Chain Rule of Neural Network is Error Back Propagation

Masayuki Tanaka

2013/11/20

1 Introduction

When I studied the convolution network, I could not figure out how to impliment an error backpropagation for the Lp pooling. I had found there was a few information about that. Through a discussion with Hiroshi Kuwajima, I finally figure out the error backpropagation is a simple chain rule of the derivatives. This framework is very general and easy to understand, but again there was a few explanatoin of the error backpropagation as the simple chain rule.

2 Notations

Here, I define some notations.

The elementwise sigmoid function is defined by

$$\boldsymbol{y} = \boldsymbol{\sigma}(\boldsymbol{x}) = \begin{bmatrix} \sigma(x_1) \\ \sigma(x_2) \\ \vdots \\ \sigma(x_N) \end{bmatrix}, \tag{1}$$

where

$$\sigma(x) = \frac{1}{1+e^{-x}}. \tag{2}$$

I use the symbol, $\circ$, for the compositional operation of functions. I also introduce the blackets, [], to specify the parameters differentiating from the arguments of the function.

$$\boldsymbol{f}(\boldsymbol{x};\boldsymbol{\theta}) = \boldsymbol{f}[\boldsymbol{\theta}](\boldsymbol{x}), \tag{3}$$

$$\boldsymbol{g}(\boldsymbol{f}(\boldsymbol{x})) = \boldsymbol{g} \circ \boldsymbol{f} \circ \boldsymbol{x}, \tag{4}$$

$$\boldsymbol{g}[\boldsymbol{\tau}](\boldsymbol{f}[\boldsymbol{\theta}](\boldsymbol{x})) = \boldsymbol{g}[\boldsymbol{\tau}] \circ \boldsymbol{f}[\boldsymbol{\theta}] \circ \boldsymbol{x}. \tag{5}$$

The one-layer neural network with a sigmoid activation function can be expressed by

$$\boldsymbol{\sigma}(\boldsymbol{W}\boldsymbol{x}) = \boldsymbol{\sigma} \circ \boldsymbol{\rho}[\boldsymbol{W}] \circ \boldsymbol{x}, \tag{6}$$

where

$$\boldsymbol{\rho}[\boldsymbol{W}](\boldsymbol{x}) = \boldsymbol{\rho}[\boldsymbol{W}] \circ \boldsymbol{x} = \boldsymbol{W}\boldsymbol{x}, \tag{7}$$

where $\boldsymbol{\rho}[\boldsymbol{W}]$ represents the weighted sum or the linear transformation, $\boldsymbol{W}$ is the weight or the parameters, and $\boldsymbol{x}$ is the input of the neural network. Here, I ommit to put the bias term to simplify the expression.

The derivatives of the vectors are defined as

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_M} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_M} \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_N}{\partial x_1} & \frac{\partial f_N}{\partial x_2} & \cdots & \frac{\partial f_N}{\partial x_M} \end{bmatrix} \tag{8}$$

3 Chain rule of the Derivatives

The chain rules of the derivatives can be expressed as

$$\frac{\partial}{\partial \boldsymbol{x}}(L \circ \boldsymbol{g}[\boldsymbol{\tau}] \circ \boldsymbol{f}[\boldsymbol{\theta}] \circ \boldsymbol{x}) = \frac{\partial L}{\partial \boldsymbol{g}} \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{f}} \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} . \tag{9}$$

The chain rule with respect to the parameter can be derived very similary as

$$\frac{\partial}{\partial \boldsymbol{\theta}}(L \circ \boldsymbol{g}[\boldsymbol{\tau}] \circ \boldsymbol{f}[\boldsymbol{\theta}] \circ \boldsymbol{x}) = \frac{\partial L}{\partial \boldsymbol{g}} \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{f}} \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\theta}} . \tag{10}$$

4 Chain Rule of Neural Network (Error Back Propagation)

4.1 Sigmoid Activation

The one-layer neural network can be expresses by the compositional form of the sigmoid activation and the linear transformation. Namely,

$$\boldsymbol{y} = \boldsymbol{\sigma}(\boldsymbol{W}\boldsymbol{x}) = \boldsymbol{\sigma}(\rho[\boldsymbol{W}](\boldsymbol{x})) = \sigma \circ \rho[\boldsymbol{W}] \circ \boldsymbol{x} , \tag{11}$$

where

$$\rho[\boldsymbol{W}] = \boldsymbol{W}\boldsymbol{x} . \tag{12}$$

Here, I ommit the bias term for the simplification.

