Deep Learning Homework 2

Ali Abbasi - 98105879

November 16, 2022

1 Problem 1

The idea of pruning network weights after training, was first introduced by LeCun, et al. at paper **Optimal Brain Damage**. In this paper they suggest Taylor series approximation of degree 2 of the loss function to be used as a metric to find importance (saliency) of each weight in the network. And then they suggest to prune the weights with the lowest importance. Pruning in this context is done by setting the weights to zero and freezing them.

1.1

So first, we find the effect of perturbing each weight on the loss function. Note that by setting a parameter to zero, the perturbation amount of that parameter is equal to the negative of value of that parameter itself.

Taylor series approximation of the loss function is given by:

$$\delta E = \sum_i g_i \delta u_i + \frac{1}{2} \sum_{i,j} h_{ij} \delta u_i \delta u_j + \mathcal{O}(\|\delta U\|^3) \tag{1.1}$$

Where u_is are weights and U is the vector of all weights. g_i is the gradient of the loss function with respect to the ith weight, h_{ij} is the Hessian of the loss function with respect to the ith and jth weights, and δU is the vector of perturbations of the weights:

$$g_i = \frac{\partial E}{\partial u_i} \tag{1.2}$$

$$h_{ij} = \frac{\partial^2 E}{\partial u_i \partial u_j} \tag{1.3}$$

So by considering $\delta u_i = u_i$ to make ith weight zero, we can find weights that have the least effect on the loss function and can set them to zero.

But more practical way of using Eq. 1.1 will be introduced in the next subsection.

1.2

As you know, computing the Hessian of the loss function is computationally expensive and of a $O(n^2)$ complexity. So the paper uses another simplification, by assuming the Hessian to be diagonal. And as mentioned, pruning is done after training the network, so it will be on a local minima of the loss function and the gradients (g_is) will be zero. And Eq. 1.1 will be simplified to:

$$\delta E = \frac{1}{2} \sum_i h_{ii} \delta u_i^2 \tag{1.4}$$

So saliency of weight u_i is given by:

$$s_i = \frac{1}{2} h_{ii} u_i^2 \tag{1.5}$$

and we can prune the weight(s) with the lowest saliency.

Our question asks to consider the Hessian to be the Identity matrix. So $h_{ii} = 1$ and saliency of weight u_i is equal to $\frac{1}{2}u_i^2$ and there would be no need to compute the Hessian.

So we can summarize the whole process as follows:

1. Train the network until reaching a good solution (local minima of the loss function).
2. Compute saliency values for each weight.
3. Prune some of the weights with the lowest saliency.
4. Go to step 1.

2 Problem 2

We know that $y_i = x_i^T b + \epsilon_i$ where $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ and we want to estimate y with $f(X) = X\hat{b}$ and find $\hat{b}$ such that empirical risk is minimized.

2.1

We can find the solution to the problem either by using the maximum likelihood approach or with minimizing the empirical risk:

$$\frac{1}{2N} \sum_i (y_i - f(x_i))^2 = \frac{1}{2N} \|y - X\hat{b}\|^2 \tag{2.1}$$

We can show that the two approaches are equivalent:

$$\epsilon \sim \mathcal{N}(0, \sigma^2 I) \implies y \sim \mathcal{N}(Xb, \sigma^2 I) \tag{2.2}$$

$$\mathcal{L} = p(y|X, b) = \prod_i \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(y_i - x_i^T b)^2\right) = \frac{1}{(2\pi\sigma^2)^{N/2}} \exp\left(-\frac{1}{2\sigma^2}\|y - Xb\|^2\right) \tag{2.3}$$

$$\ln \mathcal{L} = -\frac{N}{2}\ln(2\pi\sigma^2) - \frac{1}{2\sigma^2}\|y - Xb\|^2 \tag{2.4}$$

$$\arg\max_b \mathcal{L} = \arg\max_b \ln \mathcal{L} = \arg\min_b \frac{1}{2}\|y - Xb\|^2 \tag{2.5}$$

