Note on Backpropagation in Neural Networks

Xin Jin

Note on Backpropagation in Neural Networks

Xin Jin
felixxinjin@gmail.com

August 8, 2019

Abstract

This note intends to facilitate low level implementation by providing an analytical perspective on neural networks. Different feedforward and recurrent neural networks are dissected through a derivation of the backpropagation update. We choose Multilayer Perceptron (MLP) which possesses the basic architecture of deep artificial neural network as a departure of introduction. Sigmoid Cross-Entropy loss is applied to MLP for an exemplification of multi-label classification. We then turn to introduce Convolutional Neural Network (CNN) — an intricate architecture which adopts filter and sub-sampling to realize a form of regularization. In the end, we illustrate Backpropagation Through Time (BPTT) to elicit Exploding / Vanishing Gradients problem and Long short-term memory (LSTM).

1 Fully Connected Neurons: Multilayer Perceptron

1.1 Structure of an elemental MLP

Multilayer Perceptron (MLP) has a structure of fully connected neurons. Neuron-like processing unit is the basic element of MLP:

$$a = \phi\left(\sum_i w_i x_i + b\right) \tag{1}$$

where x_i are input to the neuron, the w_i are the weights, b is the bias, ϕ is the activation function, and a is the unit's activation.

For simplicity but without loss of generality, we consider an elemental MLP with only 3 layers:

The 1st Layer (Input Layer) is composed of M neurons with no connection to each other. The activation-input relation in this layer is the intact transit of input and bias:

$$\begin{bmatrix} \overrightarrow{a}_1^{(in)} & \dots & \overrightarrow{a}_N^{(in)} \end{bmatrix} = \begin{bmatrix} 1 \dots 1 \\ a_{1,1}^{(in)} \dots a_{1,N}^{(in)} \\ \vdots \\ a_{M,1}^{(in)} \dots a_{M,N}^{(in)} \end{bmatrix} = \begin{bmatrix} 1 \dots 1 \\ x_{1,1}^{(in)} \dots x_{1,N}^{(in)} \\ \vdots \\ x_{M,1}^{(in)} \dots x_{M,N}^{(in)} \end{bmatrix} \tag{2}$$

In practical use, the M neurons are playing a role of dispatching M features $(x_{1,i}^{(in)} \dots x_{M,i}^{(in)})$ of the ith sample $x_i,\ i \in [1, \dots, N]$ respectively to the next layer.

The 2nd Layer (Hidden Layer) has K neurons each processes M activation of Input Layer as input:

$$\left[\overrightarrow{a}_1^{(h)} \dots \overrightarrow{a}_N^{(h)}\right] = \begin{bmatrix} 1 \dots 1 \\ \phi\left(\overrightarrow{w}_1^{(h)}\overrightarrow{a}_1^{(in)}\right) \dots \phi\left(\overrightarrow{w}_1^{(h)}\overrightarrow{a}_N^{(in)}\right) \\ \vdots \\ \phi\left(\overrightarrow{w}_K^{(h)}\overrightarrow{a}_1^{(in)}\right) \dots \phi\left(\overrightarrow{w}_K^{(h)}\overrightarrow{a}_N^{(in)}\right) \end{bmatrix} \tag{3}$$

where we choose sigmoid function $\phi(z) = \left(\frac{1}{1+e^{-z}}\right)$ as activation function in Hidden Layer.

The 3rd Layer (Output Layer) possesses C neurons, where C is the number of classes. The activation of the cth neuron represents the predicted conditional probability that the ith input sample belongs to class c:

$$\left[\overrightarrow{a}_1^{(out)} \dots \overrightarrow{a}_N^{(out)}\right] = \begin{bmatrix} \phi\left(\overrightarrow{w}_1^{(out)}\overrightarrow{a}_1^{(h)}\right) \dots \phi\left(\overrightarrow{w}_1^{(out)}\overrightarrow{a}_N^{(h)}\right) \\ \vdots \\ \phi\left(\overrightarrow{w}_c^{(out)}\overrightarrow{a}_1^{(h)}\right) \dots \phi\left(\overrightarrow{w}_c^{(out)}\overrightarrow{a}_N^{(h)}\right) \\ \vdots \\ \phi\left(\overrightarrow{w}_C^{(out)}\overrightarrow{a}_1^{(h)}\right) \dots \phi\left(\overrightarrow{w}_C^{(out)}\overrightarrow{a}_N^{(h)}\right) \end{bmatrix} \tag{4}$$

where $a_{c,i}^{(out)} = \phi\left(\overrightarrow{w}_c^{(out)}\overrightarrow{a}_i^{(h)} + b_c\right) = \hat{P}(y_i = c \mid x_i),\ i \in [1, \dots, N],\ c \in [1, \dots, C]$ and $\phi(z) = \left(\frac{1}{1+e^{-z}}\right)$.

