Back propagation reference

January 13, 2017

Abstract

This document serves as a reference for students, engineers and researchers who occasionally need to work through the analytic computation of gradients and back-propagation update calculations. The document is an ongoing effort and there may be errors. Please feel free to send pull requests with new layers or corrections.

Contents

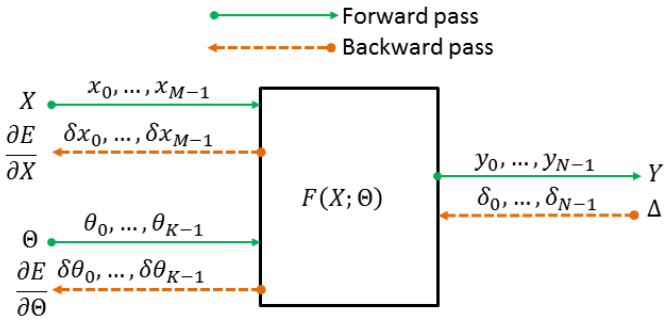

Figure 1: This image illustrates a general view of a network layer that is useful to understand back propagation. The layer takes inputs X, parameters Θ and produces an output Y. Computing the output is performed during the forward pass. Updating the parameters and passing the error signal Δ back from the next layer is performed during the backward pass.

1 Intro

We will consider here the most general function description of a neural network layer where the network takes in a multi-dimensional input X and a set of tunable parameters Θ. It produces a multi-dimensioanl output Y. When X, Y, Θ are scalars we will refer to them as x, y, θ. Instead of Θ and θ we may also use W, w, B, b if the context is related to convolution. Figure 1 illustrates this setup. We will try to preserve this notation throughout the document to aid in understanding and implementation. Each Section is intended to be independent of the others although there may be cross referencing.

2 Neuron activation functions

Neuron activation functions take a single input and produce a single output. There are two groups of such functions: those with tunable parameters and those without. However in the general case we can write down the following function

$$y = f(x, \theta), \tag{1}$$

where x is a scalar input, y is a scalar output and θ is a vector of tunable parameters. The backprop equations here are

$$\frac{\mathrm{d}E}{\mathrm{d}x} = \delta \frac{\mathrm{d}f(x,\theta)}{\mathrm{d}x} \tag{2}$$

$$\frac{\partial E}{\partial \theta_i} = \delta \frac{\partial f(x,\theta)}{\partial \theta_i}. \tag{3}$$

2.1 ReLU

The **Re**ctified **L**inear **U**nit function is given by:

$$f(x) = \begin{cases} 0 & x < 0 \\ x & x \geq 0 \end{cases} \tag{4}$$

Finding its derivative is straightforward. We simply find the derivative for each section in its domain. We can directly compute the backprop gradient:

$$\frac{\mathrm{d}E}{\mathrm{d}x} = \delta \frac{\mathrm{d}f}{\mathrm{d}x} \tag{5}$$

$$= \delta \begin{cases} 0 & x < 0 \\ 1 & x \geq 0 \end{cases} \tag{6}$$

$$= [\delta]_{x \geq 0} \tag{7}$$

In other words ReLU acts like an on/off switch for error signals. If the input x was greater than or equal to zero then the error signal δ will be passed back to the next layer down. If not then no error signal will be funneled through and anything behind this neuron will not be updated.

2.2 Leaky ReLU

2.3 TanH

For the hyperbolic tangent function (or TanH),

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{8}$$

Finding its derivative can be done using the quotient rule,

$$\frac{\mathrm{d}f}{\mathrm{d}x} = \frac{\mathrm{d}}{\mathrm{d}x}\frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{9}$$

$$= \frac{(e^x + e^{-x})(e^x + e^{-x}) - (e^x - e^{-x})(e^x - e^{-x})}{(e^x + e^{-x})^2} \tag{10}$$

$$= \frac{(e^x + e^{-x})^2 - (e^x - e^{-x})^2}{(e^x + e^{-x})^2} \tag{11}$$

$$= 1 - \left(\frac{e^x - e^{-x}}{e^x + e^{-x}}\right)^2 \tag{12}$$

$$= 1 - \left(\tanh(x)\right)^2. \tag{13}$$

Plugging the derivative into equation 2, we get

$$\frac{\mathrm{d}E}{\mathrm{d}x} = \delta\frac{\mathrm{d}f}{\mathrm{d}x} \tag{14}$$

$$= \delta\left(1 - y^2\right). \tag{15}$$

2.4 Sigmoid

The Sigmoid function is defined as:

$$\sigma(x) = \frac{1}{1 + e^{-x}}. \tag{16}$$

Its derivative is simply

$$\frac{\mathrm{d}\sigma}{\mathrm{d}x} = -\frac{-e^{-x}}{\left(1 + e^{-x}\right)^2} \tag{17}$$

$$= \frac{1}{1 + e^{-x}}\frac{e^{-x}}{1 + e^{-x}} \tag{18}$$

$$= \frac{1}{1 + e^{-x}}\left(1 - \frac{1}{1 + e^{-x}}\right) \tag{19}$$

$$= \sigma(x)\left(1 - \sigma(x)\right). \tag{20}$$

Plugging the derivative into equation 2, we get:

$$\frac{\mathrm{d}E}{\mathrm{d}x} = \delta\frac{\mathrm{d}\sigma}{\mathrm{d}x} \tag{21}$$

$$= \delta y\left(1 - y\right). \tag{22}$$

2.5 Absolute value

The absolute value is defined as:

$$f(x) = |x| \tag{23}$$

It can be said that $|x|$ is equivalent to $\sqrt{x^2}$ and therefore its dirivative can be written as:

$$\frac{\mathrm{d}f}{\mathrm{d}x} = \frac{\mathrm{d}}{\mathrm{d}x}\sqrt{x^2} \tag{24}$$

$$= \frac{\mathrm{d}}{\mathrm{d}x}(x^2)^{\frac{1}{2}} \tag{25}$$

Using the chain rule we get:

$$\frac{\mathrm{d}f}{\mathrm{d}x} = \frac{1}{2}(x^2)^{-\frac{1}{2}}(2x) \tag{26}$$

$$= \frac{x}{\sqrt{x^2}} \tag{27}$$

$$= \frac{x}{|x|} \tag{28}$$

Plugging the derivative into equation 2, we get:

$$\frac{\mathrm{d}E}{\mathrm{d}x} = \delta\frac{x}{|x|} \tag{29}$$

where $x \neq 0$

2.6 Power

3 Windowed functions

The family of functions which considers a window of a spatial input includes among others pooling and convolutional layers. There are often many ways to perform back propagation across these layers and usually depends on the desired performance considerations when implemented in a computer. To index into our input X which will typically be $C \times H \times W$, we will use the following: $i \in 0 \cdots H-1$ traverses the rows (y direction in image), $j \in 0 \cdots W-1$ traverses the columns (x direction in image) and $c \in 0 \cdots C-1$ traverses the channels (colors or filters direction often called depth).

3.1 Max pooling

Max pooling takes in a

$$f(X) = \max_{i \in 0 \cdots n} x_i \tag{30}$$

3.2 Max unpooling

3.3 Average pooling

An average pooling layer performs down-sampling by computing the mean representation of the input X with the use of an appropriate window function. Given a region determined by the window function represented by the vector X_k with a set of activations $\{x_1, ...x_{|X_k|}\} \in X_k$, the operation for average pooling can be defined as follows:

$$f(X_k) = \frac{1}{|X_k|} \sum_{i \in X_k} x_i \tag{31}$$

The function f receives an input vector which defines a region within X and produces a scalar output representing the mean of the region. In the most common cases f is applied to dijoint (non-overlapping) regions. To calculate the backpropagation in this layer we use the same principal shown in section 2 above and start by computing $\frac{\mathrm{d}f}{\mathrm{d}x}$, which is calculated as follows:

$$\frac{\mathrm{d}f}{\mathrm{d}x} = \frac{\mathrm{d}}{\mathrm{d}x} \frac{1}{|X_k|} \sum_{i \in X_k} x_i \tag{32}$$

$$= \frac{1}{|X_k|} \tag{33}$$

Finally, plugging the derivative into equation 2, we get:

$$\frac{\mathrm{d}E}{\mathrm{d}x} = \delta \frac{1}{|X_k|} \tag{34}$$

4 Restructuring functions

There are numerous cases where the structure of the input data need to be adjusted such as flattening, cropping, padding, resizing and more. In these cases the challenge is to figure out how to perform back prop on the data which has been moved around, cut off or added.