And we can find minimum of $\frac{1}{2}\|y - Xb\|^2$ by setting the gradient to zero:

$$\frac{\partial}{\partial b}\frac{1}{2}\|y - Xb\|^2 = -X^T(y - Xb) = 0 \tag{2.6}$$
$$X^T y = X^T X b \tag{2.7}$$
$$\tag{2.8}$$

And we know that X is full column rank (rank$(A) = d$), so $X^T X$ is invertible and we can find b as:

$$\hat{b} = (X^T X)^{-1} X^T y = X^\dagger y \tag{2.9}$$

Where $X^\dagger$ is the Moore-Penrose pseudo-inverse of X.

2.2

First we show that $\frac{1}{N}\|(XX^\dagger - I)\epsilon\|^2$ is equal to empirical risk:

$$\hat{y} = X\hat{b} \tag{2.10}$$
$$y = Xb + \epsilon \tag{2.11}$$
$$\epsilon = y - Xb \tag{2.12}$$
$$(XX^\dagger - I)\epsilon = XX^\dagger\epsilon - \epsilon \tag{2.13}$$
$$= XX^\dagger y - XX^\dagger Xb - \epsilon \tag{2.14}$$
$$= X\hat{b} - Xb - \epsilon \qquad (XX^\dagger X = X) \tag{2.15}$$
$$= \hat{y} - y \tag{2.16}$$

Now we calculate expected value of it using following properties of matrix $A \triangleq XX^\dagger$.

1. $A^2 = XX^\dagger XX^\dagger = X(X^TX)^{-1}X^TX(X^TX)^{-1}X^T = X(X^TX)^{-1}X^T = XX^\dagger = A$
2. $A^T = A \implies A^TA = A^2 = A$
3. We can show that rank$(A) =$ rank$(X) = d$. We can do that using SVD decomposition of X and $X^\dagger$.

$$XX^\dagger = U\Sigma V^T V\Sigma^\dagger U^T = U\Sigma\Sigma^\dagger U^T = U\begin{bmatrix} I_d & 0 \\ 0 & 0 \end{bmatrix} U^T \tag{2.17}$$

4. Using last section's result, number of A has d eigenvalues equal to 1 and $N - d$ eigenvalues equal to 0. So rank$(A) = d$.
(We could show that eigenvalues of A being ones and zeros using $A^2 = A$ too.)
5. Defining $B \triangleq A - I$, we can show that:

$$B^TB = (A - I)(A - I) = A^2 - 2A + I = I - A = -B \tag{2.18}$$

6. Using 4, $-B$ has d eigenvalues equal to 0 and $N - d$ eigenvalues equal to 1.
7. trace$(-B) = \sum_{i=1}^{N} \lambda_i(-B) = N - d$

Finally we are ready!

$$
\begin{aligned}
\mathbb{E}\Big[\|(XX^\dagger - I)\epsilon\|^2\Big] &= \mathbb{E}\Big[\|B\epsilon\|^2\Big] && (2.19)\\
&= \mathbb{E}\Big[\epsilon^T B^T B \epsilon\Big] = \mathbb{E}\Big[\epsilon^T (-B)\epsilon\Big] && (2.20)\\
&= \mathbb{E}\Big[\sum_{i=1}^{N}\sum_{j}^{N} -B_{i,j}\epsilon_i\epsilon_j\Big] && (2.21)\\
&= \sum_{i=1}^{N}\sum_{j=1}^{N} -B_{i,j}\,\mathbb{E}[\epsilon_i\epsilon_j] && (2.22)\\
&= \sum_{i\neq j}^{N} -B_{i,j}\,\mathbb{E}[\epsilon_i\epsilon_j] + \sum_{i=1}^{N} -B_{i,i}\,\mathbb{E}[\epsilon_i^2] && (2.23)
\end{aligned}
$$

