Vincent Fan

## Q1 Optimization

### 1.1.1 Minimum Norm Solution

We know that $x_i$ is contained in the span of $X$ since it is a row vector of $X$.

We start by calculating the gradient of loss:

$$\nabla_{w_t} L_i(x_i, w_t) = \frac{d}{dw_t}(\|x_i w_t - t_i\|_2^2) = 2x_i^{\mathrm{T}}(x_i w_t - t_i)$$

So, note that each weight update does not leave the span of $X$.

Then we can set it to zero to get the unique solution:

$$2x_i^{\mathrm{T}}(x_i w_t - t_i) = 0$$

$$2x_i^{\mathrm{T}} x_i w_t = 2x_i^{\mathrm{T}} t_i$$

$$w_t = \frac{x_i^{\mathrm{T}} t_i}{x_i^{\mathrm{T}} x_i}$$

Note that since $x_i$ is a single datum, $x_i^{\mathrm{T}} x_i$ gives a scaler and $t_i$ is also a scaler.

We can let $a = \frac{t_i}{x_i^{\mathrm{T}} x_i}$, thus $w_t = x_i^{\mathrm{T}} a$.

Therefore, we can say that each weight update can be expressed in a linear combination of rows of $X$ since the gradient is always spanned by the rows of $X$. So, we can let our final zero loss weight to be: $\widehat{w} = X^{\mathrm{T}} a$ for some $a \in \mathbb{R}^n$

$$X\widehat{w} - t = XX^{\mathrm{T}}\mathrm{a} - \mathrm{t} = 0$$

$$a = \left(XX^{\mathrm{T}}\right)^{-1} t$$

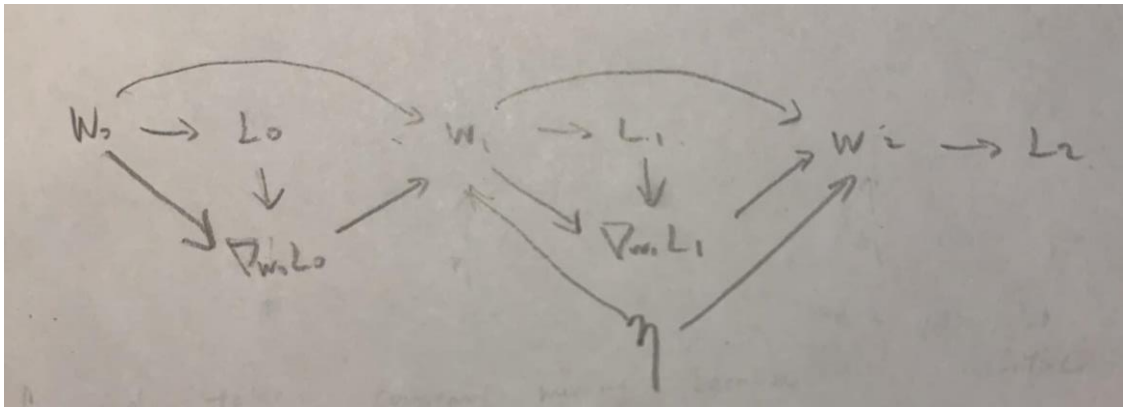
$$\widehat{w} = X^{\mathrm{T}}\left(XX^{\mathrm{T}}\right)^{-1} t$$

Which is identical to the minimum norm solution obtained by gradient descent from HW1.

## Q2 Gradient-based Hyper-parameter Optimization

### 2.1 Computation Graph of Learning Rates

2.1.1



2.1.2

The forward pass takes $O(1)$ space. On the other hand, the memory complexity for back propagation is $O(t)$ because from the graph, we can see that the learning rate is going into every weight update. Thus, it needs to store all variables for $t$ iterations and so linear in $t$.

2.1.3

The potential problem is that it will likely run out of memory before able to perform back propagation, this is because back propagation only happens when we finish forward pass. However, since we need to store the entire computation graph in memory to reference every weight update. Therefore, we will run out of memory before reaching convergence.