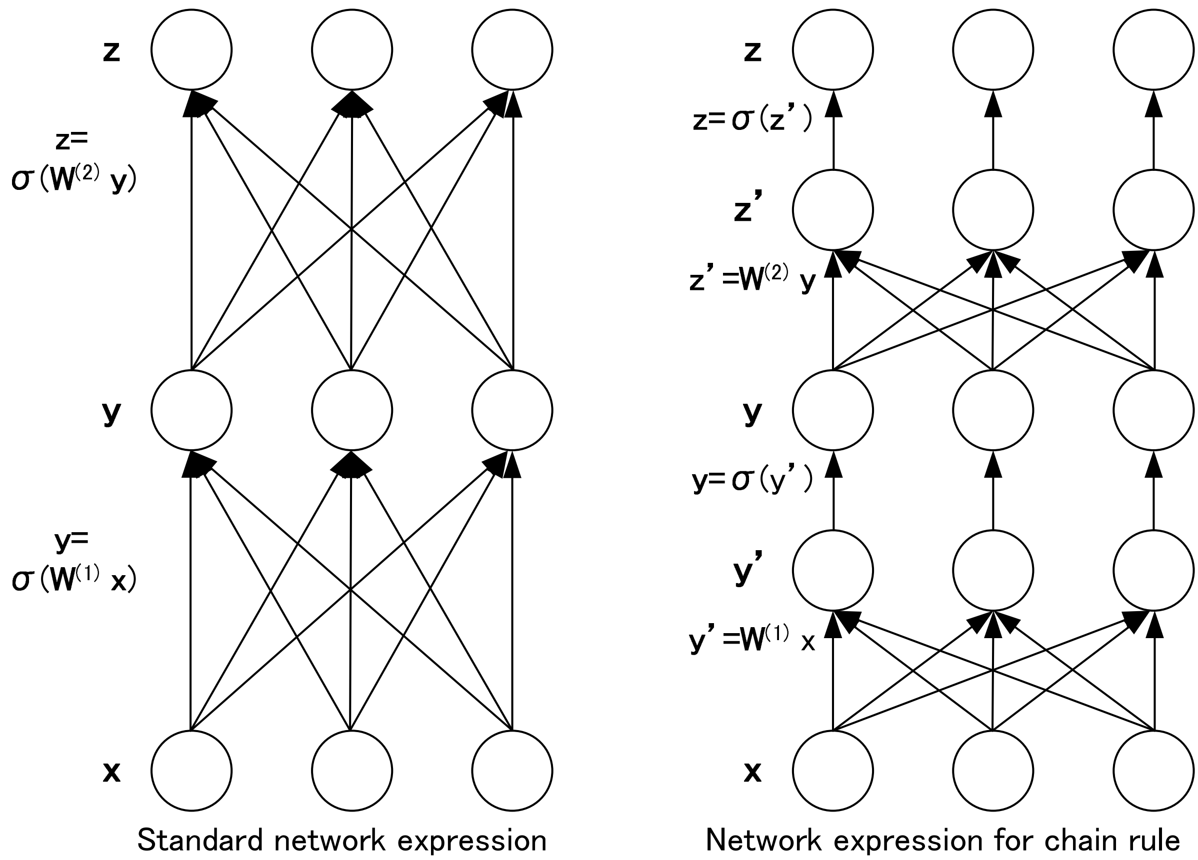

Figure 1: The two expressions of the two-layer neural network.