When $i \neq j$, ϵ_i and ϵ_j are independent, so $\mathbb{E}[\epsilon_i\epsilon_j] = \mathbb{E}[\epsilon_i]\,\mathbb{E}[\epsilon_j] = 0$. And $\mathbb{E}[\epsilon_i^2] = \mathbb{E}[\epsilon_i]^2 + Var(\epsilon_i) = \sigma^2$. So we have:

$$
\begin{aligned}
\frac{1}{N}\,\mathbb{E}\Big[\|(XX^\dagger - I)\epsilon\|^2\Big] &= \frac{1}{N}\sum_{i=1}^{N} -B_{i,i}\sigma^2 && (2.24)\\
&= \frac{1}{N}\,\text{trace}(-B)\sigma^2 && (2.25)\\
&= \frac{N-d}{N}\,\sigma^2 && (2.26)
\end{aligned}
$$

2.3

As number of linearly-independent features of X increases, columns of X span a broader subspace of R^N. So we can find $\hat{b}$ such that $\hat{y}$ is closer to y; thus strength of the model increases. We can observe the same thing by result of last part of the problem. As d approaches to N, expected value of the empirical loss approaches to zero. And when $d = N$, X is square matrix and invertible we have $X^\dagger = X^{-1}$ and we can find $\hat{b}$ as $X^{-1}y$ and $\hat{y} = Xb = y$ exactly.

3 Problem 3

3.1

This part is similar to Eq. 2.6. But we will solve it again for the sake of completeness!

$$\hat{\beta} = \arg\min_{\beta} \text{MSE}(\beta) \tag{3.1}$$

$$= \arg\min_{\beta} \frac{1}{N}\sum_{i=1}^{N}(y_i - \beta^T x_i)^2 \tag{3.2}$$

$$= \arg\min_{\beta} \frac{1}{N}\|Y - X\beta\|^2 \tag{3.3}$$

$$\implies \frac{\partial}{\partial\beta}\text{MSE}(\beta) = \frac{-1}{N}X^T(Y - X\beta) = 0 \tag{3.4}$$

$$\implies \hat{\beta} = (X^T X)^{-1}X^T Y \tag{3.5}$$

$$\tag{3.6}$$

3.2

Note that in this question, we are going to add L2 regularization to the MSE loss function and not the least square error. So don't be surprised if you see an extra N in the solution!

If we add an L2 regularization term to the loss function, we have:

$$\hat{\beta} = \arg\min_{\beta} \text{MSE}(\beta) + \lambda\|\beta\|^2 \tag{3.7}$$

$$\implies \frac{2}{N}X^T(X\beta - Y) + 2\lambda\beta = 0 \tag{3.8}$$

$$(\frac{1}{N}X^T X + \lambda I)\beta = \frac{1}{N}X^T Y \tag{3.9}$$

$$(X^T X + \lambda N I)\beta = X^T Y \tag{3.10}$$

$$\hat{\beta} = (X^T X + \lambda N I)^{-1}X^T Y \tag{3.11}$$

3.3

1. F is non-singular and $\Sigma X = XF$. So:

$$\Sigma X = XF \xrightarrow{\times\Sigma^{-1}} X = \Sigma^{-1}XF \tag{3.12}$$

$$\xrightarrow{\times F^{-1}} XF^{-1} = \Sigma^{-1}X \tag{3.13}$$

$$\xrightarrow{\text{Transpose}} F^{-T}X^T = X^T\Sigma^{-1} \qquad \Sigma, \Sigma^{-1} \text{ are symmetric} \tag{3.14}$$

So we have:

$$\begin{aligned} \beta^* &= (X^T\Sigma^{-1}X)^{-1}X^T\Sigma^{-1}Y && (3.15)\\ &= (F^{-T}X^TX)^{-1}F^{-T}X^TY && (3.16)\\ &= (X^TX)^{-1}F^TF^{-T}X^TY \qquad \text{using } (AB)^{-1} = B^{-1}A^{-1} && (3.17)\\ &= (X^TX)^{-1}X^TY && (3.18)\\ &= \hat{\beta} && (3.19) \end{aligned}$$

