Multi-layer Perceptron

Chang Lu

1 Model

Let $\mathbf{x} \in \mathbb{R}^{s \times n}$ be the inputs, where n is the number of samples; and s is the dimension of each feature vector. We first define a hidden layer t in multi-layer perceptron (MLP):

$$\mathbf{z}^{(t)} = \mathbf{w}^{(t)}\mathbf{h}^{(t-1)} + \mathbf{b}^{(t)}, \tag{1}$$

$$\mathbf{h}^{(t)} = g^{(t)}\left(\mathbf{z}^{(t)}\right). \tag{2}$$

Here, $\mathbf{h}^{(t)} \in \mathbb{R}^{d^{(t)} \times n}$ is the output of the t-th layer. $\mathbf{h}^{(0)} = \mathbf{x}$. $\mathbf{w}^{(t)} \in \mathbb{R}^{d^{(t)} \times d^{(t-1)}}$ is the weight to map the output of the $(t-1)$-th layer into the intermediate output $\mathbf{z}$. $\mathbf{b}^{(t)} \in \mathbb{R}^{d^{(t)} \times 1}$. $g^{(t)}$ is an activation function such as ReLU, sigmoid, or tanh. Typically, $g^{(t)}$ is the same among hidden layers in MLP, except the output layer. We denote the activation function in hidden layers as g.

In the output layer, the activation function and dimension of weights depend on specific tasks and labels. Let C be the output dimension and f be the activation function, the output layer is defined as follows:

$$\mathbf{z}^{(T)} = \mathbf{w}^{(T)}\mathbf{h}^{(T-1)} + \mathbf{b}^{(T)}, \tag{3}$$

$$\hat{\mathbf{y}} = f\left(\mathbf{z}^{(T)}\right). \tag{4}$$

Here, $\mathbf{w}^{(T)} \in \mathbb{R}^{C \times d^{(T-1)}}$. Traditionally, there is an argument about counting the layers in MLP. In our project, we count one layer by the weight w. Figure 1 shows an example of a 2-layer MLP. It contains two hidden weight variables. We call it 2-layer MLP (input neurons, hidden neurons, and output neurons).

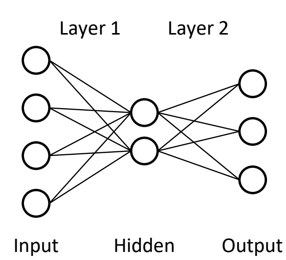

Figure 1: A 2-layer MLP

2 Objective Function

The objective function depends on various tasks. We use a multi-class classification as an example. The objective function is selected as multi-class cross-entropy loss. Let C be the output dimension, i.e., number of categories and $\mathbf{y} \in \{0,1\}^{C \times n}$ be a one-hot vector for the ground-truth, the prediction of MLP is $\hat{\mathbf{y}} \in \mathbb{R}^{C \times n}$ is a probability distribution of each entry from $\mathbf{z}^{(T)}$, calculated by a softmax activation function:

$$\hat{\mathbf{y}}_c = \frac{e^{\mathbf{z}_c^{(T)}}}{\sum_{k=1}^{C} e^{\mathbf{z}_k^{(T)}}}. \tag{5}$$

For an input sample $\mathbf{x}_i$, the multi-class cross-entropy loss $L(\mathbf{x}_i, \mathbf{y}_i \mid \mathbf{w}^{(1)}, \dots, \mathbf{w}^{(T)})$ is:

$$
\begin{aligned}
L(\mathbf{x}_i, \mathbf{y}_i \mid \mathbf{w}^{(1)}, \dots, \mathbf{w}^{(T)}) &= -\sum_{c=1}^{C} \mathbf{y}_{c,i} \log \hat{\mathbf{y}}_{c,i} &(6)\\
&= -\sum_{c=1}^{C} \mathbf{y}_{c,i} \left(\mathbf{z}_{c,i}^{(T)} - \log \sum_{k=1}^{C} e^{\mathbf{z}_{i,k}^{(T)}} \right) \\
&= \log \sum_{c=1}^{C} e^{\mathbf{z}_{c,i}^{(T)}} - \sum_{c=1}^{C} \mathbf{y}_{c,i} \mathbf{z}_{c,i}^{(T)}. &(7)
\end{aligned}
$$

