Homework 2

1. Optimization

1.1 Mini-Batch Stochastic Gradient Descent (SGD)

1.1.1 Minimum Norm Solution

Recall from Question 3.3.2 from HW1, we find that the solution obtained by gradient descent is $w^* = X^T(XX^T)^{-1}t$ for $Xw^* = t$.

Let $w_0 = 0, d > n$.
Assume mini-batch SGD converges to a solution $\hat{w}$ such that $X\hat{w} = t$.

WTS: $\hat{w} = w^*$

Since x_j is the jth row of martix X, we know that x_j is contained in the span of X.

$$
\begin{aligned}
\frac{1}{b}\nabla_{w_t} L(x_j, w_t) &= \frac{1}{bn}\frac{\partial}{\partial w_t}||x_j w_t - t_j||_2^2 \\
&= \frac{2}{bn}x_j^T(x_j w_t - t_j)
\end{aligned}
$$

Notice that the gradient is spanned by the rows of X.

Now, we need to prove convergence of weights by setting the gradient of loss to weight to zero.

$$
\begin{aligned}
\frac{2}{bn}x_j^T(x_j w_t - t_j) &= 0 \\
x_j^T(x_j w_t - t_j) &= 0 \\
x_j^T x_j w_t - x_j^T t_j &= 0 \\
x_j^T x_j w_t &= x_j^T t_j \\
w_t &= \frac{x_j^T t_j}{x_j^T x_j} \\
&= \frac{t_j}{x_j^T x_j} x_j^T
\end{aligned}
$$

Notice that $t_j \in \mathbb{R}$ and $x_j^T x_j \in \mathbb{R}$. Therefore, $\frac{t_j}{x_j^T x_j} \in \mathbb{R}$. Let $c = \frac{t_j}{x_j^T x_j}$. Then, we have $w_t = cx_j^T$.

Clearly, the update steps of mini-batch SGD never leavess the span of X. Thus, we can say that every updated weight can be wrriten in terms of a linear combination of rows of X.

We can thus write $\hat{w} = X^T a$ for some $a \in \mathbb{R}^n$. Thus,

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

Therefore,

$$\begin{aligned}
XX^T a &= -t \\
a &= (\mathbf{XX}^T)^{-1} t \qquad \text{Since when } n > d,\, XX^T \text{ is invertible} \\
X^T a &= X^T (XX^T)^{-1} t \\
\hat{w} &= w^*
\end{aligned}$$

1.2 Adaptive Methods

1.2.1 Minimum Norm Solution

Let $d > n$.
Assume the RMSProp optimizer converges to a solution.
As hinted, let $x_1 = [2, 1]$, $w_0 = [0, 0]$, $t = [2]$.
As clarified in piazza @503, x_1 is a row in the data matrix X and w_0 is a column vector.

