Back Propagation Summary

0. Overview

This text is mainly about back propagation in deep learning. The relevant implementation in *Mint* merely includes linear, convolutional, max-pooling and flat modules, up to now.

1. Notation

The data flow in deep neural networks mainly includes two type, i.e. vectors and maps. Given the commonly-adopted batch-based training pattern, one additional dimension is always added. Formally, we note the vector in size (N, M) and map in size (N, C, H, W), where N is batch size, M vector length, C channel numbers, H and W the height and width respectively.

The back propagation is always incited by objective function. Here, we note $\mathcal{L}$ and aim at its minimum as default.

2. Linear

Formulate the forwarding process in lth linear layer with M^l neurons as

$$z^l = x^l \cdot W^l + b^l,\ \ x^{l+1} = \sigma(z^l),$$

where $x^l \in \mathbb{R}^{N, M^{l-1}}$ is the input of lth linear layer, $z^l \in \mathbb{R}^{N, M^l}$ the output and x^{l+1} its activation by $\sigma(.)$ as the input for the following layer, and $W^l \in \mathbb{R}^{M^{l-1}, M^l}$ and $b^l \in \mathbb{R}^{M^l}$ are the weight matrix and bias to be updated.

2.1 Gradient on Weight

Start with the gradient for each element of the weight matrix. Given the output is multi-dimensioned, the gradient should sum-up those on different dimensions. Formally,

$$\frac{\partial \mathcal{L}}{\partial w^l_{i,j}} = \sum_{n=0}^{N-1} \sum_{m=0}^{M^l-1} \frac{\partial \mathcal{L}}{\partial z^l_{n,m}} \frac{\partial z^l_{n,m}}{\partial w^l_{i,j}}.$$

Note that in the forwarding process,

$$z^l_{n,m} = \sum_{k=0}^{M^{l-1}-1} x^l_{n,k} \cdot w^l_{k,m} + b^l_m,$$

which indicates that $z^l_{n,m}$ is irrelevant with $w^l_{i,j}$ if $m \neq j$. Therefore, we are free to simplify the gradient as

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial w^l_{i,j}} &= \sum_{n=0}^{N-1} \frac{\partial \mathcal{L}}{\partial z^l_{n,j}} \frac{\partial z^l_{n,j}}{\partial w^l_{i,j}} \\ &= < \frac{\partial \mathcal{L}}{\partial z^l_{*,j}}, \frac{\partial z^l_{*,j}}{\partial w^l_{i,j}} > \\ &= < \frac{\partial \mathcal{L}}{\partial z^l_{*,j}}, x^l_{*,i} >\end{aligned}$$

where $\frac{\partial \mathcal{L}}{\partial z^l_{*,j}}$ denotes jth column of $\frac{\partial \mathcal{L}}{\partial z^l}$, and $x^l_{*,i}$ the ith column of x^l.

To achieve a more elegant form, note the fact that $\frac{\partial \mathcal{L}}{\partial W^l} \in \mathbb{R}^{M^{l-1},M^l}$, $\frac{\partial \mathcal{L}}{\partial z^l} \in \mathbb{R}^{N,M^l}$ and $x^l \in \mathbb{R}^{N,M^{l-1}}$. The final gradient is thus a matrix multiplication as

$$\frac{\partial \mathcal{L}}{\partial W^l} = {x^l}^{\mathrm{T}} \cdot \frac{\partial \mathcal{L}}{\partial z^l}.$$

2.2 Gradient on Bias

Similarly, the gradient on bias can be written as

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial b^l_i} &= \sum_{n=0}^{N-1} \sum_{m=0}^{M^l-1} \frac{\partial \mathcal{L}}{\partial z^l_{n,m}} \frac{\partial z^l_{n,m}}{\partial b^l_i} \\ &= \sum_{n=0}^{N-1} \frac{\partial \mathcal{L}}{\partial z^l_{n,i}}.\end{aligned}$$