1.2 Multi-Label MLP and Sigmoid Cross-Entropy loss

Multi-Label MLP is able to make prediction for Multi-Label Classification, where the sample x_i can belong to more than one class. If x_i belonging to a certain class doesn't influence the decision for another class (i.e. $P(y_i = c \mid x_i), c \in [1 \dots C]$ are independent), Multi-Label Classification problem can be split into C binary Classification problems. The solution of C binary Classification problems is to learn weights $\mathbf{w}$ by maximizing the likelihood $L(\mathbf{w})$ below:

$$L(\mathbf{w}) = \prod_{c=1}^{C} \prod_{i=1}^{n} \left(a_{c,i}^{(out)} \right)^{P(y_i = c|x_i)} \left(1 - a_{c,i}^{(out)} \right)^{1 - P(y_i = c|x_i)} \tag{5}$$

In practice, the equivalent approach of minimizing Sigmoid Cross-Entropy Loss $J(\mathbf{w})$ makes calculation convenient:

$$J(\mathbf{w}) = -\sum_{c=1}^{C} \sum_{i=1}^{n} \left[P(y_i = c \mid x_i) \log \left(a_{c,i}^{(out)} \right) + (1 - P(y_i = c \mid x_i)) \log \left(1 - a_{c,i}^{(out)} \right) \right] \tag{6}$$

1.3 Training MLP via Backpropagation

In order to update the weights through Backpropagation, we fisrt apply forward propagation to generate the activation of the output layer through (2),(3), and (4) with initial weight values. We then use gradient descent to update the weights in each layer:

$$\frac{\partial J(\mathbf{w})}{\partial w_{c,k}^{(out)}} = \left(a_{c,i}^{(out)} - P(y_i = c \mid x_i) \right) a_{k,i}^{(h)} \tag{7}$$

$$\frac{\partial J(\mathbf{w})}{\partial w_{k,m}^{(h)}} = \left(a_{c,i}^{(out)} - P(y_i = c \mid x_i) \right) w_{c,k}^{(out)} a_{k,i}^{(h)} \left(1 - a_{k,i}^{(h)} \right) a_{m,i}^{(in)} \tag{8}$$

$$w_{c,k}^{(out)} = w_{c,k}^{(out)} - \eta \frac{\partial J(\mathbf{w})}{\partial w_{c,k}^{(out)}} \tag{9}$$

$$w_{k,m}^{(h)} = w_{k,m}^{(h)} - \eta \frac{\partial J(\mathbf{w})}{w_{k,m}^{(h)}} \tag{10}$$

Where η is the learning rate. The derivation of Equation (7) and (8) is detailed in Appendix.

2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are composed of multiple Convolutional Layer / Pooling Layer pair followed by Fully Connection Layer at the end [1, 2]. We exhibit below a basic CNN with a single Convolutional Layer / Pooling Layer pair with a Fully Connection Layer.

2.1 Convolution Layer

Convolution Layer can be imagined as a variant of Hidden Layer in which the inner product of the weight and the input is replaced by convolution:

$$A_{c_{out}} = \phi\left(X * W_{c_{out}}^{(conv)} + b_{c_{out}}^{(conv)}\right) \tag{11}$$

$$A_{c_{out}}(i,j) = \phi\left(\sum_{k_1=0}^{m_1-1}\sum_{k_2=0}^{m_2-1} X(i-k_1, j-k_2) W_{c_{out}}^{(conv)}(k_1,k_2) + b_{c_{out}}^{(conv)}\right) \tag{12}$$

where $X \in R^{n_1 \times n_2}$ is the input matrix, $W_{c_{out}}^{(conv)} \in R^{m_1 \times m_2}$, $c_{out} \in \{1, \ldots, C_{out}\}$ is the kernel matrix for the c_{out}th channel, $b_{c_{out}}^{(conv)}$ is the bias for the c_{out}th channel, and $\phi(z) = \left(\frac{1}{1+e^{-z}}\right)$. We consider the zero-padded edges are not added to the original input matrix X, i.e. the kernel cannot exceed the boundary of input during the convolution which leads $i \in \{m_1, \ldots, n_1\}, j \in \{m_2, \ldots, n_2\}$.

2.2 Pooling Layer

Subsampling is performed in pooling layer, where the average value (mean-pooling) or the max value (max-pooling) is calculated in each sub-region. Here we use nonoverlapping mean-pooling as an example:

$$S_{c_{out}}(i,j) = \frac{1}{p_1 p_2}\sum_{k_1=1}^{p_1}\sum_{k_2=1}^{p_2} A_{c_{out}}(p_1 i' - k_1, p_2 j' - k_2) \tag{13}$$

where p_1,p_2 are pooling size in each dimension, $i' = m_1 + i \in \left\{m_1 + 1, \ldots, m_1 + \frac{n_1}{p_1}\right\}$ and $j' = m_1 + j \in \left\{m_2 + 1, \ldots, m_2 + \frac{n_2}{p_2}\right\}$.

