CSC413H1 - Assignment 1

1.2.1: Training converges when $\frac{1}{n}||X\hat{\mathbf{w}}-\mathbf{t}||_2^2$ achieves its minimum, or equivalently when $\nabla_{\hat{\mathbf{w}}}\frac{1}{n}||X\hat{\mathbf{w}}-\mathbf{t}||_2^2 = \frac{2}{n}X^T(X\hat{\mathbf{w}}-\mathbf{t}) = 0$. To solve for $\hat{\mathbf{w}}$ in this scenario, note that

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

Since $d < n, X^TX$ is invertible, we can multiply $(X^TX)^{-1}$ to both sides to yield

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

1.2.2: Substituting the previous answer and $\mathbf{t} = X\mathbf{w}^* + \boldsymbol{\epsilon}$ into the loss yields

$$\begin{aligned}
\frac{1}{n}||X\hat{\mathbf{w}}-\mathbf{t}||_2^2 = \frac{1}{n}||X(X^TX)^{-1}X^T\mathbf{t}-\mathbf{t}||_2^2 &= \frac{1}{n}||X(X^TX)^{-1}X^T(X\hat{\mathbf{w}}+\boldsymbol{\epsilon}) - (X\hat{\mathbf{w}}+\boldsymbol{\epsilon})||_2^2 \\
&= \frac{1}{n}||X(X^TX)^{-1}X^TX\hat{\mathbf{w}} + X(X^TX)^{-1}X^T\boldsymbol{\epsilon} - X\hat{\mathbf{w}} - \boldsymbol{\epsilon}||_2^2 \\
&= \frac{1}{n}||X\hat{\mathbf{w}} - X\hat{\mathbf{w}} + (X(X^TX)^{-1}X^T - I)\boldsymbol{\epsilon}||_2^2 \\
&= \frac{1}{n}||(X(X^TX)^{-1}X^T - I)\boldsymbol{\epsilon}||_2^2
\end{aligned}$$

as desired. Notice that this equals

$$\begin{aligned}
&\frac{1}{n}(X(X^TX)^{-1}X^T\boldsymbol{\epsilon}-\boldsymbol{\epsilon})^T(X(X^TX)^{-1}X^T\boldsymbol{\epsilon}-\boldsymbol{\epsilon}) \\
&= \frac{1}{n}(\boldsymbol{\epsilon}^TX(X^TX)^{-1}X^T\boldsymbol{\epsilon} - \boldsymbol{\epsilon}^TX(X^TX)^{-1}X^T\boldsymbol{\epsilon} - \boldsymbol{\epsilon}^TX(X^TX)^{-1}X^T\boldsymbol{\epsilon} + \boldsymbol{\epsilon}^T\boldsymbol{\epsilon}) = \boldsymbol{\epsilon}^T\boldsymbol{\epsilon} - \boldsymbol{\epsilon}^TX(X^TX)^{-1}X^T\boldsymbol{\epsilon}
\end{aligned}$$

since $((X^TX)^{-1})^T = ((X^TX)^T)^{-1} = (X^TX)^{-1}$. Then, the expectation of this error is

$$\begin{aligned}
\frac{1}{n}\mathbb{E}\Big[\boldsymbol{\epsilon}^T\boldsymbol{\epsilon} - \boldsymbol{\epsilon}^TX(X^TX)^{-1}X^T\boldsymbol{\epsilon}\Big] &= \frac{1}{n}\mathbb{E}\Big[\mathrm{tr}(\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T - \boldsymbol{\epsilon}^TX(X^TX)^{-1}X^T\boldsymbol{\epsilon})\Big] \text{ since the inside is a scalar} \\
&= \frac{1}{n}\mathbb{E}\Big[\mathrm{tr}(\boldsymbol{\epsilon}^T\boldsymbol{\epsilon})\Big] - \frac{1}{n}\mathbb{E}\Big[\mathrm{tr}(\boldsymbol{\epsilon}^TX(X^TX)^{-1}X^T\boldsymbol{\epsilon}\Big] \text{ by linearity of trace} \\
&= \frac{1}{n}\mathrm{tr}\Big[\mathbb{E}(\boldsymbol{\epsilon}^T\boldsymbol{\epsilon})\Big] - \frac{1}{n}\mathrm{tr}\Big[\mathbb{E}(\boldsymbol{\epsilon}^TX(X^TX)^{-1}X^T\boldsymbol{\epsilon})\Big] \text{ since } \mathbb{E}[\mathrm{tr}(A)] = \mathrm{tr}[\mathbb{E}(A)] \\
&= \frac{1}{n}\mathrm{tr}\Big[\mathbb{E}(\Sigma_{i=1}^n\boldsymbol{\epsilon}_i^2)\Big] - \frac{1}{n}\mathrm{tr}\Big[\mathbb{E}(\boldsymbol{\epsilon}\boldsymbol{\epsilon}^TX(X^TX)^{-1}X^T)\Big] \\
&= \frac{1}{n}\mathrm{tr}\Big[\Sigma_{i=1}^n\mathbb{E}(\boldsymbol{\epsilon}_i^2)\Big] - \frac{1}{n}\mathrm{tr}\Big[\mathbb{E}(\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T)X(X^TX)^{-1}X^T\Big] \text{ by independence} \\
&= \frac{1}{n}\mathrm{tr}(n\sigma^2) - \frac{1}{n}\mathrm{tr}\Big[\sigma^2X(X^TX)^{-1}X^T\Big] \\
&= \sigma^2 - \frac{\sigma^2}{n}\mathrm{tr}\Big[(X^TX)^{-1}X^TX\Big] \\
&= \sigma^2 - \frac{\sigma^2 d}{n}
\end{aligned}$$

