AI6104 - Mathematics for AI

Tutorial 6 - Matrix Calculus

Problem 1

Let A be an invertible $m \times m$ matrix whose elements are functions of a scalar parameter α. Prove that

$$\frac{\partial A^{-1}}{\partial \alpha} = -A^{-1}\frac{\partial A}{\partial \alpha}A^{-1}$$

Hint: by the definition of inversion, we have $A^{-1}A = I$.

> *Solution:*
>
> By the definition of inversion, we have
>
> $$A^{-1}A = I$$
>
> Then we differentiate on both sides,
>
> $$\frac{\partial A^{-1}}{\partial \alpha} + A^{-1}\frac{\partial A}{\partial \alpha}A^{-1} = 0$$
>
> which concludes the proof.

Problem 2

We will look at the backpropagation in one layer of a neural network (see Figure 1).

In neural networks, a layer f is a function of input X and weight W, where the output is $Y = f(X, W)$. In this problem, we assume a linear layer, $Y = f(X, W) = XW$. If we consider the input X has N samples, and each sample, $x^{(i)}$, is a D-dimensional vector, then X is an $N \times D$ matrix. Similarly, we have corresponding N output vectors, and each $y^{(i)}$ is M-dimensional, which forms a $N \times M$ matrix Y. Thus, the weight matrix W has shape $D \times M$. Similar layers as described above are embedded into larger neural networks with loss, usually a scalar, L.

(a) During backpropagation, we want to know the gradient of loss with respect to the input X and weight W. Write down $\frac{\partial L}{\partial X}$ and $\frac{\partial L}{\partial W}$ using chain rule.

> *Solution:*
>
> $$\frac{\partial L}{\partial X} = \frac{\partial L}{\partial Y}\frac{\partial Y}{\partial X}, \quad \frac{\partial L}{\partial W} = \frac{\partial L}{\partial Y}\frac{\partial Y}{\partial W}$$

(b) According to part (a), discuss the difficulties of calculating $\frac{\partial L}{\partial X}$ and $\frac{\partial L}{\partial W}$ using the matrix multiplications explicitly.

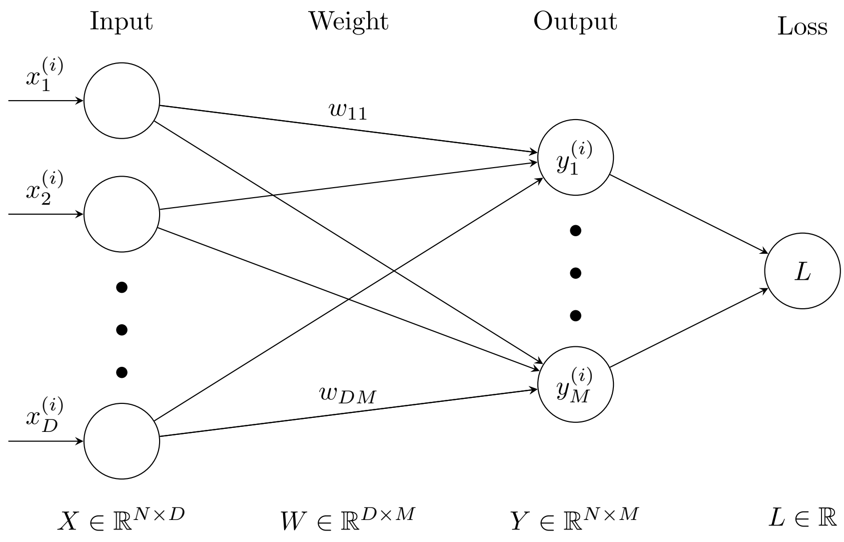

Figure 1: An illustration for Problem 2

Solution:
Calculating the $\frac{\partial L}{\partial X}$ using matrix multiplication directly needs $\frac{\partial Y}{\partial X}$, which is a Jacobian matrix. According to the settings above, this Jacobian matrix is of size $(N \times M) \times (N \times D)$. In a common deep learning setup, we might have $N = 64$ and $M = D = 4096$, which would result in $64 \cdot 4096 \cdot 64 \cdot 4096 \approx 6.8 \times 10^{10}$ scalar values. Using 32-bit floating point, this Jacobian matrix will take 256 GB of memory to store.