2.3 Fully Connection Layer

Fully Connetion Layer is similar to the output layer which contains neuron-like processing units. Additionally, Fully Connetion Layer vectorizes and concatenates the output of pooling layer.

$$z = Z\left(\{S_{c_{out}}\}_{c_{out}=1,\ldots,C_{out}}\right) \tag{14}$$

$$\{S_{c_{out}}\}_{c_{out}=1,\ldots,C_{out}} = Z^{-1}(z) \tag{15}$$

Equation (14) denotes the process of vectorizing the output of each channel by column scan and concatenates them to form a whole string. Equation (15) represents the reverse process, where $z \in R^{C_{out}(n_1/p_1)(n_2/p_2)\times 1}$. The whole string is then input into the activation function for predicting class labels.

$$\overrightarrow{y} = \begin{bmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_c \\ \vdots \\ \hat{y}_C \end{bmatrix} = \phi\left(W^{(f)} z + \overrightarrow{b}\right) \tag{16}$$

where $\hat{y}_c = \hat{P}\left(y = c \mid X\right)$, C is the number of classes, $W^{(f)} \in R^{C \times C_{out}(n_1/p_1)(n_2/p_2)}$, $\overrightarrow{b} \in R^{C\times 1}$, and $\phi\left(z\right) = \left(\frac{1}{1+e^{-z}}\right)$.

2.4 Backpropagation in CNNs

We choose Cross-Entropy as the Loss Function in this note, since it outperforms Squared Error Loss for some circumstances [3]. The Cross-Entropy Loss J is specified as below:

$$J = -\sum_{c=1}^{C} P\left(y = c \mid X\right) \log\left(\hat{y}_c\right) \tag{17}$$

We still begin with performing forward propagation to generate $\overrightarrow{y}$ with initial value of weight and bias in Convolution Layer and Fully Connection Layer. Then we resort to gradient descent to update the weights and bias from Fully Connection Layer to Convolution Layer:

$$\frac{\partial J}{\partial W_{c,j}^{(f)}} = -\left(1 - \hat{y}_c\right) z_j \tag{18}$$

$$\frac{\partial J}{\partial b_c^{(f)}} = -\left(1 - \hat{y}_c\right) \tag{19}$$

$$\frac{\partial J}{\partial W_{c_{out}}^{(conv)}\left(k_1, k_2\right)} = \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} \alpha * X_{rot180} \tag{20}$$

$$\frac{\partial J}{\partial b_{c_{out}}^{(conv)}} = \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} \alpha\left(i, j\right) \tag{21}$$

$$W_{c,j}^{(f)} = W_{c,j}^{(f)} - \eta \frac{\partial J}{\partial W_{c,j}^{(f)}} \tag{22}$$

$$b_c^{(f)} = b_c^{(f)} - \eta \frac{\partial J}{\partial b_c^{(f)}} \tag{23}$$

$$W_{c_{out}}^{(conv)}\left(k_1, k_2\right) = W_{c_{out}}^{(conv)}\left(k_1, k_2\right) - \eta \frac{\partial J}{\partial W_{c_{out}}^{(conv)}\left(k_1, k_2\right)} \tag{24}$$

$$b_{c_{out}}^{(conv)} = b_{c_{out}}^{(conv)} - \eta \frac{\partial J}{\partial b_{c_{out}}^{(conv)}} \tag{25}$$

The derivation of Equation (18),(19),(20), and (21) can be refer to Appendix for the details.

3 Recurrent Neural Network

Different from a feedforward network (MLP as an example) dealing with independent and identically distributed data, Recurrent Neural Network (RNN) processes input data in a certain order and dependent of each other. Base on such purpose, RNN is designed to has input layer/hidden layers/output layer unit at each time instance with connection between hidden layers in the same level at two adjacent time instances [4, 5].

3.1 Structure of a Single Layer RNN

We consider a single layer RNN in which only one hidden layer locates at each time instance. The hidden layer activation at time instance τ is the function of $\overrightarrow{x}_\tau$ the input data at the time instance τ and $\overrightarrow{h}_{\tau-1}$ the hidden layer activation at time instance $\tau - 1$:

$$\overrightarrow{h}_\tau = \phi_h \left(W_{hh}^{\mathrm{T}} \overrightarrow{h}_{\tau-1} + W_{xh}^{\mathrm{T}} \overrightarrow{x}_\tau + \overrightarrow{b}_h \right) \tag{26}$$

where $\phi_h(\cdot)$ is the activation fucntion of the hidden layer and $\overrightarrow{b}_h$ is the bias vector.

Output layer pre-activation at instance τ is calculated from weighting matrix from hidden layer to output layer W_{hy}, hidden layer activation $\overrightarrow{h}_\tau(\mathrm{X})$, and the bias at output layer $\overrightarrow{b}_y$ as

$$\overrightarrow{\hat{y}}_\tau = \phi_y \left(W_{hy}^{\mathrm{T}} \overrightarrow{h}_\tau + \overrightarrow{b}_{\mathrm{y}} \right) \tag{27}$$

where $\phi_y(\cdot)$ is the activation fucntion of the output layer which is the softmax function usde in this note, and $\overrightarrow{\hat{y}}_\tau = [\hat{y}_1^\tau, \ldots, \hat{y}_c^\tau, \ldots, \hat{y}_C^\tau]$ is the predicted probability distribution at instance τ.