where $\mathbb{E}(\boldsymbol{\epsilon}_i^2) = \mathrm{Var}(\boldsymbol{\epsilon}_i) + \mathbb{E}(\boldsymbol{\epsilon}_i)^2 = \sigma^2$ and $\mathbb{E}(\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T) = \begin{bmatrix} \mathbb{E}(\boldsymbol{\epsilon}_1^2) & \dots & \mathbb{E}(\boldsymbol{\epsilon}_1)\mathbb{E}(\boldsymbol{\epsilon}_n) \\ \vdots & \ddots & \vdots \\ \mathbb{E}(\boldsymbol{\epsilon}_n)\mathbb{E}(\boldsymbol{\epsilon}_1) & \dots & \mathbb{E}(\boldsymbol{\epsilon}_n^2) \end{bmatrix} = \sigma^2 I_n.$

1.3.1: $\hat{\mathbf{w}}^T\mathbf{x_1} = \begin{bmatrix}\hat{w}_1 & \hat{w}_2\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix} = \hat{w}_1 + \hat{w}_2 = t_1 = 3$, so the equation of the line is $\hat{w}_2 = -\hat{w}_1 + 3$. Thus, there are infinitely many $\hat{\mathbf{w}}$ satisfying $\hat{\mathbf{w}}^T\mathbf{x_1} = t_1$.

1.3.2: Assuming that the gradient is spanned by the rows of X, we substitute $\hat{\mathbf{w}} = X^T\mathbf{a}$ for $\mathbf{a} \in \mathbb{R}^n$ into the objective from 1.2.1 to yield

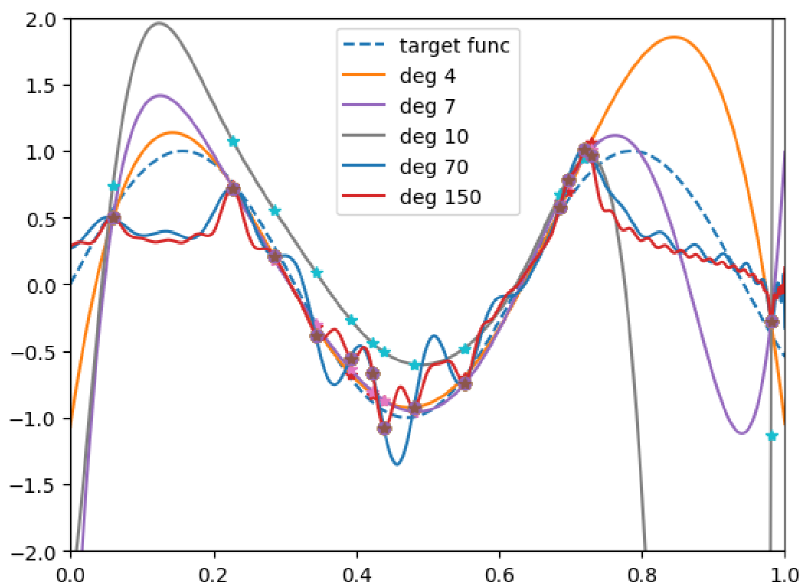
$$\begin{aligned} X^T(X\hat{\mathbf{w}} - \mathbf{t}) = 0 &\iff (XX^T)^{-1}XX^T(X\hat{\mathbf{w}} - \mathbf{t}) = 0 \iff X\hat{\mathbf{w}} - \mathbf{t} = 0 \iff XX^T\mathbf{a} - \mathbf{t} = 0 \\ &\iff XX^T\mathbf{a} = \mathbf{t} \iff \mathbf{a} = (XX^T)^{-1}\mathbf{t} \end{aligned}$$