2.3* Gradient to Propagate

The gradient to propagate to the previous layer is $\frac{\partial \mathcal{L}}{\partial x^l}$, which is of size (N, M^{l-1}). Start with each element:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial x^l_{n,j}} &= \sum_{m=0}^{M^l-1} \frac{\partial \mathcal{L}}{\partial z^l_{n,m}} \frac{\partial z^l_{n,m}}{\partial x^l_{n,j}} \\ &= \sum_{m=0}^{M^l-1} \frac{\partial \mathcal{L}}{\partial z^l_{n,m}} w^l_{j,m} \\ &= < \frac{\partial \mathcal{L}}{\partial z^l_{n,*}}, w^l_{j,*} >.\end{aligned}$$

The form is similar to the gradient on weights, and the only difference is the matrix multiplied on the gradient from the following layer:

$$\frac{\partial \mathcal{L}}{\partial x^l} = \frac{\partial \mathcal{L}}{\partial z^l} \cdot {W^l}^T.$$

3. Convolutional

3.0 Overview

Convolutional layers are more complicated than linear ones, due to various additional hyper-parameters, including *kernel size*, *stride*, *padding*, *dilate*, *transposed* and so on. In this text, we temporarily discuss the cases where only the first three are involved, i.e. (F, S, P). In addition, we only consider the 2-dimensioned case for simplicity.

In the following convolutional parts, we formulate the forwarding process as

$$Z^l = X^l * K^l + b^l,\ X^{l+1} = \sigma(Z^l),$$

where $Z^l \in \mathbb{R}^{N,C^l,H^l,W^l}$ the output and X^{l+1} its activation, $X^l \in \mathbb{R}^{N,C^{l-1},H^{l-1},W^{l-1}}$, $K^l \in \mathbb{R}^{C^l,C^{l-1},F^l,F^l}$ the kernel and b^l the bias.

3.1 Convolution with Unit Stride and No Padding

With unit stride, we can develop the output as

$$z^l_{n,c,h,w} = \sum_{c'=0}^{C^{l-1}-1} \sum_{i=0}^{F^l-1} \sum_{j=0}^{F^l-1} x^l_{n,c',h+i,w+j} k^l_{c,c',i,j}.$$

3.1.1 Gradient on Kernel

As computing the gradient in linear section, we study each element,

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial k^l_{c,c',i,j}} &= \sum_{n=0}^{N-1} \sum_{h=0}^{H^l-1} \sum_{w=0}^{W^l-1} \frac{\partial \mathcal{L}}{\partial z^l_{n,c,h,w}} \frac{\partial z^l_{n,c,h,w}}{\partial k^l_{c,c',i,j}} \\ &= \sum_{n=0}^{N-1} \sum_{h=0}^{H^l-1} \sum_{w=0}^{W^l-1} \frac{\partial \mathcal{L}}{\partial z^l_{n,c,h,w}} x^l_{n,c',h+i,w+j} \\ &= \sum_{n=0}^{N-1} \sum_{h=0}^{H^l-1} \sum_{w=0}^{W^l-1} \delta^l_{n,c,h,w} x^l_{n,c',h+i,w+j},\end{aligned}$$

where $\delta^l_{n,c,h,w} = \partial \mathcal{L} / \partial z^l_{n,c,h,w}$. The result of simplification is quite similar to the convolution form, though, in fact, so it is indeed, if we exchange the dimension of batch size and channel, i.e.

$$\frac{\partial \mathcal{L}}{\partial k^l_{c,c',i,j}} = \sum_{n=0}^{N-1} \sum_{h=0}^{H^l-1} \sum_{w=0}^{W^l-1} \tilde{x}^l_{c',n,i+h,j+w} \tilde{\delta}^l_{c,n,h,w},$$

where $\tilde{*}$ notes the variable after the exchange, then the result is equivalent to a convolution on a map (batch size c', channel n, size $H^{l-1} \times W^{l-1}$) with a kernel (out channel c, input channel n, size $H^l \times W^l$).