3.2 Backpropagation Through Time (BPTT) and Exploding / Vanishing Gradients problem

The Loss function J for RNN is the sum of all the loss functions at each time instance. Here we choose Cross-Entropy as the Loss Function:

$$J = \sum_{\tau=1}^{T} J^{(\tau)} = \sum_{\tau=1}^{T} \left(-\sum_{c=1}^{C} y_{c,\tau} \log\left(\hat{y}_{c,\tau}\right) \right) \tag{28}$$

As W_{hh} is include in the hidden layer activation at each time instance, we need to apply Backpropagation Through Time (BPTT) to RNN which will expand the partial derivative in the time demension when we calculate gradient descent to update the weights:

$$\begin{aligned}\frac{\partial J^{(\tau)}}{\partial W_{hh}} &= \frac{\partial \mathrm{J}^{(\tau)}}{\partial \overrightarrow{\hat{y}}_\tau}\frac{\partial \overrightarrow{\hat{y}}_\tau}{\partial \overrightarrow{h}_\tau}\left(\sum_{t=1}^{\tau}\frac{\partial \overrightarrow{h}_\tau}{\partial \overrightarrow{h}_t}\frac{\partial \overrightarrow{h}_t}{\partial W_{hh}}\right) \\ &= \frac{\partial \mathrm{J}^{(\tau)}}{\partial \overrightarrow{\hat{y}}_\tau}\frac{\partial \overrightarrow{\hat{y}}_\tau}{\partial \overrightarrow{h}_\tau}\left(\sum_{t=1}^{\tau}\prod_{i=t+1}^{\tau}\frac{\partial \overrightarrow{h}_i}{\partial \overrightarrow{h}_{i-1}}\frac{\partial \overrightarrow{h}_t}{\partial W_{hh}}\right) \\ &= \frac{\partial \mathrm{J}^{(\tau)}}{\partial \overrightarrow{\hat{y}}_\tau}\frac{\partial \overrightarrow{\hat{y}}_\tau}{\partial \overrightarrow{h}_\tau}\left(\sum_{t=1}^{\tau}\prod_{i=t+1}^{\tau}W_{hh}\frac{\partial \overrightarrow{h}_t}{\partial W_{hh}}\right)\end{aligned} \tag{29}$$

$\prod_{i=t+1}^{\tau} W_{hh}$ will lead $\frac{\partial J^{(\tau)}}{\partial \mathrm{W}_{hh}}$ to trend towards positive infinity when $|W_{hh}| > 1$ and $\tau - t$ is large. Conversely, $\frac{\partial J^{(\tau)}}{\partial \mathrm{W}_{hh}}$ will trend towards zero when $|W_{hh}| < 1$ and $\tau - t$ is large. The former is called Exploding Gradients problem and the other is called Vanishing Gradients problem.

3.3 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) [6] is a RNN architecture designed to overcome Exploding / Vanishing Gradients problem. LSTM is introduce 3 gates (Input gate i_τ, Forget gate f_τ, Output gate o_τ) to suppress $\overrightarrow{a}_\tau$ the input activation at the time instance τ, $\overrightarrow{C}_{\tau-1}$ the cell state at the time instance $\tau - 1$, $\overrightarrow{C}_\tau$ the cell state at the time instance τ respectively:

The calculation of gates are specified below, where σ denotes sigmoid function:

$$\overrightarrow{i}_\tau = \sigma\left(\overrightarrow{z_i}_\tau\right) = \sigma\left(W_{xi}\overrightarrow{x}_\tau + W_{hi}\overrightarrow{h}_{\tau-1} + \overrightarrow{b}_i\right) \tag{30}$$

$$\overrightarrow{f}_\tau = \sigma\left(\overrightarrow{z_f}_\tau\right) = \sigma\left(W_{xf}\overrightarrow{x}_\tau + W_{hf}\overrightarrow{h}_{\tau-1} + \overrightarrow{b}_f\right) \tag{31}$$

$$\overrightarrow{o}_\tau = \sigma\left(\overrightarrow{z_o}_\tau\right) = \sigma\left(W_{xo}\overrightarrow{x}_\tau + W_{ho}\overrightarrow{h}_{\tau-1} + \overrightarrow{b}_o\right) \tag{32}$$