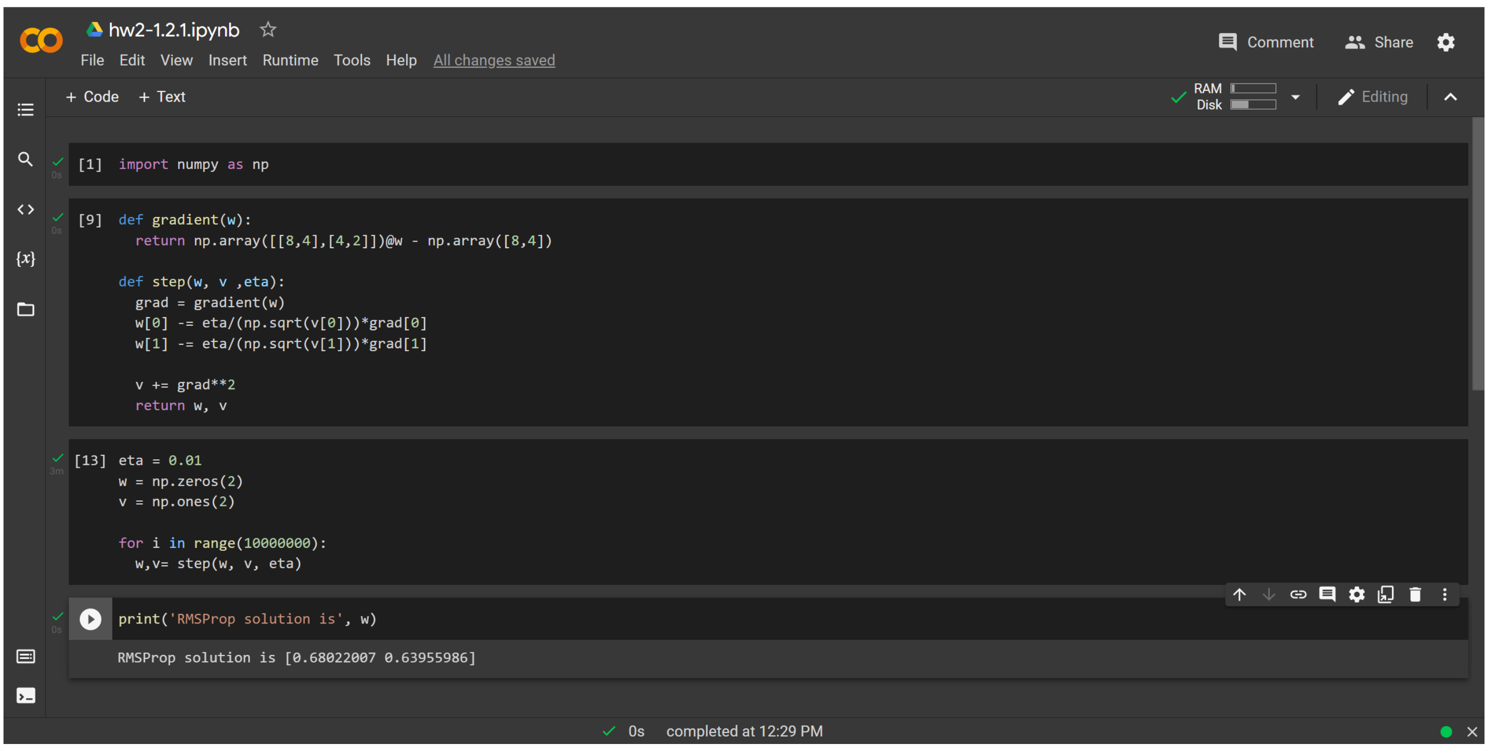
$$\begin{aligned}
w^* &= x_1 (x_1^T x_1)^{-1} t \\
&= x_1 \cdot \frac{1}{5} \cdot 2 \\
&= \frac{2}{5} x_1 \\
&= \begin{pmatrix} \frac{4}{5} \\ \frac{2}{5} \end{pmatrix} \\
&= \begin{pmatrix} 0.8 \\ 0.4 \end{pmatrix}
\end{aligned}$$

Thus, the minimum norm solution is $\frac{2}{5} x_1$.
For the RMSProp optimizer:

$$
\begin{aligned}
\nabla_w L &= \frac{2}{n} x_1 (x_1^T w - t) \\
&= \frac{2}{n} (x_1 x_1^T w - x_1 t) \\
&= \frac{2}{n} \left[\begin{pmatrix} 2 \\ 1 \end{pmatrix} \begin{pmatrix} 2 & 1 \end{pmatrix} w - \begin{pmatrix} 2 \\ 1 \end{pmatrix} t \right] \\
&= \frac{2}{n} \left[\begin{pmatrix} 4 & 2 \\ 2 & 1 \end{pmatrix} w - \begin{pmatrix} 2 \\ 1 \end{pmatrix} t \right] \\
&= \frac{2}{n} \left[\begin{pmatrix} 4 & 2 \\ 2 & 1 \end{pmatrix} w - \begin{pmatrix} 4 \\ 2 \end{pmatrix} \right]
\end{aligned}
$$

Let $n = 1$. Then, we have $\nabla_w L = \left[\begin{pmatrix} 8 & 4 \\ 4 & 2 \end{pmatrix} w - \begin{pmatrix} 8 \\ 4 \end{pmatrix} \right]$

Then, we need to check whether it converges to the minimum norm solution. Inspired by piazza @460, I decided to write some code to see what the RMSProp converges to.

