing $D \times D$ matracies for k layer and N training sample. $\rightarrow O(kND^2)$

M(Efficient): Stores D dimension vector for k layer and N samples $\rightarrow O(kND)$

Hard Coding Networks

Sort two numbers

3.1 $\max(x_1, x_2) = \frac{1}{2}(x_1 + x_2) + \frac{1}{2}|x_1 - x_2|$ $\quad \min(x_1, x_2) = \frac{1}{2}(x_1 + x_2) - \frac{1}{2}|x_1 - x_2|$



$$W^{(2)} W^{(1)} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{2}(x_1+x_2) - \frac{1}{2}|x_1 - x_2| \\ \frac{1}{2}(x_1+x_2) + \frac{1}{2}|x_1 + x_2| \end{bmatrix}$$

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} ex_1 + fx_2 \\ gx_1 + hx_2 \end{bmatrix} = \begin{bmatrix} a(ex_1 + fx_2) + b(gx_1 + hx_2) \\ c(ex_1 + fx_2) + d(gx_1 + hx_2) \end{bmatrix}$$

$$\begin{bmatrix} a(ex_1 + fx_2) + b(gx_1 + hx_2) \\ c(ex_1 + fx_2) + d(gx_1 + hx_2) \end{bmatrix} = \begin{bmatrix} \frac{1}{2}(x_1+x_2) - \frac{1}{2}|x_1 - x_2| \\ \frac{1}{2}(x_1+x_2) + \frac{1}{2}|x_1 + x_2| \end{bmatrix} \rightarrow$$

$a = \frac{1}{2}$ $\quad b = -\frac{1}{2}$ $\quad c = \frac{1}{2}$ $\quad d = \frac{1}{2}$

$e = 1$ $\quad f = 1$ $\quad g = 1$ $\quad h = -1$

$\phi^{(2)}(z) = z$

$\phi^{(1)}(z) = |z|$

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

$$b^{(1)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad b^{(2)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\phi^{(1)}(z) = |z| \qquad \phi^{(2)}(z) = z$$

Perform Sort

3.2 Merge sort