The Feed-Forward operation of LSTM:

$$\overrightarrow{a}_\tau = \sigma\left(\overrightarrow{z_a}_\tau\right) = \tanh\left(W_{xa}\overrightarrow{x}_\tau + W_{hi}\overrightarrow{h}_{\tau-1} + \overrightarrow{b}_a\right) \tag{33}$$

$$\overrightarrow{C}_\tau = \left(\overrightarrow{a}_\tau \odot \overrightarrow{i}_\tau\right) \oplus \left(\overrightarrow{C}_{\tau-1} \odot \overrightarrow{f}_\tau\right) \tag{34}$$

$$\overrightarrow{h}_\tau = \tanh\left(\overrightarrow{C}_\tau\right) \odot \overrightarrow{o}_\tau \tag{35}$$

After the feed-forward operation is performed with initial value of weight and bias, the backpropagation is conducted to update weight and bias by minimizing the loss which we use the sum of squared of the errors (SSE) loss as an example:

$$J = \sum_{\tau=1}^{T} J^{(\tau)} = \sum_{\tau=1}^{T} \left(\frac{1}{2} \left(\overrightarrow{y}_{\tau} - \overrightarrow{h}_{\tau} \right)^{2} \right) \tag{36}$$

The calculation of the partial derivatives of loss function regarding W_{xa},$W_{xi}, W_{xf}, W_{xo}, \overrightarrow{b}_a$ are detailed below. The partial derivatives of loss function regarding $W_{hf},\ W_{xo},$ $W_{ho},\ W_{hi},\ \overrightarrow{b}_i,\ \overrightarrow{b}_f, \overrightarrow{b}_o$ have the very similar structure and are not specified here.

$$\frac{\partial J}{\partial W_{xa}} = \begin{cases} When\ \tau = T: \\ = \frac{\partial J^{(T)}}{\partial \overrightarrow{C}_T} \frac{\partial \overrightarrow{C}_T}{\partial \overrightarrow{a}_T} \frac{\partial \overrightarrow{a}_T}{W_{xa}} \\ = \left(\overrightarrow{h}_T - \overrightarrow{y}_T \right) \odot \left(1 - \tanh^2 \left(\overrightarrow{C}_T \right) \right) \odot \overrightarrow{o}_T \odot \overrightarrow{i}_T \odot \left(1 - \tanh^2 \left(\overrightarrow{z}_{aT} \right) \right) \overrightarrow{x}_T \\ When\ \tau < T: \\ = \frac{\partial J^{(T)}}{\partial \overrightarrow{C}_\tau} \frac{\partial \overrightarrow{C}_\tau}{\partial \overrightarrow{a}_\tau} \frac{\partial \overrightarrow{a}_\tau}{W_{xa}} + \ldots + \frac{\partial J^{(\tau)}}{\partial \overrightarrow{C}_\tau} \frac{\partial \overrightarrow{C}_\tau}{\partial \overrightarrow{a}_\tau} \frac{\partial \overrightarrow{a}_\tau}{W_{xa}} \\ = \left(\overrightarrow{h}_T - \overrightarrow{y}_T \right) \odot \left(1 - \tanh^2 \left(\overrightarrow{C}_T \right) \right) \odot \overrightarrow{o}_T \odot \overrightarrow{f}_T \ldots \overrightarrow{f}_{\tau+1} \odot \overrightarrow{i}_\tau \odot \left(1 - \tanh^2 \left(\overrightarrow{z}_{a\tau} \right) \right) \overrightarrow{x}_\tau \\ + \ldots + \left(\overrightarrow{h}_\tau - \overrightarrow{y}_\tau \right) \odot \left(1 - \tanh^2 \left(\overrightarrow{C}_\tau \right) \right) \odot \overrightarrow{o}_\tau \odot \overrightarrow{i}_\tau \odot \left(1 - \tanh^2 \left(\overrightarrow{z}_{a\tau} \right) \right) \overrightarrow{x}_\tau \end{cases} \tag{37}$$

$$\frac{\partial J}{\partial W_{xi}} = \begin{cases} When\ \tau = T: \\ = \frac{\partial J^{(T)}}{\partial \overrightarrow{C}_T} \frac{\partial \overrightarrow{C}_T}{\partial \overrightarrow{i}_T} \frac{\partial \overrightarrow{i}_T}{W_{xi}} \\ = \left(\overrightarrow{h}_T - \overrightarrow{y}_T \right) \odot \left(1 - \tanh^2 \left(\overrightarrow{C}_T \right) \right) \odot \overrightarrow{o}_T \odot \overrightarrow{a}_T \odot \left(1 - \tanh^2 \left(\overrightarrow{z}_{iT} \right) \right) \overrightarrow{x}_T \\ When\ \tau < T: \\ = \frac{\partial J^{(T)}}{\partial \overrightarrow{C}_\tau} \frac{\partial \overrightarrow{C}_\tau}{\partial \overrightarrow{i}_\tau} \frac{\partial \overrightarrow{i}_\tau}{W_{xi}} + \ldots + \frac{\partial J^{(\tau)}}{\partial \overrightarrow{C}_\tau} \frac{\partial \overrightarrow{C}_\tau}{\partial \overrightarrow{i}_\tau} \frac{\partial \overrightarrow{i}_\tau}{W_{xi}} \\ = \left(\overrightarrow{h}_T - \overrightarrow{y^T} \right) \odot \left(1 - \tanh^2 \left(\overrightarrow{C}_T \right) \right) \odot \overrightarrow{o}_T \odot \overrightarrow{f}_T \ldots \overrightarrow{f}_{\tau+1} \odot \overrightarrow{o}_\tau \odot \overrightarrow{a}_\tau \odot \left(1 - \tanh^2 \left(\overrightarrow{z}_{i\tau} \right) \right) \overrightarrow{x}_\tau \\ + \ldots + \left(\overrightarrow{h}_\tau - \overrightarrow{y}_\tau \right) \odot \left(1 - \tanh^2 \left(\overrightarrow{C}_\tau \right) \right) \odot \overrightarrow{o}_\tau \odot \overrightarrow{a}_\tau \odot \left(1 - \tanh^2 \left(\overrightarrow{z}_{i\tau} \right) \right) \overrightarrow{x}_\tau \end{cases} \tag{38}$$