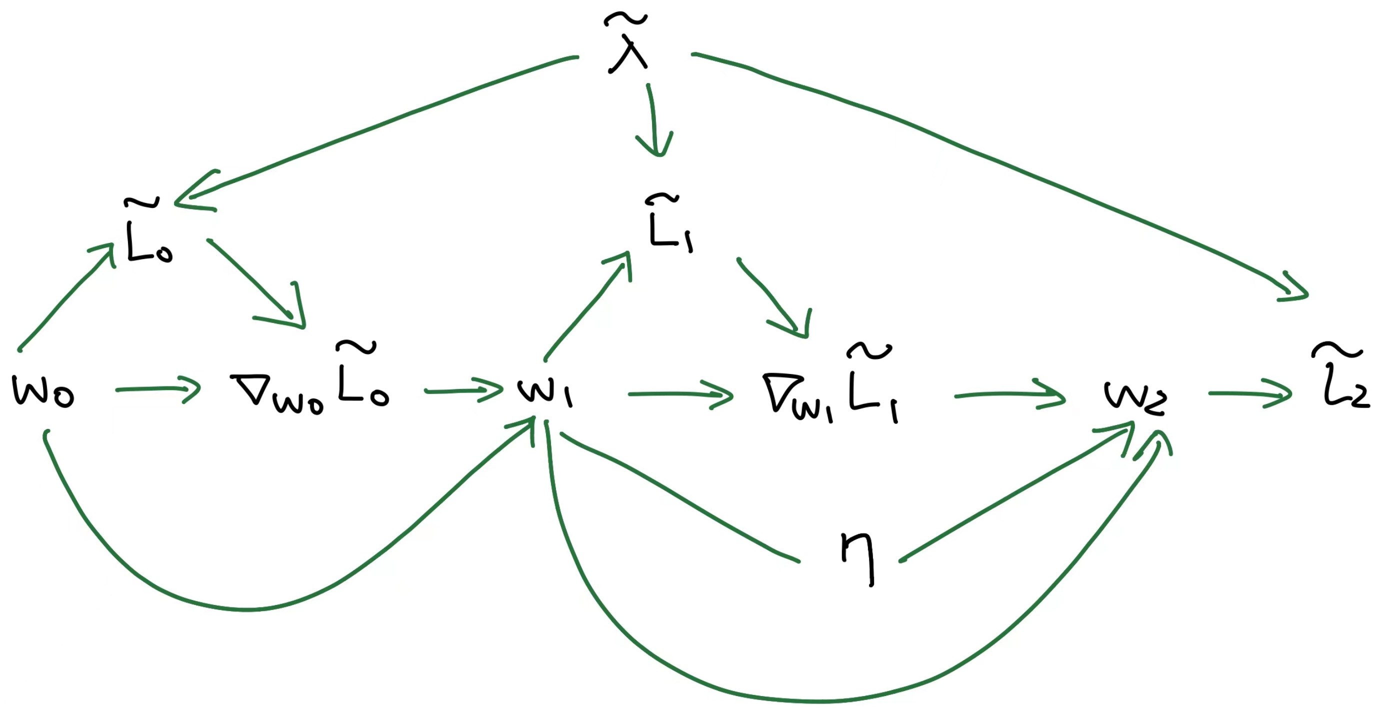


Clearly, the RMSProp result converges to a different solution when $\eta = 0.01$ and therefore, RMSProp does not always obtain the minimum norm solution which is $\begin{pmatrix} 0.8 \\ 0.4 \end{pmatrix}$.

2. Gradient-based Hyper-parameter Optimization

2.1 Computation Graph

2.1.1



2.1.2

The memory complexity for the forward-propagaion is O(1) and for the standard backward-propagation is O(t).