Using these notation, the two-layer neural nerwork as shown in Fig. 1 can be expressed by

$$\boldsymbol{z} = \boldsymbol{\sigma} \circ \rho[\boldsymbol{W}^{(2)}] \circ \boldsymbol{\sigma} \circ \rho[\boldsymbol{W}^{(1)}] \circ \boldsymbol{x} , \tag{13}$$

where hidden variables are

$$\begin{aligned}
\boldsymbol{z} &= \boldsymbol{\sigma} \circ \boldsymbol{z}' , & (14)\\
\boldsymbol{z}' &= \rho[\boldsymbol{W}^{(2)}] \circ \boldsymbol{y} , & (15)\\
\boldsymbol{y} &= \boldsymbol{\sigma} \circ \boldsymbol{y}' , & (16)\\
\boldsymbol{y}' &= \rho[\boldsymbol{W}^{(1)}] \circ \boldsymbol{x} . & (17)
\end{aligned}$$

The cost function is defined like the square error. Namely,

$$L = \sum_k L_k \text{ where, } L_k = ||\boldsymbol{t}_k - \boldsymbol{z}_k(\boldsymbol{x}_k)||_2^2 \tag{18}$$

where $\boldsymbol{t}_k$ is the teach data. The training of the network is perfomed by minimizing the cost function with respect to parameters, $\{\boldsymbol{W}^{(1)}, \boldsymbol{W}^{(2)}\}$. The chain rule is independent to the cost function. One can consider any kind of the cost function such as a cross entropy.

Now, it is ready to derive the derivatives of L_k with respect to the parameters by using the chain rule. Once one can derive the derivatives of L_k, one can obtain the derivatives of L by simply summing up.

$$\begin{aligned}
\frac{\partial L_k}{\partial \boldsymbol{W}^{(2)}} &= \frac{\partial L_k}{\partial \boldsymbol{z}} \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{z}'} \frac{\partial \boldsymbol{z}'}{\partial \boldsymbol{W}^{(2)}} = \boldsymbol{\delta}_{\boldsymbol{z}'} \frac{\partial \boldsymbol{z}'}{\partial \boldsymbol{W}^{(2)}} & (19)\\
\frac{\partial L_k}{\partial \boldsymbol{W}^{(1)}} &= \frac{\partial L_k}{\partial \boldsymbol{z}} \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{z}'} \frac{\partial \boldsymbol{z}'}{\partial \boldsymbol{y}} \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{y}'} \frac{\partial \boldsymbol{y}'}{\partial \boldsymbol{W}^{(1)}} = \boldsymbol{\delta}_{\boldsymbol{y}'} \frac{\partial \boldsymbol{y}'}{\partial \boldsymbol{W}^{(1)}} & (20)\\
\boldsymbol{\delta}_{\boldsymbol{z}'} &= \frac{\partial L_k}{\partial \boldsymbol{z}} \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{z}'} & (21)\\
\boldsymbol{\delta}_{\boldsymbol{y}'} &= \frac{\partial L_k}{\partial \boldsymbol{z}} \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{z}'} \frac{\partial \boldsymbol{z}'}{\partial \boldsymbol{y}} \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{y}'} & (22)
\end{aligned}$$

The "error" of $\boldsymbol{\delta}_{\boldsymbol{z}'}$ and $\boldsymbol{\delta}_{\boldsymbol{y}'}$ can be propageted layer-by-layer backwardly.