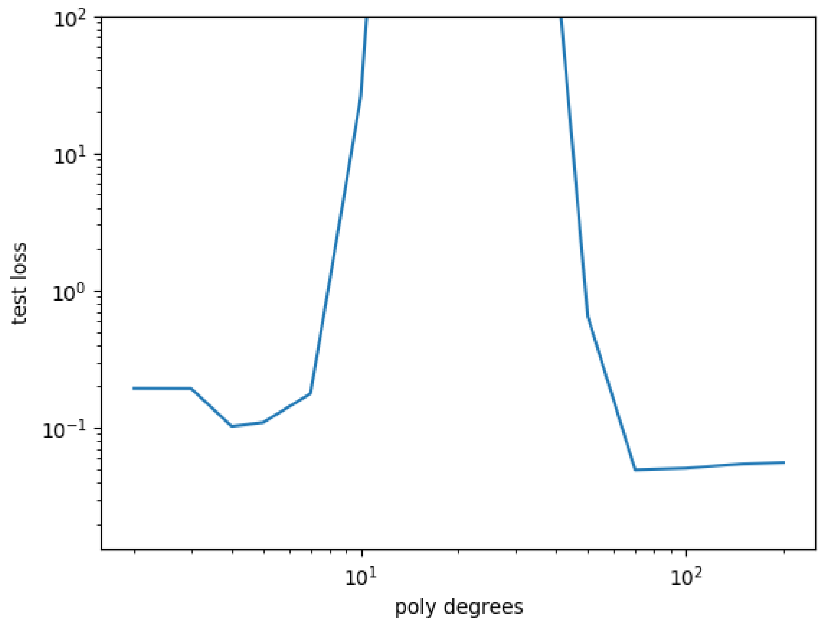
where the first and last equivalences hold since $d > n$ and XX^T is invertible. Thus, the unique minimizer is

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

1.3.4: The code for `fit_poly`, the plot of the polynomials, and the plot of the test losses vs. degree of polynomial are as follows:

```
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).dot(t)
    else:
        W = np.linalg.inv(X_expand.T @ X_expand) @ X_expand.T.dot(t)
    return W
```





Overparameterization does not always lead to overfitting. As shown in the last plot, the test loss decreases when the degree of polynomial is around 10^2, indicating that a higher degree polynomial does not always have a higher test error.

2.1.2: Suppose that $\mathbf{x}$ has dimensions $D \times 1$ and $\mathbf{z}_1, \mathbf{h}_1, \mathbf{z}_2, \mathbf{h}_2, \mathbf{g}$ have dimensions $K \times 1$. Define $\mathbf{h}_{1i}$ as the

i-th component of $\mathbf{h}_1$ and similarly for $\mathbf{z}_{1i}, \mathbf{h}_{2i}, \mathbf{z}_{2i}$.