2.2 Optimal Learning Rates

2.2.1

Recall that $L = \frac{1}{n}||X\hat{w} - t||_2^2$ and $\nabla L = \frac{2}{n}X^T(X\hat{w} - t)$.
Therefore, we have $w_{i+1} = w_i - \frac{2\eta}{n}X^T(Xw_i - t)$.
Accordingly, we know that $w_1 = w_0 - \frac{2\eta}{n}X^T(Xw_0 - t)$.
As hinted, let $a = Xw_0 - t$.
Then, we have $w_1 = w_0 - \frac{2\eta}{n}X^Ta$.
Then,

$$
\begin{aligned}
L_1 &= \frac{1}{n}||Xw_1 - t||_2^2 \\
&= \frac{1}{n}||X(w_0 - \frac{2\eta}{n}X^T a) - t||_2^2 \\
&= \frac{1}{n}||Xw_0 - \frac{2\eta}{n}XX^T a - t||_2^2 \\
&= \frac{1}{n}||Xw_0 - t - \frac{2\eta}{n}XX^T a||_2^2 \\
&= \frac{1}{n}||a - \frac{2\eta}{n}XX^T a||_2^2 \\
&= \frac{1}{n}||a - cXX^T a||_2^2 && \text{let } c = \tfrac{2\eta}{n} \\
&= \frac{1}{n}\left[a - cXX^T a\right]^T \left[a - cXX^T a\right] \\
&= \frac{1}{n}\left[a^T - \left[cXX^T a\right]^T\right] \left[a - cXX^T a\right] && \text{since } (A-B)^T = A^T - B^T \\
&= \frac{1}{n}\left[a^T - ca^T XX^T\right] \left[a - cXX^T a\right] && \text{since } (ABC)^T = C^T B^T A^T \\
&= \frac{1}{n}\left[a^T a - a^T cXX^T a - ca^T XX^T a + c^2 a^T XX^T XX^T a\right] \\
&= \frac{1}{n}\left[a^T a - 2a^T cXX^T a + c^2 a^T XX^T XX^T a\right] \\
&= \frac{1}{n}a^T \left[I - 2cXX^T + c^2 XX^T XX^T\right] a \\
&= \frac{1}{n}a^T \left[I - cXX^T\right]^2 a \\
&= \frac{1}{n}a^T \left[I - \frac{2\eta}{n}XX^T\right]^2 a \\
&= \frac{1}{n}\left[Xw_0 - t\right]^T \left[I - \frac{2\eta}{n}XX^T\right]^2 \left[Xw_0 - t\right]
\end{aligned}
$$

2.2.3

$$
\begin{aligned}
\nabla_\eta L_1 &= \nabla_\eta \left[\frac{1}{n}a^T \left[I - \frac{2\eta}{n}XX^T\right]^2 a\right] \\
&= \frac{1}{n}\nabla_\eta \left[a^T \left[I - \frac{2\eta}{n}XX^T\right]^2 a\right] \\
&= \frac{1}{n}a^T 2\left[I - \frac{2\eta}{n}XX^T\right]\left[-\frac{2}{n}XX^T\right] a
\end{aligned}
$$

Then, we need to set it to zero for GD.

Thus, we have,

$$
\begin{aligned}
a^T 2\left[I-\frac{2\eta^*}{n}XX^T\right]\left[-\frac{2}{n}XX^T\right]a=0 \\
a^T\left[I-\frac{2\eta^*}{n}XX^T\right]\left[XX^T\right]a=0 \\
a^T[XX^T-\frac{2\eta^*}{n}XX^TXX^T]a=0 \\
a^TXX^Ta-a^T\frac{2\eta^*}{n}XX^TXX^Ta=0 \\
a^T\frac{2\eta^*}{n}XX^TXX^Ta=a^TXX^Ta \\
\frac{2\eta^*}{n}=\frac{a^TXX^Ta}{a^TXX^TXX^Ta} \\
\eta^*=\frac{na^TXX^Ta}{2a^TXX^TXX^Ta} \\
\eta^*=\frac{n(X^Ta)^TX^Ta}{2(XX^Ta)^TXX^Ta} \\
\eta^*=\frac{n}{2}\cdot\frac{(X^Ta)^T(X^Ta)}{(XX^Ta)^T(XX^Ta)}
\end{aligned}
$$