$$\begin{aligned}
\boldsymbol{\delta}_{\boldsymbol{z}} &= \frac{\partial L_k}{\partial \boldsymbol{z}} & (23)\\
\boldsymbol{\delta}_{\boldsymbol{z}'} &= \frac{\partial L_k}{\partial \boldsymbol{z}} = \boldsymbol{\delta}_{\boldsymbol{z}} \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{z}'} = \frac{\partial L_k}{\partial \boldsymbol{z}} \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{z}'} & (24)\\
\boldsymbol{\delta}_{\boldsymbol{y}} &= \frac{\partial L_k}{\partial \boldsymbol{y}} = \boldsymbol{\delta}_{\boldsymbol{z}'} \frac{\partial \boldsymbol{z}'}{\partial \boldsymbol{y}} = \frac{\partial L_k}{\partial \boldsymbol{z}} \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{z}'} \frac{\partial \boldsymbol{z}'}{\partial \boldsymbol{y}} & (25)\\
\boldsymbol{\delta}_{\boldsymbol{y}'} &= \frac{\partial L_k}{\partial \boldsymbol{y}'} = \boldsymbol{\delta}_{\boldsymbol{y}} \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{y}'} = \frac{\partial L_k}{\partial \boldsymbol{z}} \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{z}'} \frac{\partial \boldsymbol{z}'}{\partial \boldsymbol{y}} \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{y}'} & (26)
\end{aligned}$$

This chain of chcalculations are called an error back propagation.

Each derivatives are calcularated as follows:

$$\begin{aligned}
\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{z}'} &= \boldsymbol{z}' \otimes (\boldsymbol{1} - \boldsymbol{z}') & (27)\\
\frac{\partial \boldsymbol{z}'}{\partial \boldsymbol{y}} &= \boldsymbol{W}^{(2)} & (28)\\
\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{y}'} &= \boldsymbol{y}' \otimes (\boldsymbol{1} - \boldsymbol{y}') & (29)\\
\frac{\partial \boldsymbol{z}'}{\partial \boldsymbol{W}^{(2)}} &= \boldsymbol{y}^T & (30)\\
\frac{\partial \boldsymbol{y}'}{\partial \boldsymbol{W}^{(1)}} &= \boldsymbol{x}^T & (31)
\end{aligned}$$

where $\otimes$ is the elementwise multiplication, $\boldsymbol{1}$ is the column vector whose element is one, and $\boldsymbol{x}^T$ represents the transpose vector of the vector $\boldsymbol{x}$.

4.2 Lp Pooling

A lp pooling is a technique of subsampling which is usually used in a convolutional network. The lp pooling and its derivatives can be expressed by

$$z = \eta_p(\boldsymbol{y}) = \left(\sum_i |y_i|^p\right)^{1/p} , \tag{32}$$

$$\frac{\partial z}{\partial y_i} = \left(\sum_i |y_i|^p\right)^{1/p-1} |y_i|^{p-1} . \tag{33}$$

Here, I derive the error back propagation, or the chain rule, of the neural network which includes the lp pooling as shown in Fig. 2.

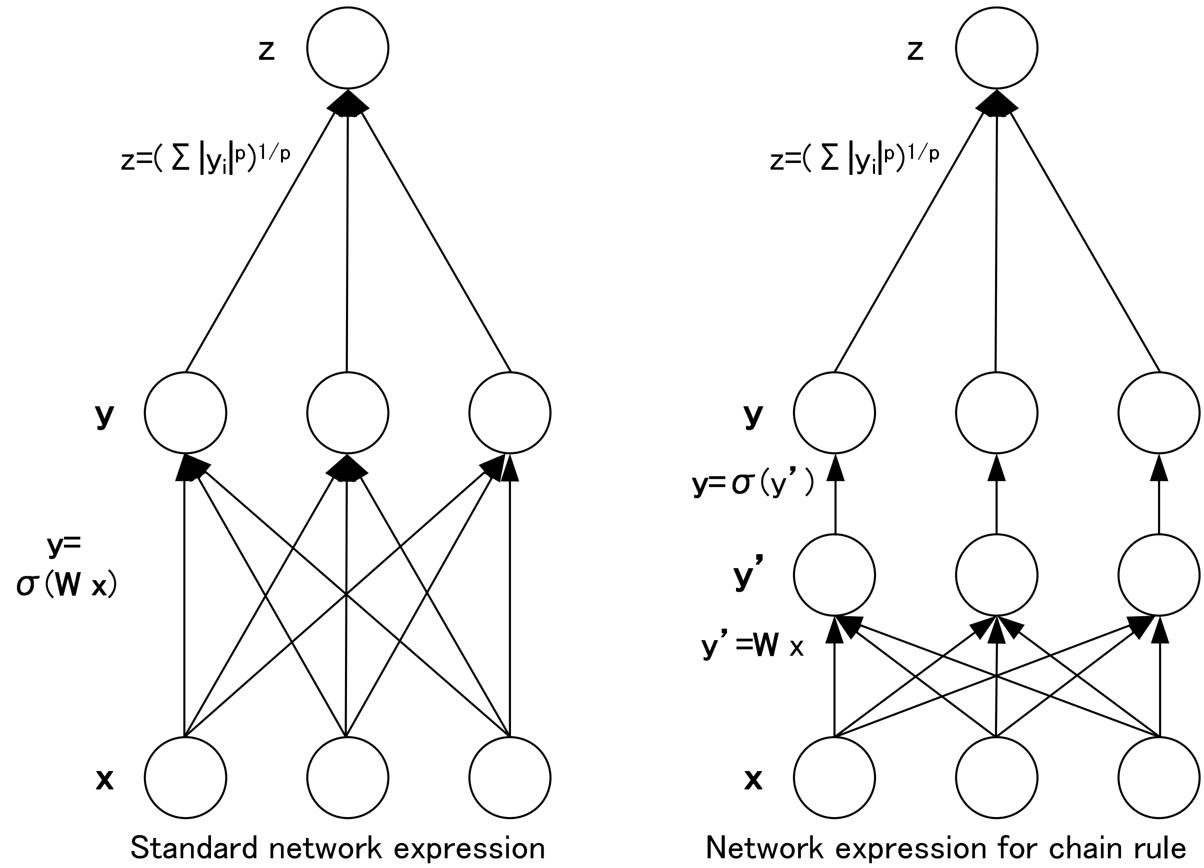

Figure 2: The two expressions of the neural network which includes the lp pooling.

The neural network can be expressed by

$$z = \eta_p \circ \boldsymbol{\sigma} \circ \rho[\boldsymbol{W}] \circ \boldsymbol{x} \tag{34}$$

The hidden variables are

$$z = \eta_p \circ \boldsymbol{y} , \tag{35}$$

$$\boldsymbol{y} = \boldsymbol{\sigma} \circ \boldsymbol{y}' , \tag{36}$$

$$\boldsymbol{y}' = \rho[\boldsymbol{W}] \circ \boldsymbol{x} . \tag{37}$$

The error back propagation described in Section 4.1 is general. One can apply to the lp pooling as well.

$$\boldsymbol{\delta}_z = \frac{\partial L_k}{\partial z} \tag{38}$$

$$\boldsymbol{\delta}_{\boldsymbol{y}} = \frac{\partial L_k}{\partial \boldsymbol{y}} = \boldsymbol{\delta}_z \frac{\partial z}{\partial \boldsymbol{y}} = \frac{\partial L_k}{\partial z} \frac{\partial z}{\partial \boldsymbol{y}} \tag{39}$$

$$\boldsymbol{\delta}_{\boldsymbol{y}'} = \frac{\partial L_k}{\partial \boldsymbol{y}'} = \boldsymbol{\delta}_{\boldsymbol{y}} \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{y}'} = \frac{\partial L_k}{\partial z} \frac{\partial z}{\partial \boldsymbol{y}} \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{y}'} \tag{40}$$

Then, one can calculate the derivative of the cost function with respect to the parameters $\boldsymbol{W}$ as

$$\frac{\partial L_k}{\partial \boldsymbol{W}} = \boldsymbol{\delta}_{\boldsymbol{y}'} \frac{\partial \boldsymbol{y}'}{\partial \boldsymbol{W}} = \boldsymbol{\delta}_{\boldsymbol{y}'} \boldsymbol{x}^T . \tag{41}$$

Acknowledgement

Thank you for good discussion with Hiroshi Kuwajima.

References

[1] UFLDL Tutorial, http://ufldl.stanford.edu/wiki/index.php/UFLDL_Tutorial