2. If $\hat{\beta} = \beta^*$, then there is a non-singular F such that $\Sigma X = XF$:

$$\begin{aligned} \beta^* = \hat{\beta} &\implies \forall Y:\ (X^TX)^{-1}X^TY = (X^T\Sigma^{-1}X)^{-1}X^T\Sigma^{-1}Y && (3.20)\\ &\implies (X^TX)^{-1}X^T = (X^T\Sigma^{-1}X)^{-1}X^T\Sigma^{-1} && (3.21)\\ &\xrightarrow{\times X^TX} X^T = (X^TX)(X^T\Sigma^{-1}X)^{-1}X^T\Sigma^{-1} && (3.22)\\ &\xrightarrow{\times\Sigma} X^T\Sigma = (X^TX)(X^T\Sigma^{-1}X)^{-1}X^T && (3.23) \end{aligned}$$

We define: $F \triangleq (X^T\Sigma^{-1}X)^{-T}(X^TX)$. So we have:

$$X^T\Sigma = F^TX^T \implies \Sigma X = XF \tag{3.24}$$

It's enough to show that F is non-singular. We know that X^TX is invertible and $X^T\Sigma^{-1}X$ is obviously invertible. And multiplying two invertible matrices gives an invertible matrix. So F is invertible and non-singular. And we have shown its existence.

With 1 and 2, we have proved what the question asked.

3.4

1. $L(\beta, \lambda_1, \lambda_2) = \|y - X\beta\|^2 + \lambda_1\|\beta\|_2^2 + \lambda_2\|\beta\|_1$
2. $L'(\beta, \lambda_2) = \|y' - X'\beta\|^2 + \lambda_2\|\beta\|_1$

We want to add datapoints to X and y such that the value of loss function L on original data, is equal to the value of the function L' on new data.

First we note that by adding vector e_i (all zeros except for the i^{th} element which is 1) to X and zero as its corresponding label, change in MSE is equal to:

$$\|0 - e_i^T\beta\|^2 = \|\beta_i\|^2 \tag{3.25}$$

So by defining $X' and y'$ as:

$$X' = \begin{bmatrix} X \\ \sqrt{\lambda_1}e_1 \\ \sqrt{\lambda_1}e_2 \\ \vdots \\ \sqrt{\lambda_1}e_d \end{bmatrix} \ \&\ y' = \begin{bmatrix} y \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{3.26}$$

$$\text{MSE}' = \|y' - X'\beta\|^2 = \|y - X\beta\|^2 + \lambda_1\|\beta\|_2^2 = \text{MSE} + \lambda_1\|\beta\|_2^2 \tag{3.27}$$

So value of the loss function L on X and y is equal to the value of the function L' on X' and y'.

4 Problem 4

The purpose of this problem is to estimate y with $\sum_{k=1}^{n} w_k x_k$. But during the training, we will multiply weights (or equivalently inputs of the model) with a gaussian noise with distribution $\delta_i \sim \mathcal{N}(1, \sigma^2)$. We name loss function with gaussian dropout as $J_D = (y - \sum_{k=1}^{n} \delta_k w_k x_k)^2$ and normal loss function (when we don't use gaussian dropout) as $J_N = (y - \sum_{k=1}^{n} w_k x_k)^2$. And during training, we are minimizing J_D.