2.3 Weight decay and L2 regularization

2.3.1

We know that $\tilde{L}=L+\tilde{\lambda}||w||_2^2$.
Also, recall from 2.2.1, $\nabla_{w_0}L=\frac{2}{n}X^T(Xw_0-t)=\frac{2}{n}X^Ta$ where $a=Xw_0-t$.
Therefore,

$$
\begin{aligned}
\nabla_{w_0}\tilde{L} &= \nabla_{w_0}L+\frac{\partial}{\partial w_0}\tilde{\lambda}||w_0||_2^2 \\
&= \frac{2}{n}X^Ta+\tilde{\lambda}\frac{\partial}{\partial w_0}||w_0||_2^2 \\
&= \frac{2}{n}X^Ta+\tilde{\lambda}2w_0
\end{aligned}
$$

According to equation (5) in handout and piazza @440, $w_{i+1}=(1-\lambda)w_i-\eta\nabla L_{w_i}(X)$.
Thus, $w_1=(1-\lambda)w_0-\eta\nabla L_{w_0}(X)$.
Therefore,

- expression for w_1 using $\tilde{L}$:

$$
\begin{aligned}
w_1 &= (1-\lambda)w_0 - \eta[\frac{2}{n}X^T a + \tilde{\lambda}2w_0] \\
&= (1-\lambda)w_0 - (2\eta\tilde{\lambda})w_0 - \eta\frac{2}{n}X^T a \\
&= w_0(1-\lambda-2\eta\tilde{\lambda}) - \frac{2\eta}{n}X^T a
\end{aligned}
$$

.

- expression for w_1 using $L : w_1 = (1-\lambda)w_0 - \eta(\frac{2}{n}X^T a)$.

2.3.2

Clarified by piazza @523, we are looking for the value of $\tilde{\lambda}$ that will result in the same regularization as the case in weight decay. i.e. w_1 from L_2 regularization = w_1 from weight decay.

If so, then $w_0(1-\lambda-2\eta\tilde{\lambda}) = (1-\lambda)w_0$ must be true according to 2.3.1.
Thus,

$$
\begin{aligned}
1-\lambda-2\eta\tilde{\lambda} &= 1-\lambda \\
-2\eta\tilde{\lambda} &= 1-\lambda-(1-\lambda) \\
2\eta\tilde{\lambda} &= (1-\lambda)-(1-\lambda) \\
2\eta\tilde{\lambda} &= 1-\lambda-1+\lambda \\
2\eta\tilde{\lambda} &= 0 \\
\tilde{\lambda} &= 0
\end{aligned}
$$

Therefore, setting $\tilde{\lambda} = 0$ will do the job.

3. Convolutional Neural Networks

3.1 Convolutional Filters

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

This convolutional filter detect edges of the input image.

3.2 Size of Conv Nets

- For CNN Architecture:
  - Number of parameters = $10+0+10+0+10=30$
  - Number of neurons = $32 \times 32+32 \times 32+16 \times 16+16 \times 16+8 \times 8+8 \times 8=2688$
- For FCNN Architecture:
  - Number of parameters = $[(32 \times 32+1) \times(32 \times 32)]+[(16 \times 16+1) \times(16 \times 16)]+[(8 \times 8+1) \times(8 \times 8)]=1119552$
  - Number of neurons = $32 \times 32+32 \times 32+16 \times 16+16 \times 16+8 \times 8+8 \times 8=2688$

  **note: according to piazza @419, input and output size are the same for FC.*

More trainable parameters tells us that this model requires more space compared to the other model, i.e. more trainable parameters lead to higher computational complexity.

3.3 Receptive Fields

$$
\begin{aligned}
R_3 & =1+\sum_{j=1}^{3}\left(F_j-1\right) \prod_{i=1}^{j-1} S_i \\
& =1+\left(F_1-1\right) S_0+\left(F_2-1\right) S_0 S_1+\left(F_3-1\right) S_0 S_1 S_2 \\
& =1+(5-1) * 1+(2-1) * 1 * 2+(5-1) * 1 * 2 * 1 \\
& =1+4+2+8 \\
& =15
\end{aligned}
$$

Therefore, the receptive field afterthe second convolutional layer is of size 15×15.

Besides changing the size of the filter/kernel, changing the number of pooling layers or changing the number of convolutional layers can both lead to change in the size of the receptive field of a neuron.