3.1.2 Gradient on Bias

The gradient on bias does not differ much from that of linear layer,

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial b_c^l} &= \sum_{n=0}^{N-1}\sum_{h=0}^{H^l-1}\sum_{w=0}^{W^l-1}\frac{\partial \mathcal{L}}{\partial z_{n,c,h,w}^l}\frac{\partial z_{n,c,h,w}^l}{\partial b_c^l}\\ &= \sum_{n=0}^{N-1}\sum_{h=0}^{H^l-1}\sum_{w=0}^{W^l-1}\frac{\partial \mathcal{L}}{\partial z_{n,c,h,w}^l}.\end{aligned}$$

3.1.3* Gradient to Propagate

Intuitively, computing the gradient to propagate back to the previous layer may also involve a convolution operation. Inspired from the linearity property, a possible choice is convolving the gradient from the following layer by the current kernel. However, the size of result is not consistent with the input map. The solution to this problem is padding. To have an insight, return to the gradient on each element,

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial x_{n,c,h,w}^l} &= \sum_{c'=0}^{C^l-1}\sum_{h'=0}^{H^l-1}\sum_{w'=0}^{W^l-1}\frac{\partial \mathcal{L}}{\partial z_{n,c',h',w'}^l}\frac{\partial z_{n,c',h',w'}^l}{\partial x_{n,c,h,w}^l}\\ &= \sum_{c'=0}^{C^l-1}\sum_{h'=0}^{H^l-1}\sum_{w'=0}^{W^l-1}\frac{\partial \mathcal{L}}{\partial z_{n,c',h',w'}^l}\frac{\partial z_{n,c',h',w'}^l}{\partial x_{n,c,h,w}^l}\mathbb{I}_{F^l}(h,h',w,w')\end{aligned}$$

where

$$\mathbb{I}_{F^l}(h,h',w,w') = \begin{cases}1, & (h',w') \in [\max(0,h-F^l+1),h] \times [\max(0,w-F^l+1),w]\\ 0, & \text{otherwise.}\end{cases}$$

Thus, the spatial summing can be limited in a $F^l \times F^l$ window, and the equation can be rewritten as

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial x_{n,c,h,w}^l} &= \sum_{c'=0}^{C^l-1}\sum_{i=0}^{F^l-1}\sum_{j=0}^{F^l-1}\frac{\partial \mathcal{L}}{\partial z_{n,c',h-F^l+1+i,w-F^l+1+j}^l}\frac{\partial z_{n,c',h-F^l+1+i,w-F^l+1+j}^l}{\partial x_{n,c,h,w}^l}\\ &= \sum_{c'=0}^{C^l-1}\sum_{i=0}^{F^l-1}\sum_{j=0}^{F^l-1}\frac{\partial \mathcal{L}}{\partial z_{n,c',h-F^l+1+i,w-F^l+1+j}^l}k_{c',c,F^l-1-i,F^l-1-j}^l.\end{aligned}$$

Here, we temporarily ignore the board problem where the index is negative. Meanwhile, the F^l in the index is annoying (maybe not :)). To solve these problems, an immediate choice is padding the map $\partial\mathcal{L}/\partial z^l$ with $F^l - 1$ zeros on each side. Now update the equation above,

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial x_{n,c,h,w}^l} &= \sum_{c'=0}^{C^l-1}\sum_{i=0}^{F^l-1}\sum_{j=0}^{F^l-1}\ddot{\delta}_{n,c',h+i,w+j}^l k_{c',c,F^l-1-i,F^l-1-j}^l\\ &= \sum_{c'=0}^{C^l-1}\sum_{i=0}^{F^l-1}\sum_{j=0}^{F^l-1}\ddot{\delta}_{n,c',h+i,w+j}^l \hat{k}_{c,c',i,j}^l,\end{aligned}$$

where $\ddot{\delta}^l$ denotes $\partial\mathcal{L}/\partial z^l$ padded, $\hat{k}^l$ is the original kernel of whom weights are rotated by 180° and output/input channels are exchanged.

Note that this form is a convolution as well as during forwarding, which is quite a beautiful math coincidence.

