# BP Derivation for MLP and CNN

N4A

December 14, 2018

## 1 Task description

Please derive a backpropagation process

(1) for the multi-layer neural network with one hidden layer, where data are in a m-dimensional feature space with n classes. Loss functions can use L2 distance or cross entropy.

(2) for the LeNet-5 CNN.

## 2 For Multilayer neural network with one hidden layer [1]

As shown in Figure 1, we denote input layer as $x$ and $x_k$ represents the value of the k-th unit of input layer. we denote $a(.)$ as the activation function, $o^{(1)}$ as the activation of the hidden layer, and $\{W^{(1)}, b^{(1)}\}$ as the weights and bias from input layer to hidden layer, then the hidden layer, denoted as $h$, is calculated as follows:

$$o^{(1)} = W^{(1)}x + b^{(1)} \tag{1}$$

$$h = a(o^{(1)}) \tag{2}$$

As the same as above, we denote output layer as a vector $y$, the activation as $o^{(2)}$, and the weights and bias from hidden layer to output layer as $\{W^{(2)}, b^{(2)}\}$, then the output vector is calculated as follows:

$$o^{(2)} = W^{(2)}h + b^{(2)} \tag{3}$$

$$y = a(o^{(2)}) \tag{4}$$

Finally, we denote the true target value as a vector $t$ and the error function as $L(t, y)$.

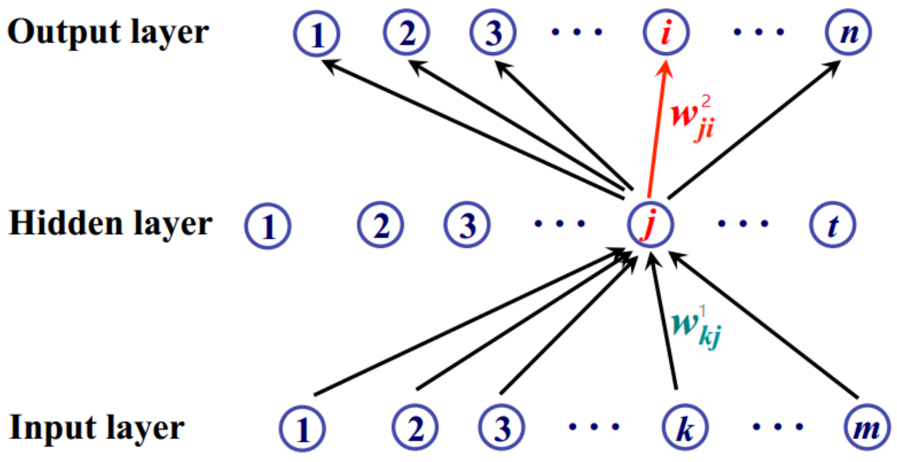

Figure 1: Multilayer neural network with one hidden layer

## 2.1 General back-propagation

Then, in the back-propagation process, the gradients of each weight and bias from can be estimate using chain derivation rule as follows:

$$\begin{aligned}\frac{\partial L(t,y)}{\partial w_{ji}^{(2)}} &= \frac{\partial L(t,y)}{\partial y_i}\frac{\partial y_i}{\partial o_i^{(2)}}\frac{\partial o_i^{(2)}}{\partial w_{ji}^{(2)}} \\ &= \frac{\partial L(t,y)}{\partial y_i}\frac{\partial y_i}{\partial o_i^{(2)}}h_j\end{aligned} \tag{5}$$

$$\begin{aligned}\frac{\partial L(t,y)}{\partial b_j^{(2)}} &= \frac{\partial L(t,y)}{\partial y_i}\frac{\partial y_i}{\partial o_i^{(2)}}\frac{\partial o_i^{(2)}}{\partial b_j^{(2)}} \\ &= \frac{\partial L(t,y)}{\partial y_i}\frac{\partial y_i}{\partial o_i^{(2)}}\end{aligned} \tag{6}$$