$$\frac{\partial J}{\partial W_{xf}} = \begin{cases} When\ \tau = T: \\ = \frac{\partial J^{(T)}}{\partial \overrightarrow{C}_T} \frac{\partial \overrightarrow{C}_T}{\partial \overrightarrow{f}_T} \frac{\partial \overrightarrow{f}_T}{W_{xf}} \\ = \left(\overrightarrow{h}_T - \overrightarrow{y}_T \right) \odot \left(1 - \tanh^2 \left(\overrightarrow{C}_T \right) \right) \odot \overrightarrow{o}_T \odot \overrightarrow{C}_{T-1} \odot \left(1 - \tanh^2 \left(\overrightarrow{z}_{fT} \right) \right) \overrightarrow{x}_T \\ When\ \tau < T: \\ = \frac{\partial J^{(T)}}{\partial \overrightarrow{C}_\tau} \frac{\partial \overrightarrow{C}_\tau}{\partial \overrightarrow{f}_\tau} \frac{\partial \overrightarrow{f}_\tau}{W_{xf}} + \ldots + \frac{\partial J^{(\tau)}}{\partial \overrightarrow{C}_\tau} \frac{\partial \overrightarrow{C}_\tau}{\partial \overrightarrow{f}_\tau} \frac{\partial \overrightarrow{f}_\tau}{W_{xf}} \\ = \left(\overrightarrow{h}_T - \overrightarrow{y^T} \right) \odot \left(1 - \tanh^2 \left(\overrightarrow{C}_T \right) \right) \odot \overrightarrow{o}_T \odot \overrightarrow{f}_T \ldots \overrightarrow{f}_{\tau+1} \odot \overrightarrow{o}_\tau \odot \overrightarrow{C}_{\tau-1} \odot \left(1 - \tanh^2 \left(\overrightarrow{z}_{f\tau} \right) \right) \overrightarrow{x}_\tau \\ + \ldots + \left(\overrightarrow{h}_\tau - \overrightarrow{y}_\tau \right) \odot \left(1 - \tanh^2 \left(\overrightarrow{C}_\tau \right) \right) \odot \overrightarrow{o}_\tau \odot \overrightarrow{C}_{\tau-1} \odot \left(1 - \tanh^2 \left(\overrightarrow{z}_{f\tau} \right) \right) \overrightarrow{x}_\tau \end{cases} \tag{39}$$

$$\begin{aligned} \frac{\partial J}{\partial W_{xo}} &= \frac{\partial J^{(\tau)}}{\partial \overrightarrow{h}_\tau} \frac{\partial \overrightarrow{h}_\tau}{\partial \overrightarrow{o}_\tau} \frac{\partial \overrightarrow{o}_\tau}{W_{xo}} \\ &= \left(\overrightarrow{h}_T - \overrightarrow{y^T} \right) \odot \tanh \left(\overrightarrow{C}_\tau \right) \odot \left(1 - \tanh^2 \left(\overrightarrow{z}_{o\tau} \right) \right) \overrightarrow{x}_\tau \end{aligned} \tag{40}$$