$$\begin{aligned}
\overline{\mathcal{J}} &= 1 \\
\overline{\mathcal{S}} &= -\overline{\mathcal{J}} \\
\overline{\mathbf{y}}' &= \begin{bmatrix} \frac{\partial S}{\partial \mathbf{y}'_1} & \dots & \frac{\partial S}{\partial \mathbf{y}'_N} \end{bmatrix}^T \overline{\mathcal{S}} = \begin{bmatrix} 0 & \dots & \frac{1}{\mathbf{y}'_t} & \dots & 0 \end{bmatrix}^T \overline{\mathcal{S}} \\
\overline{\mathbf{y}} &= \begin{bmatrix} \frac{\partial \mathbf{y}'_1}{\partial \mathbf{y}_1} & \dots & \frac{\partial \mathbf{y}'_1}{\partial \mathbf{y}_N} \\ \vdots & \ddots & \vdots \\ \frac{\partial \mathbf{y}'_N}{\partial \mathbf{y}_1} & \dots & \frac{\partial \mathbf{y}'_N}{\partial \mathbf{y}_N} \end{bmatrix}^T \overline{\mathbf{y}}' = \mathrm{softmax}'(\mathbf{y})^T \overline{\mathbf{y}}' \\
\overline{\mathbf{g}} &= \begin{bmatrix} \frac{\partial \mathbf{y}_1}{\partial \mathbf{g}_1} & \dots & \frac{\partial \mathbf{y}_1}{\partial \mathbf{g}_K} \\ \vdots & \ddots & \vdots \\ \frac{\partial \mathbf{y}_N}{\partial \mathbf{g}_1} & \dots & \frac{\partial \mathbf{y}_N}{\partial \mathbf{g}_K} \end{bmatrix}^T \overline{\mathbf{y}} = \mathbf{W}^{(3)T} \overline{\mathbf{y}} \\
\overline{\mathbf{h}_1} &= \begin{bmatrix} \mathbf{h}_{21} \overline{\mathbf{g}_1} \\ \vdots \\ \mathbf{h}_{2K} \overline{\mathbf{g}_K} \end{bmatrix} = \mathbf{h}_2 \circ \overline{\mathbf{g}} \\
\overline{\mathbf{h}_2} &= \begin{bmatrix} \mathbf{h}_{11} \overline{\mathbf{g}_1} \\ \vdots \\ \mathbf{h}_{1K} \overline{\mathbf{g}_K} \end{bmatrix} = \mathbf{h}_1 \circ \overline{\mathbf{g}} \\
\overline{\mathbf{z}_1} &= \begin{bmatrix} \mathbb{I}(\mathbf{z}_{11} > 0) \\ \vdots \\ \mathbb{I}(\mathbf{z}_{1K} > 0) \end{bmatrix} \circ \overline{\mathbf{h}_1} \text{ where } \mathbb{I} \text{ is the indicator function} \\
\overline{\mathbf{z}_2} &= \begin{bmatrix} \sigma(\mathbf{z}_{21}) \\ \vdots \\ \sigma(\mathbf{z}_{2K}) \end{bmatrix} \circ \begin{bmatrix} 1 - \sigma(\mathbf{z}_{21}) \\ \vdots \\ 1 - \sigma(\mathbf{z}_{2K}) \end{bmatrix} \circ \overline{\mathbf{h}_2} \\
\overline{\mathbf{x}} &= \begin{bmatrix} \frac{\partial \mathbf{y}_1}{\partial \mathbf{x}_1} & \dots & \frac{\partial \mathbf{y}_1}{\partial \mathbf{x}_D} \\ \vdots & \ddots & \vdots \\ \frac{\partial \mathbf{y}_N}{\partial \mathbf{x}_1} & \dots & \frac{\partial \mathbf{y}_N}{\partial \mathbf{x}_D} \end{bmatrix}^T \overline{\mathbf{y}} + \begin{bmatrix} \frac{\partial \mathbf{z}_{11}}{\partial \mathbf{x}_1} & \dots & \frac{\partial \mathbf{z}_{11}}{\partial \mathbf{x}_D} \\ \vdots & \ddots & \vdots \\ \frac{\partial \mathbf{z}_{1K}}{\partial \mathbf{x}_1} & \dots & \frac{\partial \mathbf{z}_{1K}}{\partial \mathbf{x}_D} \end{bmatrix}^T \overline{\mathbf{z}_1} + \begin{bmatrix} \frac{\partial \mathbf{z}_{21}}{\partial \mathbf{x}_1} & \dots & \frac{\partial \mathbf{z}_{21}}{\partial \mathbf{x}_D} \\ \vdots & \ddots & \vdots \\ \frac{\partial \mathbf{z}_{2K}}{\partial \mathbf{x}_1} & \dots & \frac{\partial \mathbf{z}_{2K}}{\partial \mathbf{x}_D} \end{bmatrix}^T \overline{\mathbf{z}_2} \\
&= \mathbf{W}^{(4)T} \overline{\mathbf{y}} + \mathbf{W}^{(1)T} \overline{\mathbf{z}_1} + \mathbf{W}^{(2)T} \overline{\mathbf{z}_2}
\end{aligned}$$

2.2.1:

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

$$\mathbf{h} = \mathrm{ReLU}(\mathbf{z}) = \begin{bmatrix} 8 \\ 1 \\ 0 \end{bmatrix}$$

$$\overline{\mathbf{h}} = \begin{bmatrix} \frac{\partial \mathbf{y}_1}{\partial \mathbf{h}_1} & \cdots & \frac{\partial \mathbf{y}_1}{\partial \mathbf{h}_3} \\ \vdots & \ddots & \vdots \\ \frac{\partial \mathbf{y}_3}{\partial \mathbf{h}_1} & \cdots & \frac{\partial \mathbf{y}_3}{\partial \mathbf{h}_3} \end{bmatrix} \overline{\mathbf{y}} = \mathbf{W}^{(2)T}\overline{\mathbf{y}} = \begin{bmatrix} -2 & 1 & -3 \\ 4 & -2 & 4 \\ 1 & -3 & 6 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -4 \\ 6 \\ 4 \end{bmatrix}$$