$$\begin{aligned}\frac{\partial L(t,y)}{\partial w_{kj}^{(1)}} &= \sum_i \frac{\partial L(t,y)}{\partial y_i}\frac{\partial y_i}{\partial o_i^{(2)}}\frac{\partial o_i^{(2)}}{\partial h_j}\frac{\partial h_j}{\partial o_j^{(1)}}\frac{\partial o_j^{(1)}}{\partial w_{kj}^{(1)}} \\ &= \sum_i \frac{\partial L(t,y)}{\partial y_i}\frac{\partial y_i}{\partial o_i^{(2)}}w_{ji}^{(2)}\frac{\partial h_j}{\partial o_j^{(1)}}x_k\end{aligned} \tag{7}$$

$$\begin{aligned}\frac{\partial L(t,y)}{\partial b_k^{(1)}} &= \sum_i \frac{\partial L(t,y)}{\partial y_i}\frac{\partial y_i}{\partial o_i^{(2)}}\frac{\partial o_i^{(2)}}{\partial h_j}\frac{\partial h_j}{\partial o_j^{(1)}}\frac{\partial o_j^{(1)}}{\partial b_k^{(1)}} \\ &= \sum_i \frac{\partial L(t,y)}{\partial y_i}\frac{\partial y_i}{\partial o_i^{(2)}}w_{ji}^{(2)}\frac{\partial h_j}{\partial o_j^{(1)}}\end{aligned} \tag{8}$$

## 2.2 L2 difference loss

For the l2 difference loss function defined as follows,

$$L(t,y) = \frac{1}{2}\sum_i (t_i - y_i)^2 \tag{9}$$

$$\frac{\partial L(t,y)}{\partial y_i} = -(t_i - y_i) = (y_i - t_i) \tag{10}$$

we substitute $\frac{\partial L(t,y)}{\partial y_i}$ in the equations from 5 to 8 with $(y_i - t_i)$. Then we can get

$$\frac{\partial L(t,y)}{\partial w_{ji}^{(2)}} = (y_i - t_i)\frac{\partial y_i}{\partial o_i^{(2)}}h_j \tag{11}$$

$$\frac{\partial L(t,y)}{\partial b_j^{(2)}} = (y_i - t_i)\frac{\partial y_i}{\partial o_i^{(2)}} \tag{12}$$

$$\frac{\partial L(t,y)}{\partial w_{kj}^{(1)}} = \sum_i (y_i - t_i)\frac{\partial y_i}{\partial o_i^{(2)}}w_{ji}^{(2)}\frac{\partial h_j}{\partial o_j^{(1)}}x_k \tag{13}$$

$$\frac{\partial L(t,y)}{\partial b_k^{(1)}} = \sum_i (y_i - t_i)\frac{\partial y_i}{\partial o_i^{(2)}}w_{ji}^{(2)}\frac{\partial h_j}{\partial o_j^{(1)}} \tag{14}$$

## 2.3 Cross entropy loss

For the cross entropy loss function defined as follows,

$$L(t,y) = \sum_i -t_i \log y_i \tag{15}$$

$$\frac{\partial L(t,y)}{\partial y_i} = -\frac{t_i}{y_i} \tag{16}$$

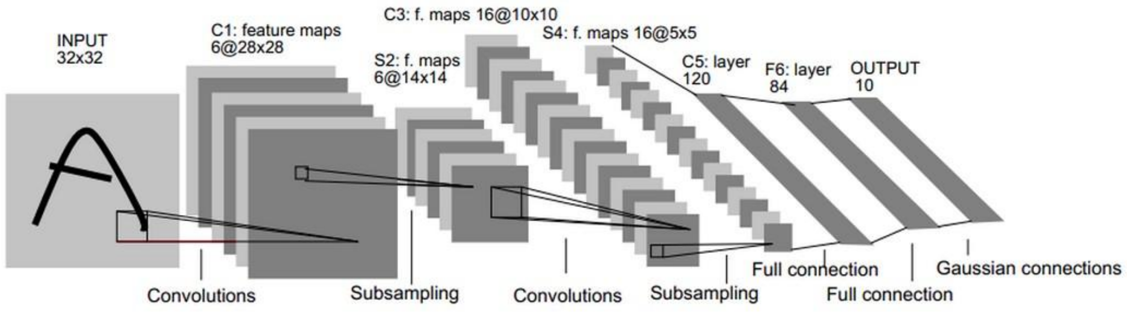

Figure 2: The overall architecture of LeNet-5

we substitute $\frac{\partial L(t,y)}{\partial y_i}$ in the equations from 5 to 8 with $-\frac{t_i}{y_i}$. Then we can get