Problem 3

Suppose a linear layer, $Y = f(X, W)$, in a neural network has activation function $A(Y)$. Calculate $\frac{\partial A}{\partial Y}$ for the following different activation functions.

(a) **ReLU**: $R(Y) = \max\{0, Y\}$

(b) **Sigmoid**: $S(Y) = \frac{1}{1+e^{-Y}}$

(c) **Tanh**: $\tanh(Y) = \frac{e^Y - e^{-Y}}{e^Y + e^{-Y}}$

Solution:

(a) $R'(Y) = \begin{cases} 1, & Y > 0 \\ 0, & Y \leq 0 \end{cases}$

(b) $S'(Y) = S(Y)(1 - S(Y))$

(c) $\tanh'(Y) = 1 - \tanh^2(Y)$

Problem 4

This problem requires computing the gradients of a full neural network. In particular we are going to compute the gradients of a neural network with one hidden layer trained with cross-entropy loss. The forward pass of single D-dimensional input sample x is as follows:

$$
\begin{aligned}
x &= \text{input} \in \mathbb{R}^D \\
z &= Wx \in \mathbb{R}^M \\
h &= \text{ReLU}(z) \in \mathbb{R}^M \\
\theta &= Uh \in \mathbb{R}^C \\
\hat{y} &= \text{softmax}(\theta) \in \mathbb{R}^C \\
J &= \text{CrossEntropy}(y, \hat{y}) \in \mathbb{R}
\end{aligned}
$$

where W and U are weight matrices. y is the true label vector. The softmax activation of the j-th output unit is

$$\hat{y_j} = \text{softmax}(\theta_j) = \frac{e^{\theta_j}}{\sum_j e^{\theta_j}}$$

The cross-entropy error function is

$$\text{CrossEntropy}(y, \hat{y}) = -\sum_j y_j \log \hat{y}_j$$

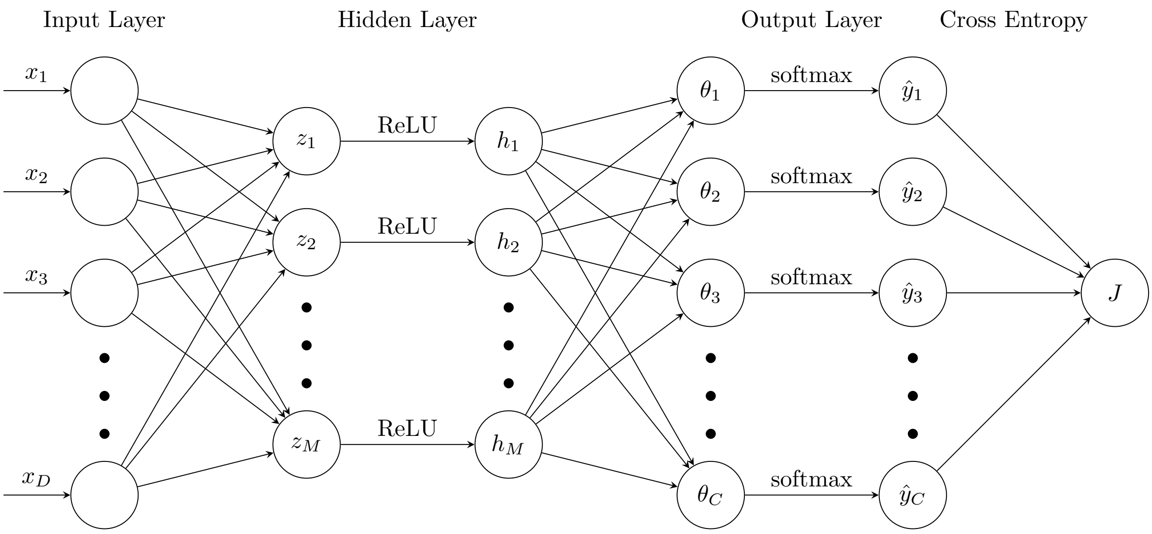

Figure 2: A neural network with one hidden layer

(a) Calculate the gradient of cross entropy error with respect to the logits, *i.e.*, $\frac{\partial J}{\partial \theta}$.