$$\overline{\mathbf{z}} = \begin{bmatrix} \mathbb{I}(\mathbf{z}_1 > 0) \\ \mathbb{I}(\mathbf{z}_2 > 0) \\ \mathbb{I}(\mathbf{z}_3 > 0) \end{bmatrix} \circ \overline{\mathbf{h}} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \circ \begin{bmatrix} -4 \\ 6 \\ 4 \end{bmatrix} = \begin{bmatrix} -4 \\ 6 \\ 0 \end{bmatrix}$$

$$\overline{\mathbf{W}^{(1)}} = (\overline{\mathbf{z}}\mathbf{x}^T)^T = (\begin{bmatrix} -4 \\ 6 \\ 0 \end{bmatrix} \begin{bmatrix} 1 & 3 & 1 \end{bmatrix})^T = \begin{bmatrix} -4 & 6 & 0 \\ -12 & 18 & 0 \\ -4 & 6 & 0 \end{bmatrix}$$

$$\overline{\mathbf{W}^{(2)}} = (\overline{\mathbf{y}}\mathbf{h}^T)^T = (\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} 8 & 1 & 0 \end{bmatrix})^T = \begin{bmatrix} 8 & 8 & 8 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

$$||\overline{\mathbf{W}^{(1)}}||_F^2 = \mathrm{trace}(\begin{bmatrix} -4 & -12 & -4 \\ 6 & 18 & 6 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} -4 & 6 & 0 \\ -12 & 18 & 0 \\ -4 & 6 & 0 \end{bmatrix}) = 572$$

$$||\overline{\mathbf{W}^{(2)}}||_F^2 = \mathrm{trace}(\begin{bmatrix} 8 & 1 & 0 \\ 8 & 1 & 0 \\ 8 & 1 & 0 \end{bmatrix} \begin{bmatrix} 8 & 8 & 8 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}) = 195$$

2.2.2:

$$||\overline{\mathbf{W}^{(1)}}||_F^2 = ||\mathbf{x}||_2^2 ||\overline{\mathbf{z}}||_2^2 = (1^2 + 3^2 + 1^2)((-4)^2 + 6^2) = (11)(52) = 572$$
$$||\overline{\mathbf{W}^{(2)}}||_F^2 = ||\mathbf{h}||_2^2 ||\overline{\mathbf{y}}||_2^2 = (8^2 + 1^2)(1^2 + 1^2 + 1^2) = (37)(3) = 195$$

2.2.3:

	T (Naive)	T (Efficient)	M (Naive)	M (Efficient)
Forward pass	NKD^2	NKD^2	$O(KD^2 + NKD)$	$O(KD^2 + NKD)$
Backward pass	$2NKD^2$	NKD^2	$O(NKD^2)$	$O(KD^2 + NKD)$
Gradient norm	NKD^2	$NK(2D+1)$	$O(NKD^2)$	$O(NKD)$

- Forward pass:
  - For both the naive and efficient methods, there are D^2 scalar multiplications per weight matrix and input vector, and there are K matrices and N vectors, resulting in NKD^2 overall.
  - Both methods need to store these K matrices with D^2 parameters each ($O(KD^2)$) and NK vectors with D parameters each ($O(NKD)$).
- Backward pass:
  - Similar to the forward pass, both methods require D^2 scalar multiplications for each of K matrices and N vectors to calculate the error vectors. However, the naive method also needs

to compute NK Jacobians of the weights, each of which is an outer product with D^2 scalar multiplications.

 - The naive method needs to store the activations ($O(NKD)$), error vectors ($O(NKD)$), weights ($O(KD^2)$), and the Jacobians of the weights ($O(NKD^2)$), resulting in $O(NKD^2 + KD^2 + 2NKD)$ or $O(NKD^2)$ overall. However, the efficient method does not need to store the Jacobians, resulting in $O(KD^2 + 2NKD)$ or $O(KD^2 + NKD)$ overall.

- Gradient norm:
 - The naive method requires D^2 scalar multiplications to calculate the norm for each of NK Jacobians, while the efficient method only requires $2D$ multiplications to calculate the norm, plus an additional multiplication at the end, for each of NK Jacobians.
 - The naive method needs to store NK Jacobians with D^2 parameters each ($O(NKD^2)$), while the efficient methods only needs to store NK vectors with D parameters each ($O(NKD)$).

3.1: $\mathbf{W}^{(1)} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$, $\mathbf{W}^{(2)} = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix}$, $\mathbf{b}^{(1)} = \mathbf{b}^{(2)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$, $\varphi^{(1)}(z) = |z|$, $\varphi^{(2)}(z) = z$.

3.2: Using merge sort: