Lecture Notes On Back Propagation Technique for CNNs (Part I)

Harry Li [‡], Ph.D.
Computer Engineering Department
College of Engineering, San Jose State University
San Jose, CA 95192, USA

Research and Development Center for Artificial Intelligence and Automation [‡]
CTI Plus Corporation, 3679 Enochs Street
Santa Clara, CA 95051, USA

Email[†]: hua.li@sjsu.edu

***Abstract*—This note is for my lecture on convolutional neural network training techniques. In particular, the back propagation techniques for CNN facial recognitions.**

I. Back Propagation Technique

In this note, mathematical formulation of back propagation technique is discussed below.

First, consider notation for a simple feed forward neural network, define input neuron $\overrightarrow{\sigma}$ and output neuron $\overrightarrow{S}$, so we have

Definition 1. *Define input and output neurons for a neural network, the input neuros*

$$\overrightarrow{\sigma} = \begin{pmatrix} \sigma_1 \\ \sigma_2 \\ . \\ . \\ . \\ \sigma_n \end{pmatrix} \tag{1}$$

and the output neurons

$$\overrightarrow{S} = \begin{pmatrix} s_1 \\ s_2 \\ . \\ . \\ . \\ s_{n_o} \end{pmatrix} \tag{2}$$

and weights $w_{i,k}$, where i for output neuron s_i, and k for input neuron σ_k, (insert Figure 1 here), hence we have an activation function below.

Fig. 1. NN structure with input and output neurons.

Definition 2. *Define an activation function as*

$$s_i = f(w_{i,k} \cdot \sigma_k) \tag{3}$$

$$s_i = f(\sum_{i=1}^{n_o} w_{i,k} \cdot s_k + \phi) = f(\sum_{i=1}^{n_o+1} w_{i,k} \cdot s_k) \tag{4}$$

Note in case of convolutional NN, the nerons σ_k in the above equation is the output of 2D convolution layer. Denote 2D convolution as

$$\sigma_k = g_{\sigma_k}(l_i(x,y) * K(x,y); k_{u,v}) \tag{5}$$

where $l_i(x,y)$ is convolution layer i, $K(x,y)$ is a convolution kernel, whose coefficients are u and v such as $((u,v) \in \omega)$, e.g., in a k-by-k patch, k are odd numbers.

Therefore, equation (4) can be updated as follows for one output neuron

$$s_i = f(w_{i,k} \cdot g_{\sigma_k}(l_i(x,y) * K(x,y); k_{u,v}|u,v \in \omega)) \tag{6}$$

and for all output neuron n_o at that layer with summation

$$s_i = f(\sum_{i=1}^{n_o+1} w_{i,k} \cdot g_{\sigma_k}(l_i(x,y) * K(x,y); k_{u,v}|u,v \in \omega)) \tag{7}$$

or in a simplified notation,

$$s_i = f(\sum_{i=1}^{n_o+1} w_{i,k} \cdot g_{\sigma_k}(l_i * K)) \tag{8}$$

Exampel 1. *For a simple NN with 2 inputs and 1 output, as given in my hand written note for the case of facial recognition for two person classes, we have input*

$$\overrightarrow{\sigma} = \begin{pmatrix} \sigma_1 \\ \sigma_2 \end{pmatrix} \tag{9}$$

and output vector

$$\overrightarrow{s} = (s_1) \tag{10}$$

and weights

$$w_{i,k} \tag{11}$$

for output index i = 1, and input index k = 1, 2. So, we have inputs (σ_1, σ_2), for example from the given simple two persons (classes) case, have two classes of feature vectors, each vector is two dimensional vector, from my hand written lecture note, we have a feature vector (1, 2) from class C_1, which can fit into the equation above.

Now, the activation function of the outpu layer is

$$s_i = f(\sum_{i=1}^{2} w_{i,k} s_k + \phi) = f(\sum_{i=1}^{3} w_{i,k} s_k) \tag{12}$$

Definition 4. *Define a transfer function* for a single layer NN, denote σ_k as input, for multilayer NN, denote s_k as the input from the previous output layer.

$$h_i = \sum_{i=1}^{n_o} w_{i,k}\sigma_k + \phi = \sum_{i=1}^{n_o+1} w_{i,k}\sigma_k \tag{13}$$

So the neuron output i is

$$s_i = f(h_i) = f(h_i(w_{i,k} \cdot \sigma_k)) \tag{14}$$

Note for CNN with convolution layer(s), we have

$$s_i = f(h_i) = f(h_i(w_{i,k} \cdot g_{\sigma_k}(l_i * K))) \tag{15}$$

Since l_i convolution data layer, we drop it from the notation and keep the kernel K in the notation for formulation of training. So,

$$s_i = f(h_i) = f(h_i(w_{i,k} \cdot g_{\sigma_k}(K))) \tag{16}$$

Exampel 2. *Continued from Example 1, for a simple NN with 2 inputs and 1 output, we have the transfer function as*

$$h_i = \sum_{i=1}^{2} w_{i,k}\sigma_k + \phi = \sum_{i=1}^{3} w_{i,k}\sigma_k \tag{17}$$

Definition 5. *Define an error at each neuron output*

$${\zeta_i}^{\mu} - {s_i}^{\mu} \tag{18}$$

note where ${\zeta_i}^{\mu}$ (pronouced as 'zeta') is a desired (correct) output i at experiment μ, and ${s_i}^{\mu}$ is the actual output i at that experiment.

Now, for CNN, the error function above can be written as

$${\zeta_i}^{\mu} - f(h_i(w_{i,k} \cdot g_{\sigma_k}(K)))^{\mu} \tag{19}$$

where both $w_{i,k}$ and K have to be learned in the training of CNN.

Definition 6. *Define total error for all neuron outputs and for all experiments*

$$D = \frac{1}{2}\sum_{\mu=1}^{m}\sum_{i=1}^{n_o}({\zeta_i}^{\mu} - {s_i}^{\mu})^2 \tag{20}$$

note where ${\zeta_i}^{\mu}$ for i equal to 1,2,...,n_o, for n_o output neurons. Total number of experiment is m. In the previous lecture example, we have two classes (persons), each class with 3 feature vectors for the hand calculation, so experiment μ is equal to 6 due to total number of 6 feature vectors for the experiments.

In case of CNN, we will have the above equation to count the convolution kernel, so we have

$$D = \frac{1}{2}\sum_{\mu=1}^{M}\sum_{i=1}^{n_o}({\zeta_i}^{\mu} - f(h_i(w_{i,k} \cdot g_{\sigma_k}(K)))^{\mu})^2 \tag{21}$$

Property 1. *Minimize error function*

$$\begin{aligned} \tfrac{\partial D}{\partial w_{i,k}} &= \tfrac{\partial}{\partial w_{i,k}}\tfrac{1}{2}\sum_{\mu=1}^{6}\sum_{i=1}^{1}({\zeta_i}^{\mu} - {s_i}^{\mu})^2 \\ &= \sum_{\mu=1}^{6}({\zeta_i}^{\mu} - {s_i}^{\mu})f^{'}({h_i}^{\mu})\frac{\partial h_i}{\partial w_{i,k}} \end{aligned} \tag{22}$$

So for the hand calculation example, we have 1 output, and input feature vector is 2 dimension,

$$\frac{\partial D}{\partial w_{1,k}} = \frac{\partial}{\partial w_{1,k}}\frac{1}{2}\sum_{\mu=1}^{6}\sum_{i=1}^{1}({\zeta_1}^{\mu}-{s_1}^{\mu})^2 = \sum_{\mu=1}^{6}({\zeta_1}^{\mu}-{s_1}^{\mu})f^{'}({h_1}^{\mu})\frac{\partial h_1}{\partial w_{1,k}} \tag{23}$$

Note the derivative of the activation function

$$f^{'}({h_1}^{\mu}) \tag{24}$$

We can choose RELU as an activation function.

Property 2. *Minimize error function with convolution kernel coefficients*

$$\begin{aligned} \tfrac{\partial D}{\partial k_{i,k}} &= \tfrac{\partial}{\partial k_{i,k}}\tfrac{1}{2}\sum_{\mu=1}^{6}\sum_{i=1}^{1}({\zeta_i}^{\mu} - {s_i}^{\mu})^2 \\ &= \sum_{\mu=1}^{6}({\zeta_i}^{\mu} - {s_i}^{\mu})f^{'}({h_i}^{\mu})\frac{\partial h_i}{\partial k_{i,k}} \end{aligned} \tag{25}$$

Property 3. *Learning by updating the weights by gradient descent*

$$w_{i,k}(t+1) = w_{i,k}(t) + \delta w_{i,k}(t) \tag{26}$$

where

$$\delta w_{i,k}(t) = -\epsilon\frac{\partial D}{\partial w_{i,k}} \tag{27}$$

for the given example, we have

$$w_{1,k}(t+1) = w_{1,k}(t) + \delta w_{1,k}(t) \tag{28}$$

where

$$\delta w_{1,k}(t) = -\epsilon\frac{\partial D}{\partial w_{1,k}} \tag{29}$$

Writing the $\delta w_{i,k}(t) = -\epsilon\frac{\partial D}{\partial w_{i,k}}$ in a vector form, we have (??? given dimension up to n, generalized case)

$$\begin{pmatrix} \frac{\partial D}{\partial w_{i,1}} \\ \frac{\partial D}{\partial w_{i,2}} \\ . \\ . \\ . \\ \frac{\partial D}{\partial w_{i,n}} \end{pmatrix} = \begin{pmatrix} ({\zeta_i}^{\mu} - {s_i}^{\mu})f^{'}({h_i}^{\mu})\frac{\partial h_i}{\partial w_{i,1}} \\ ({\zeta_i}^{\mu} - {s_i}^{\mu})f^{'}({h_i}^{\mu})\frac{\partial h_i}{\partial w_{i,2}} \\ . \\ . \\ . \\ ({\zeta_i}^{\mu} - {s_i}^{\mu})f^{'}({h_i}^{\mu})\frac{\partial h_i}{\partial w_{i,n}} \end{pmatrix} \tag{30}$$