$$\frac{\partial J}{\partial b_a} = \begin{cases} When\ \tau = T: \\ = \frac{\partial J^{(T)}}{\partial \overrightarrow{C}_T} \frac{\partial \overrightarrow{C}_T}{\partial \overrightarrow{a}_T} \frac{\partial \overrightarrow{a}_T}{b_a} \\ = \left(\overrightarrow{h}_T - \overrightarrow{y}_T \right) \odot \left(1 - \tanh^2 \left(\overrightarrow{C}_T \right) \right) \odot \overrightarrow{o}_T \odot \overrightarrow{i}_T \odot \left(1 - \tanh^2 \left(\overrightarrow{z}_{aT} \right) \right) \\ When\ \tau < T: \\ = \frac{\partial J^{(T)}}{\partial \overrightarrow{C}_\tau} \frac{\partial \overrightarrow{C}_\tau}{\partial \overrightarrow{a}_\tau} \frac{\partial \overrightarrow{a}_\tau}{W_{xa}} + \ldots + \frac{\partial J^{(\tau)}}{\partial \overrightarrow{C}_\tau} \frac{\partial \overrightarrow{C}_\tau}{\partial \overrightarrow{a}_\tau} \frac{\partial \overrightarrow{a}_\tau}{W_{xa}} \\ = \left(\overrightarrow{h}_T - \overrightarrow{y}_T \right) \odot \left(1 - \tanh^2 \left(\overrightarrow{C}_T \right) \right) \odot \overrightarrow{o}_T \odot \overrightarrow{f}_T \ldots \overrightarrow{f}_{\tau+1} \odot \overrightarrow{i}_\tau \odot \left(1 - \tanh^2 \left(\overrightarrow{z}_{a\tau} \right) \right) \\ + \ldots + \left(\overrightarrow{h}_\tau - \overrightarrow{y}_\tau \right) \odot \left(1 - \tanh^2 \left(\overrightarrow{C}_\tau \right) \right) \odot \overrightarrow{o}_\tau \odot \overrightarrow{i}_\tau \odot \left(1 - \tanh^2 \left(\overrightarrow{z}_{a\tau} \right) \right) \end{cases} \tag{41}$$

4 Appendix

4.1 Derivation of Equations

Derivation of Equation (7): Using the chain rule, the partial derivative of loss with respect to the weight in output layer can be calculated as:

$$\begin{aligned}\frac{\partial J(\mathbf{w})}{\partial w_{c,k}^{(out)}} &= \frac{\partial J(\mathbf{w})}{\partial a_{c,i}^{(out)}}\frac{\partial a_{c,i}^{(out)}}{\partial w_{c,k}^{(out)}} \\ &= -\left(\frac{P\left(y_i=c\mid x_i\right)}{a_{c,i}^{(out)}}-\frac{\left(1-P\left(y_i=c\mid x_i\right)\right)}{1-a_{c,i}^{(out)}}\right)\frac{\partial\phi\left(z_{c,i}^{(out)}\right)}{\partial z_{c,i}^{(out)}}\frac{\partial z_{c,i}^{(out)}}{\partial w_{c,k}^{(out)}} \\ &= -\left(\frac{P\left(y_i=c\mid x_i\right)}{a_{c,i}^{(out)}}-\frac{\left(1-P\left(y_i=c\mid x_i\right)\right)}{1-a_{c,i}^{(out)}}\right)a_{c,i}^{(out)}\left(1-a_{c,i}^{(out)}\right)a_{k,i}^{(h)} \\ &= \left(a_{c,i}^{(out)}-P\left(y_i=c\mid x_i\right)\right)a_{k,i}^{(h)}\end{aligned} \tag{42}$$

where $z_{c,i}^{(out)}=a_{1,i}^{(h)}w_{c,1}^{(out)}+\ldots+a_{k,i}^{(h)}w_{c,k}^{(out)}+\ldots+a_{K,i}^{(h)}w_{c,K}^{(out)}$.

Derivation of Equation (8):

$$\begin{aligned}\frac{\partial J(\mathbf{w})}{\partial w_{k,m}^{(h)}} &= \frac{\partial J(\mathbf{w})}{\partial a_{c,i}^{(out)}}\frac{\partial a_{c,i}^{(out)}}{\partial a_{k,i}^{(h)}}\frac{\partial a_{k,i}^{(h)}}{\partial w_{k,m}^{(h)}} \\ &= -\left(\frac{P\left(y_i=c\mid x_i\right)}{a_{c,i}^{(out)}}-\frac{\left(1-P\left(y_i=c\mid x_i\right)\right)}{1-a_{c,i}^{(out)}}\right)a_{c,i}^{(out)}\left(1-a_{c,i}^{(out)}\right)w_{c,k}^{(out)}a_{k,i}^{(h)}\left(1-a_{k,i}^{(h)}\right)a_{m,i}^{(in)} \\ &= \left(a_{c,i}^{(out)}-P\left(y_i=c\mid x_i\right)\right)w_{c,k}^{(out)}a_{k,i}^{(h)}\left(1-a_{k,i}^{(h)}\right)a_{m,i}^{(in)}\end{aligned} \tag{43}$$

Derivation of Equation (18):

$$\begin{aligned}\frac{\partial J}{\partial W_{c,j}^{(f)}} &= \frac{\partial J}{\partial \hat{y}_c}\frac{\partial \hat{y}_c}{\partial W_{c,j}^{(f)}} \\ &= -\frac{1}{\hat{y}_c}\frac{\partial\phi\left(\sum_{j=0}^{C_{out}(n_1/p_1)(n_2/p_2)-1}W_{c,j}^{(f)}z+b_c^{(f)}\right)}{\partial W_{c,j}^{(f)}} \\ &= -\frac{1}{\hat{y}_c}\hat{y}_c\left(1-\hat{y}_c\right)z_j \\ &= -\left(1-\hat{y}_c\right)z_j\end{aligned} \tag{44}$$