$$\frac{\partial L(t,y)}{\partial w_{ji}^{(2)}} = -\frac{t_i}{y_i}\frac{\partial y_i}{\partial o_i^{(2)}}h_j \tag{17}$$

$$\frac{\partial L(t,y)}{\partial b_j^{(2)}} = -\frac{t_i}{y_i}\frac{\partial y_i}{\partial o_i^{(2)}} \tag{18}$$

$$\frac{\partial L(t,y)}{\partial w_{kj}^{(1)}} = \sum_i -\frac{t_i}{y_i}\frac{\partial y_i}{\partial o_i^{(2)}}w_{ji}^{(2)}\frac{\partial h_j}{\partial o_j^{(1)}}x_k \tag{19}$$

$$\frac{\partial L(t,y)}{\partial b_k^{(1)}} = \sum_i -\frac{t_i}{y_i}\frac{\partial y_i}{\partial o_i^{(2)}}w_{ji}^{(2)}\frac{\partial h_j}{\partial o_j^{(1)}} \tag{20}$$

### 2.4 Sigmoid activation function

If the activation function $a(.)$ is the sigmoid function defined as follows,

$$a(x) = \frac{1}{1 + exp^{-x}} \tag{21}$$

$$\frac{\partial a(x)}{\partial x} = a(x)(1 - a(x)) \tag{22}$$

then the corresponding $\frac{\partial a(x)}{\partial x}$ can be substituted with $a(x)(1 - a(x))$ as the same as above.

## 3 For the LeNet-5 CNN [3, 2]

As shown in Figure 2, LetNet-5 consists of three different kind of operations: convolution, pooling and full connection layer. The full connection layer is just the network operation we introduce in section 2. So here we take a look at what are convolution and pooling operation at first.

### 3.1 Convolution in CNN

In Mathematics, for discrete variables, Convolution function is defined as follows:

$$(I * K)_{ij} = \sum_{m=0}^{k_1-1}\sum_{n=0}^{k_2-1} I(m,n)K(i-m, j-n) \tag{23}$$

Convolution is commutative, meaning that we can equivalently write:

$$\begin{aligned}(I * K)_{ij} &= \sum_{m=0}^{k_1-1}\sum_{n=0}^{k_2-1} I(i-m, j-n)K(m,n) \\ &= \sum_{m=0}^{k_1-1}\sum_{n=0}^{k_2-1} I(i+m, j+n)K(-m,-n)\end{aligned} \tag{24}$$

Usually, the latter format is more straightforward to implement in a machine learning model. We substitute kernel $K(-m,-n)$ with a flipped kernel $K(m,n)$, then we get the cross correlation function:

$$(I \otimes K)_{ij} = \sum_{m=0}^{k_1-1}\sum_{n=0}^{k_2-1} I(i+m, j+n)K(m,n) \tag{25}$$

Because in CNN, kernel function K is a matrix with parameters to be trained, the cross correlation function could get the same result as convolution function with a flipped kernel matrix K. For cross correlation function is more straightforward to implement, we use it implement convolution layer in CNN.

In the case of images, we could have as input an image with height $H$, width $W$ and $C = 3$ channels (red, blue and green) such that $I \in R^{H\times W\times C}$. The number of output channels id $D$. Subsequently for a bank of $D$ filters, we have $K \in R^{k_1\times k_2\times C\times D}$ and biases $b \in R^D$, one for each filter. The output from this convolution procedure is as follows:

$$(I * K_d)_{ij} = \sum_{m=0}^{k_1-1}\sum_{n=0}^{k_2-1}\sum_{c=1}^{C} K_{m,n,c,d}\cdot I_{i+m,j+n,c} + b_d \tag{26}$$

We denote the activation function as $a(.)$, the activation of the l-th convolution layer as $x^l_{ij}$, and the output vector of the l-th convolution layer as $o^l$, then,

$$x^1_{ijd} = (I * K_d)_{ij} \tag{27}$$

$$o^1_{ijd} = a(x^1_{ijd}) \tag{28}$$

The following convolution layer is just the same as the first with the input being the output of in the previous layer in Figure 2.

