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 $A^{-1}A = I$.

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.

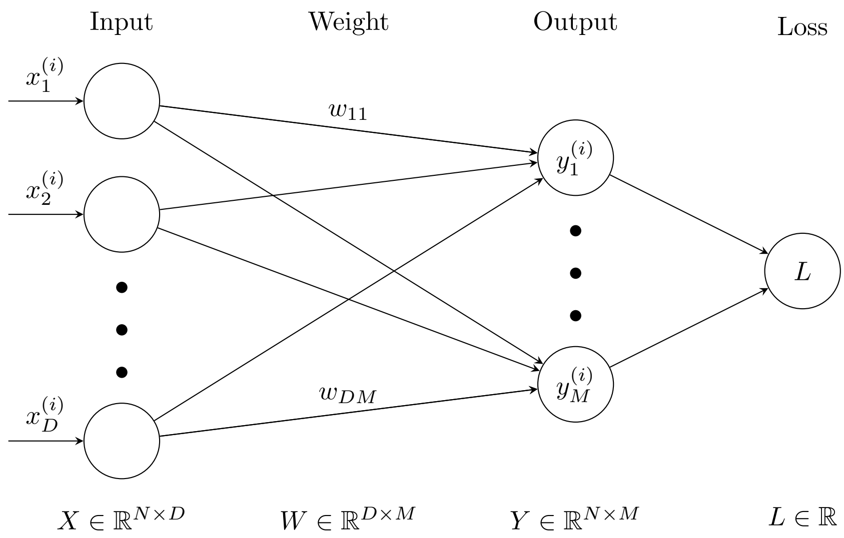

Figure 1: An illustration for Problem 2

(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.

(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.

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

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

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

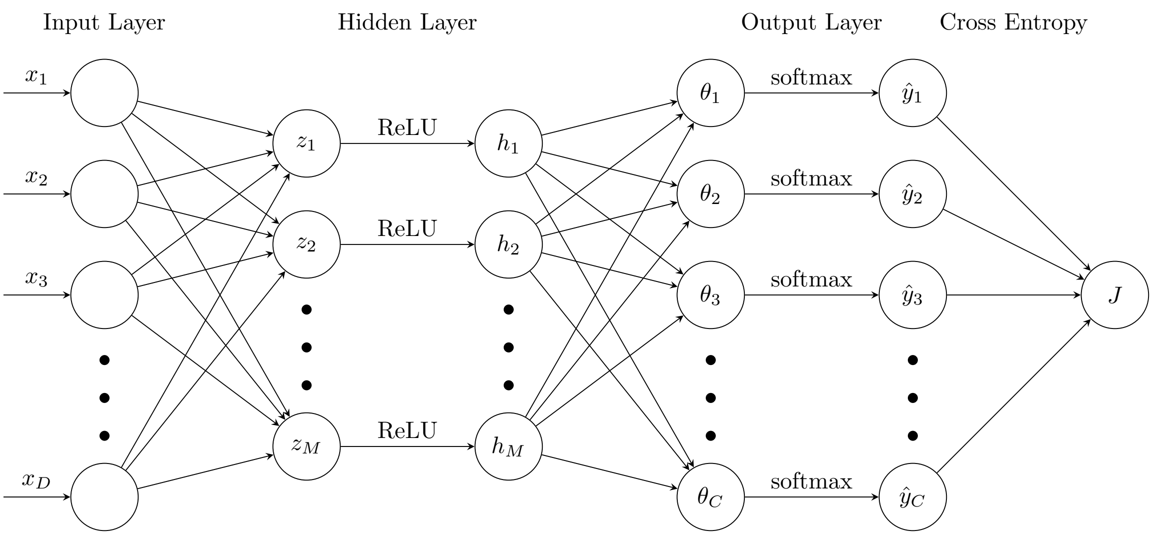

Figure 2: A neural network with one hidden layer

(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.