4.1

$$\begin{aligned}
\nabla_{w_i} J_D &= \nabla_{w_i} (y - \sum_{k=1}^{n} \delta_k w_k x_k)^2 && (4.1)\\
&= -2(y - \sum_{k=1}^{n} \delta_k w_k x_k)\delta_i x_i && (4.2)\\
&= -2(y\delta_i x_i - \delta_i^2 x_i^2 w_i - \sum_{k \neq i} \delta_k \delta_i w_k x_k x_i) && (4.3)\\
\mathbb{E}\big[\nabla_{w_i} J_D\big] &= -2(y\,\mathbb{E}[\delta_i] x_i - \mathbb{E}[\delta_i^2] x_i^2 w_i - \sum_{k \neq i} \mathbb{E}[\delta_k \delta_i] w_k x_k x_i) && (4.4)\\
\mathbb{E}[\delta_i] &= 1 && (4.5)\\
\mathbb{E}[\delta_i^2] &= \mathbb{E}[\delta_i]^2 + var(\delta_i) = 1 + \sigma^2 && (4.6)\\
\mathbb{E}[\delta_k \delta_i] &= \mathbb{E}[\delta_k]\,\mathbb{E}[\delta_i] = 1 \qquad \text{for } i \neq j && (4.7)\\
\implies \mathbb{E}\big[\nabla_{w_i} J_D\big] &= -2(y x_i - (1+\sigma^2) x_i^2 w_i - \sum_{k \neq i} w_k x_k x_i) && (4.8)\\
&= -2(y x_i - \sigma^2 x_i^2 w_i - \sum_{k=1}^{n} w_k x_k x_i) && (4.9)\\
&= -2(y - \sum_{k=1}^{n} w_k x_k) x_i + 2\sigma^2 x_i^2 w_i && (4.10)\\
\nabla_{w_i} J_N &= \nabla_{w_i} (y - \sum_{k=1}^{n} w_k x_k)^2 && (4.11)\\
\implies \mathbb{E}\big[\nabla_{w_i} J_N\big] &= \nabla_{w_i} J_N + 2\sigma^2 x_i^2 w_i && (4.12)
\end{aligned}$$

4.2

Note how the term added to the gradient of normal loss function resembles ordinary weight decay regularization! It's like $\nabla_{w_i} \sigma^2 \sum_i x_i^2 w_i^2$ is added to the gradient of normal loss function. Which is like a version of L2 regularization where $\lambda = \sigma^2$ and each w_i is punished differently proportional to the square of its corresponding element in data, x_i^2.

5 Problem 5

We want to minimize $w^T H w$ and we know $H = Q\Lambda Q^T$.

5.1

$$w_t = w_{t-1} - \epsilon \nabla_w (w^T H w)|_{w=w_{t-1}} = w_{t-1} - 2\epsilon H w_{t-1} \qquad H \text{ is symmetric} \tag{5.1}$$

5.2

$$w_t = (I - 2\epsilon H) w_{t-1} \tag{5.2}$$
$$\implies w_t = (I - 2\epsilon H)^t w_0 \tag{5.3}$$

5.3

We know that H is decomposable using eigen decomposition. So $I - 2\epsilon H$ too is decomposable using Q as its matrix of eigen-vectors and its eigenvalues are $1 - 2\epsilon\lambda_i$ where λ_i are the eigenvalues of H.

$$w^* = \lim_{t\to\infty} w_t = \lim_{t\to\infty} (I - 2\epsilon H)^t w_0 \tag{5.4}$$
$$= \lim_{t\to\infty} Q A^t Q^T w_0 \tag{5.5}$$

Where

$$A = \begin{bmatrix} 1-2\epsilon\lambda_1 & 0 & \cdots & 0 \\ 0 & 1-2\epsilon\lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1-2\epsilon\lambda_n \end{bmatrix}$$

and

$$A^t = \begin{bmatrix} (1-2\epsilon\lambda_1)^t & 0 & \cdots & 0 \\ 0 & (1-2\epsilon\lambda_2)^t & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & (1-2\epsilon\lambda_n)^t \end{bmatrix}$$

So if absolute value of all eigenvalues of A are less than 1, then w^* will converge and will converge to zero. Otherwise, it will diverge.

$$-1 < 1 - 2\epsilon\lambda_i < 1 \implies -2 < -2\epsilon\lambda_i < 0 \implies 0 < \epsilon\lambda_i < 1 \tag{5.6}$$
$$\implies \begin{cases} \forall i, \lambda_i > 0 \implies H \text{ is Positive Definite} \\ \forall i, \epsilon < \frac{1}{\lambda_i} \implies \epsilon\epsilon < \frac{1}{\lambda_{max}} \end{cases} \tag{5.7}$$