For all input samples $\mathbf{x}$, the total loss is an average of the losses for all samples:

$$
\begin{aligned}
L(\mathbf{x}, \mathbf{y} \mid \mathbf{w}^{(1)}, \dots, \mathbf{w}^{(T)}) &= -\frac{1}{n} \sum_{i=1}^{n} \sum_{c=1}^{C} \mathbf{y}_{c,i} \log \hat{\mathbf{y}}_{c,i} \\
&= \frac{1}{n} \sum_{i=1}^{n} \left(\log \sum_{c=1}^{C} e^{\mathbf{z}_{c,i}^{(T)}} - \sum_{c=1}^{C} \mathbf{y}_{c,i} \mathbf{z}_{c,i}^{(T)} \right). &(8)
\end{aligned}
$$

2.1 A Trick for Stable Softmax and Log-Sum-Exp

When calculating e^x for softmax function in Equation (5) and Equation (7), x can be large so that there may be a numerical problem. To alleviate this problem, we can take advantage a property of softmax:

$$
\begin{aligned}
\operatorname{softmax}(x_c \mid x_1, \dots, x_C) &= \frac{e^{x_c}}{\sum_{k=1}^{C} e^{x_c}} = \frac{e^{-\lambda} \cdot e^{x_c}}{e^{-\lambda} \cdot \sum_{k=1}^{C} e^{x_c}} = \frac{e^{x_c - \lambda}}{\sum_{k=1}^{C} e^{x_c - \lambda}} \\
&= \operatorname{softmax}(x_c - \lambda \mid x_1 - \lambda, \dots, x_C - \lambda). &(9)
\end{aligned}
$$

Similarly, we can get

$$
\log \sum_{c=1}^{C} e^{x_c} = \log \left(e^{\lambda} \cdot e^{-\lambda} \cdot \sum_{c=1}^{C} e^{x_c} \right) = \lambda + \log \sum_{c=1}^{C} e^{x_c - \lambda}. \tag{10}
$$

Let $\lambda = \max\{x_1, x_2, \dots, x_C\}$, for $\forall c \geq 1$ and $c \leq C$, we have $x_c - \lambda \leq 0 \Rightarrow e^{x_c - \lambda} \leq 1$. In this way, the softmax and log-sum-exp operations can be numerically stable.

3 Back-propagation

We use the multi-class classification for predictions and ReLU activation for hidden layers. The trainable variables are $\mathbf{w}^{(1)}, \dots, \mathbf{w}^{(T)}$ and $\mathbf{b}^{(1)}, \dots, \mathbf{b}^{(T)}$. We need to calculate gradients for all of these variables. For $\mathbf{w}^{(t)}$, we need to apply the chain rule:

$$
\frac{\partial L}{\partial \mathbf{w}^{(t)}} = \frac{\partial L}{\partial \mathbf{z}^{(t)}} \frac{\partial \mathbf{z}^{(t)}}{\partial \mathbf{w}^{(t)}} \in \mathbb{R}^{d^{(t-1)} \times d^{(t)}}.
$$

For the first term, it is not easy to directly get the gradient of $\mathbf{z}^{(t)}$. For the second term, it is a matrix-by-matrix derivative, we cannot directly calculate either. Therefore, we first consider the loss of one sample: $L_i = L(\mathbf{x}_i, \mathbf{y}_i)$. Then we seek to pass gradients from the $(t+1)$-th layer w.r.t. the k-th row of $\mathbf{w}^t$. Let

$$
\begin{aligned}
\phi(t, i) &= \frac{\partial L_i}{\partial \mathbf{z}_{:i}^{(t)}} \in \mathbb{R}^{1 \times d^{(t)}}, \\
\psi(t, i, k) &= \frac{\partial L_i}{\partial \mathbf{w}_{k:}^{(t)\top}} \in \mathbb{R}^{1 \times d^{(t-1)}},
\end{aligned}
$$

we have

$$\phi(t,i) = \underbrace{\phi(t+1,i)}_{1\times d^{(t+1)}} \underbrace{\frac{\partial \mathbf{z}_{:i}^{(t+1)}}{\partial \mathbf{z}_{:i}^{(t)}}}_{d^{(t+1)}\times d^{(t)}},$$

$$\psi(t,i,k) = \underbrace{\phi(t,i)}_{1\times d^{(t)}} \underbrace{\frac{\partial \mathbf{z}_{:i}^{(t)}}{\partial {\mathbf{w}_{k:}^{(t)}}^{\top}}}_{d^{(t)}\times d^{(t-1)}}.$$