4.1 Padding

4.2 Cropping

5 Loss functions

5.1 Euclidean

The Euclidean loss is defined by the average of the square of the residuals and is written as

$$E(X,Y) = \frac{1}{CHW} \sum_{i,j,k}^{C,H,W} \frac{1}{2} \left(X_{ijk} - Y_{ijk}\right)^2 \tag{35}$$

where here we arbitrarily assume an input X with three dimensions and an output Y with the same structure. The loss function gradient which then typically becomes the first set of error signals Δ can be written as

$$\begin{aligned}
\delta_{cmn} =& \frac{\partial E}{\partial X_{cmn}} && (36)\\
=& \frac{1}{CHW} \sum_{i,j,k}^{C,H,W} \left(X_{ijk} - Y_{ijk}\right) \frac{X_{ijk}}{\partial X_{cmn}} && (37)\\
=& \frac{1}{CHW} \left(X_{cmn} - Y_{cmn}\right) && (38)
\end{aligned}$$

or in matrix form

$$\Delta = \frac{1}{CHW} \left(X - Y\right) \tag{39}$$

5.2 Cross entropy

5.3 Discrete total variation

The discrete total variation can be written as

$$TV(X) = \sum_{i,j} \left(\left(X_{ij+1} - X_{ij}\right)^2 + \left(X_{i+1j} - X_{ij}\right)^2 \right)^{1/2} = \sum_{i,j} \left| \nabla_{ij} X \right|_2, \tag{40}$$

where $i \in (0 \dots H-1)$ and $j \in (0 \dots W-1)$. X is used to represent the image and has dimensions $H \times W$. For the sake of simplicity we will assume a single channel because the result shown here will simply be redone for all channels and all examples in a batch. In this case the *total variation* is a loss and therefore our backprop update only requires that we compute the gradient of TV with respect to the image. Also note that for convenience of notation we use $\left|\nabla_{ij} X\right|_2$ to indicate the magnitude of the forward discrete derivative at (ij).

$$
\begin{aligned}
\frac{\partial E}{\partial X_{mn}} =& \frac{\partial TV(X)}{\partial X_{mn}} \\
=& \sum_{i,j} \frac{\partial \left|\nabla_{ij} X\right|_2}{\partial X_{mn}} \\
=& \sum_{i,j} \frac{1}{2} \frac{1}{\left|\nabla_{ij} X\right|_2} \left[2(X_{ij+1} - Xij)(\frac{\partial X_{ij+1}}{\partial X_{mn}} - \frac{\partial X_{ij}}{\partial X_{mn}}) + 2(X_{i+1j} - Xij)(\frac{\partial X_{i+1j}}{\partial X_{mn}} - \frac{\partial X_{ij}}{\partial X_{mn}}) \right] \\
=& \sum_{i,j} \frac{1}{\left|\nabla_{ij} X\right|_2} \left[\frac{\partial X_{ij+1}}{\partial X_{mn}} \left(X_{ij+1} - X_{ij} \right) \right] + \sum_{i,j} \frac{1}{\left|\nabla_{ij} X\right|_2} \left[\frac{\partial X_{i+1j}}{\partial X_{mn}} \left(X_{i+1j} - X_{ij} \right) \right] - \\
& \sum_{i,j} \frac{1}{\left|\nabla_{ij} X\right|_2} \left[\frac{\partial X_{ij}}{\partial X_{mn}} \left(X_{ij+1} - X_{ij} \right) \right] - \sum_{i,j} \frac{1}{\left|\nabla_{ij} X\right|_2} \left[\frac{\partial X_{ij}}{\partial X_{mn}} \left(X_{i+1j} - X_{ij} \right) \right] \\
=& \frac{(X_{mn} - X_{mn-1})}{\left|\nabla_{mn-1} X\right|_2} + \frac{(X_{mn} - X_{m-1n})}{\left|\nabla_{m-1n} X\right|_2} - \frac{(X_{mn+1} - X_{mn})}{\left|\nabla_{mn} X\right|_2} - \frac{(X_{m+1n} - X_{mn})}{\left|\nabla_{mn} X\right|_2}
\end{aligned}
\tag{41}
$$

6 Matrix functions

6.1 Fully connected layer (inner product)

The fully connected layer takes a fixed size input and linearly mixes every input value with every other and then adds an offset to produce an output with a defined size. This is written in matrix form as

$$F(x; W, b) = Wx + b, \tag{42}$$

and elementwise as

$$y_i = b_i + \sum_k W_{ik} x_k, \tag{43}$$

where x is a vector which has typically been flattened from a multi-dimensional input, W is a mixing matrix, b is an offset vector and y is the result vector matching the required dimensions. We can write our backprop equations as

$$\frac{\partial E}{\partial x_n} = \sum_i \delta_i \frac{\partial y_i}{\partial x_n}. \tag{44}$$

$$\frac{\partial E}{\partial W_{mn}} = \sum_i \delta_i \frac{\partial y_i}{\partial W_{mn}}. \tag{45}$$

$$\frac{\partial E}{\partial b_m} = \sum_i \delta_i \frac{\partial y_i}{\partial b_m}. \tag{46}$$