(b) According to above information, compute all of the network's gradients, $\frac{\partial J}{\partial U}, \frac{\partial J}{\partial W}, \frac{\partial J}{\partial x}$. *Hint: In practice, you may transpose your vectors to match the dimensions.*

Solution:

(a) It is easy to show that

$$\frac{\partial J}{\partial \hat{y}} = -\frac{y}{\hat{y}}$$

For the gradient of softmax function, we look at a single entry first,

$$\begin{aligned}\frac{\partial \hat{y}_j}{\partial \theta_k} &= \begin{cases} \frac{e^{\theta_j}\left(\sum_k e^{\theta_k} - e^{\theta_j}\right)}{\left(\sum_k e^{\theta_k}\right)^2}, \ j = k \\ -\frac{e^{\theta_j} e^{\theta_k}}{\left(\sum_k e^{\theta_k}\right)^2}, \ j \neq k \end{cases} \\ &= \begin{cases} \hat{y}_j\left(1 - \hat{y}_j\right) \\ -\hat{y}_j \hat{y}_k \end{cases}\end{aligned}$$

By chain rule, we can derive each entry of the gradient

$$\begin{aligned}\frac{\partial J}{\partial \theta_k} &= \sum_j \frac{\partial J}{\partial \hat{y}_j}\frac{\partial \hat{y}_j}{\partial \theta_k} \\ &= \frac{\partial J}{\partial \hat{y}_k}\frac{\partial \hat{y}_k}{\partial \theta_k} - \sum_{j \neq k} \frac{\partial J}{\partial \hat{y}_j}\frac{\partial \hat{y}_j}{\partial \theta_k} \\ &= -y_k\left(1 - \hat{y}_k\right) + \sum_{j \neq k} y_j \hat{y}_k \\ &= \hat{y}_k - y_k\end{aligned}$$

Therefore, the gradient can be written as

$$\frac{\partial J}{\partial \theta} = \hat{y} - y$$

(b) Recall that $\mathrm{ReLU}(x) = \max(x, 0)$, which gives the gradient

$$\mathrm{ReLU}' = \begin{cases} 1, & \text{if } x \geq 0 \\ 0, & \text{if otherwise} \end{cases} = \mathrm{sgn}(\mathrm{ReLU(x)})$$

where sgn(x) is the sign function. We then write down the chain rule for the gradients

$$\begin{aligned}\frac{\partial J}{\partial U} &= \frac{\partial J}{\partial \theta}\frac{\partial \theta}{\partial U} \\ \frac{\partial J}{\partial W} &= \frac{\partial J}{\partial z}\frac{\partial z}{\partial W} \\ &= \frac{\partial J}{\partial \theta}\frac{\partial \theta}{\partial h}\frac{\partial h}{\partial z}\frac{\partial z}{\partial W} \\ \frac{\partial J}{\partial x} &= \frac{\partial J}{\partial z}\frac{\partial z}{\partial x} \\ &= \frac{\partial J}{\partial \theta}\frac{\partial \theta}{\partial h}\frac{\partial h}{\partial z}\frac{\partial z}{\partial x}\end{aligned}$$

It is noticed that $\frac{\partial J}{\partial \theta}$ and $\frac{\partial J}{\partial z}$ are used multiple times, it is convenient to denote that

$$\delta_1 = \frac{\partial J}{\partial \theta} = (\hat{y} - y)^\top$$

and

$$\begin{aligned}\delta_2 = \frac{\partial J}{\partial z} &= \frac{\partial J}{\partial \theta}\frac{\partial \theta}{\partial h}\frac{\partial h}{\partial z} \\ &= \delta_1 \frac{\partial \theta}{\partial h}\frac{\partial h}{\partial z} \\ &= \delta_1 U \circ \operatorname{sgn}(h)\end{aligned}$$

where $\circ$ denotes element-wise multiplication. Here we transpose the vectors to match the dimension. Now we can use δ_1 and δ_2 to compute our gradients. You may check the dimensions of all the terms in the gradients.

$$\begin{aligned}\frac{\partial J}{\partial U} &= \delta_1^\top h^\top \\ \frac{\partial J}{\partial W} &= \delta_2^\top x^\top \\ \frac{\partial J}{\partial x} &= (\delta_2 W)^\top\end{aligned}$$