In this way, we only need to calculate $\phi(T,i)$, $\frac{\partial \mathbf{z}_{:i}^{(t+1)}}{\partial \mathbf{z}_{:i}^{(t)}}$, and $\frac{\partial \mathbf{z}_{:i}^{(t)}}{\partial {\mathbf{w}_{k:}^{(t)}}^{\top}}$, which is easier and intuitive. Then a recursive way can be applied to calculate each $\mathbf{w}^{(t)}$.

$$\begin{aligned}
\phi(T,i) &= \frac{\partial}{\partial \mathbf{z}_{:i}^{(T)}} \left(\log \sum_{c=1}^{C} e^{\mathbf{z}_{c,i}^{(T)}} - \sum_{c=1}^{C} \mathbf{y}_{c,i} \mathbf{z}_{c,i}^{(T)} \right) \\
&= \frac{\partial}{\partial \mathbf{z}_{:i}^{(T)}} \log \sum_{c=1}^{C} e^{\mathbf{z}_{c,i}^{(T)}} - \frac{\partial}{\partial \mathbf{z}_{:i}^{(T)}} \sum_{c=1}^{C} \mathbf{y}_{c,i} \mathbf{z}_{c,i}^{(T)} \\
&= \frac{1}{\sum_{c=1}^{C} e^{\mathbf{z}_{c,i}^{(T)}}} \cdot \frac{\partial}{\partial \mathbf{z}_{:i}^{(T)}} \left(\sum_{c=1}^{C} e^{\mathbf{z}_{c,i}^{(T)}} \right) - \frac{\partial}{\partial \mathbf{z}_{:i}^{(T)}} \left(\sum_{c=1}^{C} \mathbf{y}_{c,i} \mathbf{z}_{c,i}^{(T)} \right) \\
&= \frac{1}{\sum_{c=1}^{C} e^{\mathbf{z}_{c,i}^{(T)}}} \cdot e^{{\mathbf{z}_{:i}^{(T)}}^{\top}} - \mathbf{y}_{:i}^{\top} \\
&= \left(\hat{\mathbf{y}}_{:i} - \mathbf{y}_{:i}\right)^{\top}. \\
\frac{\partial \mathbf{z}_{:i}^{(t+1)}}{\partial \mathbf{z}_{:i}^{(t)}} &= \frac{\partial \mathbf{z}_{:i}^{(t+1)}}{\partial \mathbf{h}_{:i}^{(t)}} \frac{\partial \mathbf{h}_{:i}^{(t)}}{\partial \mathbf{z}_{:i}^{(t)}} = \mathbf{w}^{(t+1)} \mathrm{Diag}\left(\mathrm{ReLU}'\left(\mathbf{z}_{:i}^{(t)}\right)\right).
\end{aligned}$$

Here, $\mathrm{ReLU}'(\cdot)$ is the derivative of ReLU function. We will give its formal definition later. $\mathrm{Diag}(\cdot)$ is a diagnal matrix of which the main diagnal is the given input vector. Therefore, we can derive

$$\begin{aligned}
\phi(t,i) &= \phi(t+1,i) \frac{\partial \mathbf{z}_{:i}^{(t+1)}}{\partial \mathbf{z}_{:i}^{(t)}} \\
&= \phi(t+1,i) \mathbf{w}^{(t+1)} \mathrm{Diag}\left(\mathrm{ReLU}'\left(\mathbf{z}_{:i}^{(t)}\right)\right) \\
&= \left(\phi(t+1,i)\mathbf{w}^{(t+1)}\right) \odot \mathrm{ReLU}'\left(\mathbf{z}_{:i}^{(t)}\right)^{\top}
\end{aligned}$$