5.4

$$w_{t+1} = w_t - \nabla_w^2 (w^T H w)^{-1} \nabla_w (w^T H w)|_{w=w_t} \tag{5.8}$$
$$= w_t - (2H)^{-1}(2Hw_t) \tag{5.9}$$
$$= w_t - w_t \tag{5.10}$$
$$= 0 \tag{5.11}$$

It will converge to minimum in a single step (of course 0 is minimum of this function if H is positive semi-definite matrix. Other wise it can take any negative value too).

5.5

Because gradient descent minimizes loss function using its (first order) derivative. But Newton method uses second order derivative to minimize the loss function. And computing second order derivatives is of $O(n^2)$ complexity and computing its inverse is of $O(n^3)$ (however there are more efficient methods for computing inverse of a matrix).

Because of more information it has about the loss function, Newton method can converge faster than gradient descent and its a good choice when second order derivative is available or easy to compute (for example, sometimes Newton method is used for logistic regression). But in neural networks where we have thousands to millions of parameters, paying cost of $O(n^2)$ for computing second order derivative is not feasible and gradient descent is a better choice.

6 Problem 6

6.1

This network can be used to learn an embedding h_1 and h_2 for each of its input x_1 and x_2 respectively such that h_1 and h_2 are close to each other in the embedding space if x_1 and x_2 are similar (if they are a positive pair!).

6.2

Considering that we have a batch of B pairs (to not be confused with bias) of inputs, we define loss function for i^{th} pair as:

$$h_1^{(i)} = \tanh(Wx_1^{(i)} + b) \tag{6.1}$$

$$h_2^{(i)} = \tanh(Wx_2^{(i)} + b) \tag{6.2}$$

$$J^{(i)} = \|h_1^{(i)} - h_2^{(i)}\|^2 \tag{6.3}$$

(we'll add regularization to the whole cost function)

$$\nabla_W J^{(i)} = 2(h_1^{(i)} - h_2^{(i)})((1 - h_1^{(i)} \odot h_1^{(i)}){x_1^{(i)}}^T + (1 - h_2^{(i)} \odot h_2^{(i)}){x_2^{(i)}}^T) \tag{6.4}$$

$$\nabla_b J^{(i)} = 2(h_1^{(i)} - h_2^{(i)})((1 - h_1^{(i)} \odot h_1^{(i)}) + (1 - h_2^{(i)} \odot h_2^{(i)})) \tag{6.5}$$

Where we have used the fact that $\tanh'(z) = 1 - \tanh^2(z)$ and $\odot$ is element wise multiplication (as we know tanh activation is applied elementwise).

So the total loss is:

$$J = \|W\|_F^2 + \sum_{i}^{B} J^{(i)} \tag{6.6}$$

$$\nabla J = \nabla \|W\|_F^2 + \sum_{i=1}^{B} \nabla J^{(i)} \tag{6.7}$$

$$\implies \nabla_W J = 2W + \sum_{i=1}^{B} 2(h_1^{(i)} - h_2^{(i)})((1 - h_1^{(i)} \odot h_1^{(i)}) {x_1^{(i)}}^T + (1 - h_2^{(i)} \odot h_2^{(i)}) {x_2^{(i)}}^T) \tag{6.8}$$

$$\nabla_b J = \sum_{i=1}^{B} 2(h_1^{(i)} - h_2^{(i)})((1 - h_1^{(i)} \odot h_1^{(i)}) + (1 - h_2^{(i)} \odot h_2^{(i)})) \tag{6.9}$$

So we can update the parameters using gradient descent as:

$$W \leftarrow W - \eta \nabla_W J \tag{6.10}$$

$$b \leftarrow b - \eta \nabla_b J \tag{6.11}$$