3.2 Convolution with Non-unit stride S and No Padding

3.2.1 Gradient on Kernel

With stride S, the forwarding process can be written as

$$z^l_{n,c,h,w} = \sum_{c'=0}^{C^{l-1}-1} \sum_{i=0}^{F^l-1} \sum_{j=0}^{F^l-1} x^l_{n,c',hS+i,wS+j} k^l_{c,c',i,j}.$$

Differentiate on each element of kernel,

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial k^l_{c,c',i,j}} &= \sum_{n=0}^{N-1} \sum_{h=0}^{H^l-1} \sum_{w=0}^{W^l-1} \frac{\partial \mathcal{L}}{\partial z^l_{n,c,h,w}} \frac{\partial z^l_{n,c,h,w}}{\partial k^l_{c,c',i,j}} \\
&= \sum_{n=0}^{N-1} \sum_{h=0}^{H^l-1} \sum_{w=0}^{W^l-1} \frac{\partial \mathcal{L}}{\partial z^l_{n,c,h,w}} x^l_{n,c',hS+i,hS+j}.
\end{aligned}$$

This form is similar to convolution, except that the stride S in the index of x^l is annoying. To get rid of stride, we can insert $S-1$ zeros between rows and columns of $\partial\mathcal{L}/\partial z^l$, thus

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial k^l_{c,c',i,j}} &= \sum_{n=0}^{N-1} \sum_{h=0}^{H^l-1} \sum_{w=0}^{W^l-1} \frac{\partial \mathcal{L}}{\partial \check{z}^l_{n,c,h,w}} x^l_{n,c',h+i,w+j} \\
&= \sum_{n=0}^{N-1} \sum_{h=0}^{H^l-1} \sum_{w=0}^{W^l-1} \check{\delta}^l_{n,c,h,w} x^l_{n,c',h+i,w+j} \\
&= \sum_{n=0}^{N-1} \sum_{h=0}^{H^l-1} \sum_{w=0}^{W^l-1} \tilde{x}^l_{c',n,h+i,w+j} \tilde{\check{\delta}}^l_{c,n,h,w},
\end{aligned}$$

where $\check{*}$ denotes the map with zeros inserted between rows and columns, $\tilde{*}$ does exchange of dimension (seen in 3.1.1).

3.2.2 Gradient on Bias

The same as before, thus omitted.

3.2.3* Gradient to Propagate

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial x^l_{n,c,h,w}} &= \sum_{c'=0}^{C^l-1} \sum_{h'=0}^{H^l-1} \sum_{w'=0}^{W^l-1} \frac{\partial \mathcal{L}}{\partial z^l_{n,c',h',w'}} \frac{\partial z^l_{n,c',h',w'}}{\partial x^l_{n,c,h,w}} \\
&= \sum_{c'=0}^{C^l-1} \sum_{h'=0}^{H^l-1} \sum_{w'=0}^{W^l-1} \delta^l_{n,c',h',w'} \frac{\partial z^l_{n,c',h',w'}}{\partial x^l_{n,c,h,w}} \mathbb{I}_{F^l}(h,h',w,w'),
\end{aligned}$$

where $\mathbb{I}_{F^l}$ is an indicator that

$$\mathbb{I}_{F^l}(h,h',w,w') = \begin{cases} 1, & (h'S, w'S) \in [\max(0, h-F^l+1), h] \times [\max(0, w-F^l+1), w] \\ 0, & \text{otherwise.} \end{cases}$$

If we can get rid of S, the situation is completely the same as in 3.1.3. Fortunately, in 3.2.1, we have already achieved this by inserting $S-1$ zeros between rows and columns. Thus, formally

$$\frac{\partial L}{\partial x^l_{n,c,h,w}} = \sum_{c'=0}^{C^l-1} \sum_{i=0}^{F^l-1} \sum_{j=0}^{F^l-1} \ddot{\check{\delta}}_{n,c',h+i,w+j} \hat{\boldsymbol{k}}^l_{c,c',i,j},$$

which means a convolution on gradient from the following layer (padded and inserted by zeros) by the flipped kernel.

3.3 Convolution with Padding

The case with padding is easy, we only need to remove the corresponding elements from the gradient from the following layer. Thus omitted.

4. Max-pooling

TODO