Here, $\odot$ denotes element-wise product for two column/row vectors. For $\frac{\partial \mathbf{z}_{:i}^{(t)}}{\partial {\mathbf{w}_{k:}^{(t)}}^{\top}}$, it is a $d^{(t)} \times d^{(t)}$ matrix. We denote it as $\mathbf{X}$.

$$\frac{\partial \mathbf{z}_{:i}^{(t)}}{\partial {\mathbf{w}_{k:}^{(t)}}^{\top}} = \mathbf{X},$$

$$\text{where } \mathbf{X}_{j:} = \begin{cases} {\mathbf{h}_{:i}^{(t-1)}}^{\top} & \text{if } j = k, \\ \mathbf{0} & \text{otherwise.} \end{cases}$$

Therefore, we can derive

$$\psi(t,i,k) = \phi(t,i)\mathbf{X} = \phi(t,i)_m \cdot \mathbf{X}_{k:} = \phi(t,i)_m \cdot {\mathbf{h}_{:i}^{(t-1)}}^{\top}.$$

Since $\psi(t,i,k)$ is for the k-th row of $\mathbf{w}^{(t)}$, for the complete $\mathbf{w}^{(t)}$, we have

$$\frac{\partial L_i}{\partial \mathbf{w}^{(t)}} = \psi(t,i) = \left[\psi(t,i,1)^\top, \psi(t,i,2)^\top, \dots, \psi(t,i,d^{(t)})^\top\right] = \mathbf{h}_{:i}^{(t-1)}\phi(t,i).$$

Here, we have already got the gradient for one sample. For all samples, we need to rewrite the $\phi(t,i)$ and $\psi(t,i)$ to a matrix form $\phi(t)$ and $\psi(t)$:

$$\frac{\partial L}{\partial \mathbf{z}^{(T)}} = \phi(T) = (\hat{\mathbf{y}} - \mathbf{y})^\top \in \mathbb{R}^{n\times C}, \tag{11}$$

$$\frac{\partial L}{\partial \mathbf{z}^{(t)}} = \phi(t) = \left(\phi(t+1)\mathbf{w}^{(t+1)}\right) \odot \mathrm{ReLU}'\left(\mathbf{z}^{(t)}\right)^\top \in \mathbb{R}^{n\times d^{(t)}}, \tag{12}$$

$$\frac{\partial L}{\partial \mathbf{w}^{(t)}} = \psi(t) = \frac{1}{n}\sum_{i=1}^{n}\psi(t,i) = \frac{1}{n}\mathbf{h}^{(t-1)}\phi(t) \in \mathbb{R}^{d^{(t-1)}\times d^{(t)}}. \tag{13}$$

Similarly, the gradient of $\mathbf{b}^{(t)}$ can be calculated as:

$$\frac{\partial L}{\partial \mathbf{b}^{(t)}} = \frac{1}{n}\sum_{i=1}^{n}\phi(t,i) \in \mathbb{R}^{1\times d^{(t)}}. \tag{14}$$

Equations (11)-(14) are the final gradients for MLP. For the dimension of matrix derivative, please refer to Matrix Calculus at Wikipedia.

Recap. We may notice that the gradients of the output layer ($t = T$) of MLP is similar to the logistic regression. In fact, the binary classification is a special form of multi-class classification, and logistic regression can be regraded as a single-layer MLP. In practice, the label of binary classification is usually 0 or 1, while the label of multi-class classification is a one-hot vector. This is the reason why we distinguish binary classification from multi-class classification.

After getting the gredients of weights and bias in MLP, there is still a derivative of ReLU activation function to be solved. The ReLU function is defined as follows:

$$\mathrm{ReLU}\left(x\right) = \begin{cases} 0 & \text{if } x < 0, \\ x & \text{otherwise,} \end{cases} = \max\{0, x\}. \tag{15}$$

It is a convex function that is sub-differential when $x \neq 0$. However, it is undifferentiable at $x = 0$. Therefore, we use subgradient of ReLU and set the gradient as 0 at $x = 0$:

$$\mathrm{ReLU}'\left(x\right) = \begin{cases} 0 & \text{if } x \leq 0, \\ 1 & \text{otherwise.} \end{cases} \tag{16}$$