The gradients are striaghtforward to compute and we write them as

$$\frac{\partial y_i}{\partial x_n} = \sum_k W_{ik} \frac{\partial x_k}{\partial x_n} = W_{in} \tag{47}$$

$$\frac{\partial y_i}{\partial W_{mn}} = \sum_k \frac{\partial W_{ik}}{\partial W_{mn}} x_k = [x_n]_{m=i} \tag{48}$$

$$\frac{\partial y_i}{\partial b_m} = \frac{\partial b_i}{\partial b_m} = [1]_{m=i} \tag{49}$$

We can now expand the backprop update equations to become

$$\frac{\partial E}{\partial x_n} = \sum_i \delta_i W_{in} = \sum_i W^T_{ni} \delta_i \tag{50}$$

$$\frac{\partial E}{\partial W_{mn}} = \sum_i \delta_i \, [x_n]_{m=i} = \delta_m x_n \tag{51}$$

$$\frac{\partial E}{\partial b_m} = \sum_i \delta_i \, [1]_{m=i} = \delta_m \tag{52}$$

or in matrix/vector form

$$\frac{\partial E}{\partial x} = W^T \delta \tag{53}$$

$$\frac{\partial E}{\partial W} = \delta x^T \tag{54}$$

$$\frac{\partial E}{\partial b} = \delta \tag{55}$$

6.2 Gram matrix

The (un-normalized) Gram matrix is computed as

$$F(X) = XX^T, \tag{56}$$

where X is a $C \times N$ matrix and we set $Y = F(X)$. We denote a row of X as X_i and an element as X_{ik}. This implies that the Gram matrix can be written as

$$Y_{ij} = \langle X_i, X_j \rangle = \sum_{k=0}^{N-1} X_{ik} X_{jk}, \tag{57}$$

where $i, j \in (0 \dots C-1)$.Now for backprop we need to compute the following backprop quantity

$$\frac{\partial E}{\partial X_{cn}} = \sum_{i,j} \delta_{ij} \frac{\partial Y_{ij}}{\partial X_{cn}}. \tag{58}$$

Here E is the total loss for a single input example and δ is the gradient signal coming down from the next layer. The batch version will simply add the gradient for each example in the batch so to simplify notation we do not include the batch version here.

$$\begin{aligned}
\frac{\partial Y_{ij}}{\partial X_{cn}} =& \frac{\partial \left(\sum_{k=0}^{N-1} X_{ik} X_{jk} \right)}{\partial X_{cn}} \\
=& \sum_{k=0}^{N-1} \left[\frac{\partial X_{ik}}{\partial X_{cn}} X_{jk} + \frac{\partial X_{jk}}{\partial X_{cn}} X_{ik} \right] \\
=& \sum_{k=0}^{N-1} \frac{\partial X_{ik}}{\partial X_{cn}} X_{jk} + \sum_{k=0}^{N-1} \frac{\partial X_{jk}}{\partial X_{cn}} X_{ik} \\
=& \left[X_{jn} \right]_{c=i} + \left[X_{in} \right]_{c=j}
\end{aligned} \tag{59}$$

So now we can write the backprop as

$$\begin{aligned}
\frac{\partial E}{\partial X_{cn}} =& \sum_{i,j} \delta_{ij} \left(\left[X_{jn} \right]_{c=i} + \left[X_{in} \right]_{c=j} \right) \\
=& \sum_{i,j} \delta_{ij} \left[X_{jn} \right]_{c=i} + \sum_{i,j} \delta_{ij} \left[X_{in} \right]_{c=j} \\
=& \sum_{j} \delta_{cj} X_{jn} + \sum_{i} \delta_{ic} X_{in} \\
=& \sum_{i} \delta_{ci} X_{in} + \sum_{i} \delta_{ic} X_{in} \\
=& \sum_{i} \left(\delta_{ci} + \delta_{ic} \right) X_{in}
\end{aligned} \tag{60}$$

and as a matrix expression

$$\frac{\partial E}{\partial X} = (\delta + \delta^T) X \tag{61}$$