**2.2 Learning Learning Rates**

2.2.1

$$L_0 = \frac{1}{n}\|Xw_0 - t\|_2^2$$

$$\frac{d}{dw_0}L_0 = \frac{2}{n}X^{\mathrm{T}}(Xw_0 - t)$$

$$w_1 = w_0 - \eta\frac{2}{n}X^{\mathrm{T}}(Xw_0 - t) = w_0 - \frac{2\eta}{n}X^{\mathrm{T}}a, \quad a = Xw_0 - t$$

$$L_1 = \frac{1}{n}\|Xw_1 - t\|_2^2 = \frac{1}{n}\left\|Xw_0 - \frac{2\eta}{n}XX^{\mathrm{T}}a - t\right\|_2^2$$

2.2.2

$$\frac{d}{d\eta}L_1 = \frac{d}{d\eta}\left(\frac{1}{n}\left\|Xw_0 - \frac{2\eta}{n}XX^{\mathrm{T}}a - t\right\|_2^2\right) = -\frac{4}{n^2}\left(X^{\mathrm{T}}Xa^{\mathrm{T}}\right)\left(Xw_0 - \frac{2\eta}{n}XX^{\mathrm{T}}a - t\right)$$

$$\frac{d^2}{d\eta^2}L_1 = \frac{d}{d\eta}\left(-\frac{4}{n^2}\left(X^{\mathrm{T}}Xa^{\mathrm{T}}\right)\left(Xw_0 - \frac{2\eta}{n}XX^{\mathrm{T}}a - t\right)\right)$$

$$= -\frac{4}{n^2}\left(-\frac{2}{n}X^{\mathrm{T}}Xa^{\mathrm{T}}XX^{\mathrm{T}}a\right) = \frac{8}{n^3}\left(XX^{\mathrm{T}}a\right)^{\mathrm{T}}XX^{\mathrm{T}}a$$

Since the derivative is positive everywhere, we know $L_1$ is convex w.r.t to $\eta$.

2.2.3

$$\frac{d}{d\eta}L_1 = -\frac{4}{n^2}\left(X^{\mathrm{T}}Xa^{\mathrm{T}}\right)\left(Xw_0 - \frac{2\eta}{n}XX^{\mathrm{T}}a - t\right)$$

Setting it to zero will give us the optimal learning rate $\eta^*$

$$-\frac{4}{n^2}\left(X^{\mathrm{T}}Xa^{\mathrm{T}}\right)\left(Xw_0 - \frac{2\eta}{n}XX^{\mathrm{T}}a - t\right) = 0$$

$$X^{\mathrm{T}}Xa^{\mathrm{T}}Xw_0 - \frac{2\eta}{n}X^{\mathrm{T}}Xa^{\mathrm{T}}XX^{\mathrm{T}}a - X^{\mathrm{T}}Xa^{\mathrm{T}}t = 0$$

$$\frac{2\eta}{n}X^{\mathrm{T}}Xa^{\mathrm{T}}XX^{\mathrm{T}}a = X^{\mathrm{T}}Xa^{\mathrm{T}}Xw_0 - X^{\mathrm{T}}Xa^{\mathrm{T}}t$$

$$\eta^* = \frac{n}{2}\frac{\left(XX^{\mathrm{T}}a\right)^{\mathrm{T}}(Xw_0 - t)}{(XX^{\mathrm{T}}a)^{\mathrm{T}}(XX^{\mathrm{T}}a)}$$

## Q3 Convolutional Neural Networks

### 3.1 Convolutional Filters

The resulting matrix is as follows:

$$\mathbf{I} * \mathbf{J} = \begin{bmatrix} -1 & 2 & 2 & -2 & 0 \\ -2 & 1 & 0 & 2 & -1 \\ 3 & 0 & 0 & 1 & -1 \\ -2 & 2 & 0 & 2 & -1 \\ 0 & -2 & 3 & -2 & 0 \end{bmatrix}$$

The convolutional filter detects edges of the input image as discussed in the lecture

### 3.2 Size of ConvNets

Using the equations given in the lecture, we can find the total number of parameters as follow:

Conv3-64: $\left((9 \times 3) + 1\right) \times 64 = 1792$

Conv3-128: $\left((9 \times 64) + 1\right) \times 128 = 73856$

Conv3-256: $\left((9 \times 128) + 1\right) \times 256 = 296168$

Conv3-256: $\left((9 \times 256) + 1\right) \times 256 = 590080$

FC-1024: $(256 + 1) \times 1024 = 263168$

FC-100: $(1024 + 1) \times 100 = 102500$

We can now obtain the total amount to be $1326564$ trainable parameters