Derivation of Equation (19):

$$\begin{aligned}\frac{\partial J}{\partial b_c^{(f)}} &= \frac{\partial J}{\partial \hat{y}_c}\frac{\partial \hat{y}_c}{\partial b_c^{(f)}} \\ &= -\frac{1}{\hat{y}_c}\frac{\partial\phi\left(\sum_{j=0}^{C_{out}(n_1/p_1)(n_2/p_2)-1}W_{c,j}^{(f)}z+b_c^{(f)}\right)}{\partial b_c^{(f)}} \\ &= -\left(1-\hat{y}_c\right)\end{aligned} \tag{45}$$

Derivation of Equation (18):

$$\begin{aligned}
\frac{\partial J}{\partial W^{(conv)}_{c_{out}}(k_1,k_2)} &= \frac{\partial J}{\partial A_{c_{out}}(i,j)} \frac{\partial A_{c_{out}}(i,j)}{\partial W^{(conv)}_{c_{out}}(k_1,k_2)} \\
&= \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} \frac{\partial J}{\partial A_{c_{out}}(i,j)} \frac{\partial\left(\phi\left(\sum_{k_1=0}^{m_1-1}\sum_{k_2=0}^{m_2-1} X(i-k_1,j-k_2)W^{(conv)}_{c_{out}}(k_1,k_2)+b^{(conv)}_{c_{out}}\right)\right)}{\partial W^{(conv)}_{c_{out}}(k_1,k_2)} \\
&= \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} \frac{\partial J}{\partial A_{c_{out}}(i,j)} A_{c_{out}}(i,j)\left(1-A_{c_{out}}(i,j)\right) X(i-k_1,j-k_2) \\
&= \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} \alpha(i,j)\, X_{rot180}(k_1-i,k_2-j) \\
&= \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} \alpha * X_{rot180}
\end{aligned} \tag{46}$$

where X_{rot180} denotes rotating X 180 degrees. $\alpha = \frac{\partial J}{\partial A_{c_{out}}(i,j)} A_{c_{out}}(i,j)$ and $\frac{\partial J}{\partial A_{c_{out}}(i,j)}$ can be obtained by devectorizing and upsampling $\frac{\partial J}{\partial z}$. Equation (47), (48), (49) provides the solving process to get analytical solution of $\frac{\partial J}{\partial A_{c_{out}}(i,j)}$.

$$\begin{aligned}
\frac{\partial J}{\partial z} &= \frac{\partial J}{\partial \hat{y}_c}\frac{\partial \hat{y}_c}{\partial z} \\
&= -\frac{1}{\hat{y}_c} W_{c,j}
\end{aligned} \tag{47}$$

$$\frac{\partial J}{\partial S_{c_{out}}} = Z^{-1}\left(\frac{\partial J}{\partial z}\right) \tag{48}$$

$$\frac{\partial J}{\partial A_{c_{out}}(i,j)} = p_1 p_2 \frac{\partial J}{\partial S_{c_{out}}}\left(\lceil i/p_1\rceil \lceil j/p_2\rceil\right) \tag{49}$$

Derivation of Equation (21):

$$\begin{aligned}
\frac{\partial J}{\partial b^{(conv)}_{c_{out}}} &= \frac{\partial J}{\partial A_{c_{out}}(i,j)} \frac{\partial A_{c_{out}}(i,j)}{\partial b^{(conv)}_{c_{out}}} \\
&= \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} \alpha(i,j) \\
&= \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} p_1 p_2 \frac{\partial J}{\partial S_{c_{out}}}\left(\lceil i/p_1\rceil \lceil j/p_2\rceil\right) A_{c_{out}}(i,j)
\end{aligned} \tag{50}$$

References

[1] Zhifei Zhang. 2016. Derivation of Backpropagation in Convolutional Neural Network (CNN). Technical Report. University of Tennessee, Knoxvill, TN.

[2] Bouvrie, J. 2006. Notes on convolutional neural networks. Chatfield, K.; Simonyan, K.; Vedaldi, A.; and Zisserman, A. 2014. Return of the devil in the details: Delving deep into convolutional nets. arXiv preprint arXiv:1405.3531.

[3] F.J. Huang and Y. LeCun. "Large-scale Learning with SVM and Convolutional for Generic Object Categorization", In: Proc. 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 284-291, 2006.

[4] Pascanu, Razvan, Mikolov, Tomas, and Bengio, Yoshua. On the difficulty of training recurrent neural networks. In Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16- 21 June 2013, pp. 1310–1318, 2013.

[5] Hojjat Salehinejad, Julianne Baarbe, Sharan Sankar, Joseph Barfett, Errol Colak, and Shahrokh Valaee, "Recent advances in recurrent neural networks," arXiv preprint arXiv:1801.01078, 2017.

[6] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. arXiv:1412.3555 [cs], December 2014. URL http://arxiv.org/abs/1412.3555.