## 3.2 Pooling

In LeNet-5, there is a (max) pooling layer each convolution layer. At the pooling layer, forward propagation results in an $N \times N$ pooling block being reduce to a single value. For Max-pooling, the value is the biggest value in the pooling block. For Average pooling, the value is the average of all values in the pooling block. We denote the output of l-th pooling layer as $p^l$, then take max-pooling for an example, we get

$$p^l_{ijd} = \max\{o^l_{mnd} | iN \le m < (i+1)N, jN \le n < (j+1)N\} \tag{29}$$

For average pooling, we just need substitute operation *max* with operation *avg* which mean get the average of the all values in the set.

## 3.3 Back propagation for each kind of layer

Convolution between the input feature map of dimension $H \times W$ with C channels and the weight kernel of dimension $k_1 \times k_2$ produces an output feature map of size $(H - k_1 + 1) \times (W - k_2 + 1)$. The number of input channels and output channels are $C$ and $D$. The gradient component for the individual weights can be obtained by applying the chain rule in the following way:

$$\begin{aligned}
\frac{\partial L(t,y)}{\partial w^l_{m',n',c,d}} &= \sum_{i=0}^{H-k_1}\sum_{j=0}^{W-k_2} \frac{\partial L(t,y)}{\partial x^l_{i,j,d}} \frac{\partial x^l_{i,j,d}}{\partial w^l_{m',n',c,d}} \\
&= \sum_{i=0}^{H-k_1}\sum_{j=0}^{W-k_2} \delta^l_{i,j,d} \frac{\partial x^l_{i,j,d}}{\partial w^l_{m',n',c,d}} \\
&= \sum_{i=0}^{H-k_1}\sum_{j=0}^{W-k_2} \delta^l_{i,j,d} \frac{\partial}{\partial w^l_{m',n',c,d}} \left(\sum_m\sum_n\sum_{c=1}^{C} w^l_{m,n,c,d} o^{l-1}_{i+m,j+n,c} + b^l_d\right) \\
&= \sum_{i=0}^{H-k_1}\sum_{j=0}^{W-k_2} \delta^l_{i,j,d} \frac{\partial}{\partial w^l_{m',n',c,d}} \left(w^l_{m',n',c,d} o^{l-1}_{i+m',j+n',c}\right) \\
&= \sum_{i=0}^{H-k_1}\sum_{j=0}^{W-k_2} \delta^l_{i,j,d} o^{l-1}_{i+m',j+n',c}
\end{aligned} \tag{30}$$

$$
\begin{aligned}
\frac{\partial L(t,y)}{\partial b_d^l} &= \sum_{i=0}^{H-k_1} \sum_{j=0}^{W-k_2} \frac{\partial L(t,y)}{\partial x_{i,j,d}^l} \frac{\partial x_{i,j,d}^l}{\partial b_d^l} \\
&= \sum_{i=0}^{H-k_1} \sum_{j=0}^{W-k_2} \delta_{i,j,d}^l \frac{\partial x_{i,j,d}^l}{\partial b_d^l} \\
&= \sum_{i=0}^{H-k_1} \sum_{j=0}^{W-k_2} \delta_{i,j,d}^l \frac{\partial}{\partial b_d^l} \left( \sum_m \sum_n \sum_{c=1}^{C} w_{m,n,c,d}^l o_{i+m,j+n,c}^{l-1} + b_d^l \right) \\
&= \sum_{i=0}^{H-k_1} \sum_{j=0}^{W-k_2} \delta_{i,j,d}^l
\end{aligned}
\tag{31}
$$

where we denote $\frac{\partial L(t,y)}{\partial x_{i,j,d}^l}$ as $\delta_{i,j,d}^l$ that represents the error in layer $l$.

There is no parameter in pooling layer. And the Gradient rooting is done in the following ways.

1. Max-pooling - the error is just assigned to where it comes from - the "winning unit" because other units in the previous layer's pooling blocks did not contribute to it hence all the other assigned values of zero
2. Average pooling - the error is multiplied by 1/N×N and assigned to the whole pooling block (all units get this same value).

## 3.4 Back propagation in LeNet-5

LeNet-5 includes full connection layer, pooling layer and convolution layer. Using chain rule and gradients in section 2 and 3.3, we can easily get the gradients of all parameters in LeNet-5.

## References

[1] Christopher Bishop. Pattern recognition and machine learning.

[2] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[3] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.