Back propagation reference

January 11, 2017

Abstract

This document serves as a reference for students, engineers and researchers who occasionally need to work through the analytical computation of gradients and backpropagation 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

1 Intro **2**

2 Neuron activation functions **3**
2.1 Non-tunable . . . 3
2.1.1 ReLU . . . 3
2.2 Tunable . . . 3
2.2.1 Leaky ReLU . . . 3

3 Windowed functions **3**
3.1 Non-tunable . . . 4
3.1.1 Max pooling . . . 4
3.1.2 Average pooling . . . 4
3.2 Tunable . . . 4

4 Loss functions **5**
4.1 Discrete total variation . . . 5

5 Matrix functions **6**
5.1 Gram matrix . . . 6

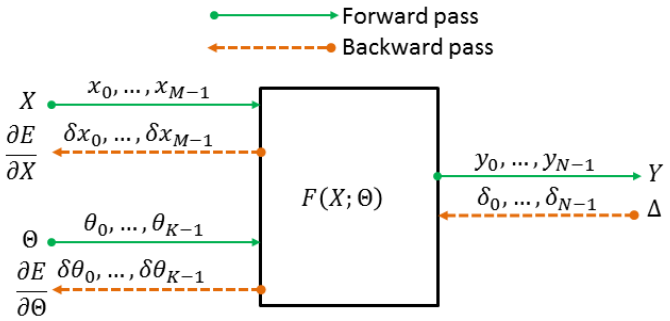

Figure 1: This image illustrates a general view of a network layer that is useful to understand back propagation. The layer takes inputs X, parameters Θ and produces an output Y. Computing the output is performed during the forward pass. Updating the parameters and passing the error signal Δ back from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

2.1.1 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 it's derivative is straightforward. We simply find the derivative for each section in it's domain.

$$\frac{\mathrm{d}E}{\mathrm{d}x} = \delta \begin{cases} 0 & x < 0 \\ 1 & x \geq 0 \end{cases} \tag{5}$$

$$= [\delta]_{x \geq 0} \tag{6}$$

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 Tunable

2.2.1 Leaky ReLU

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). Once again we can split the types of windowed functions into those that are tunable and non-tunable.

3.1 Non-tunable

3.1.1 Max pooling

Max pooling takes in a

$$f(X) = \max_{i \in 0 \cdots n} x_i \tag{7}$$

3.1.2 Average pooling

3.2 Tunable

4 Loss functions

4.1 Discrete total variation

The discrete total variation can be written as

$$TV(U) = \sum_{i,j} \left((U_{ij+1} - U_{ij})^2 + (U_{i+1j} - U_{ij})^2 \right)^{1/2}, \tag{8}$$

where $i \in (0 \dots H-1)$ and $j \in (0 \dots W-1)$. U 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

$$\begin{aligned}
\frac{\partial E}{\partial U_{mn}} =& \frac{\partial TV(U)}{\partial U_{mn}} \\
=& \sum_{i,j} \frac{\partial \left((U_{ij+1} - U_{ij})^2 + (U_{i+1j} - U_{ij})^2 \right)^{1/2}}{\partial U_{mn}} \\
=& \sum_{i,j} \frac{1}{2} \frac{1}{|\nabla U_{ij}|_2} \left[2(U_{ij+1} - Uij)(\frac{\partial U_{ij+1}}{\partial U_{mn}} - \frac{\partial U_{ij}}{\partial U_{mn}}) + 2(U_{i+1j} - Uij)(\frac{\partial U_{i+1j}}{\partial U_{mn}} - \frac{\partial U_{ij}}{\partial U_{mn}}) \right] \\
=& \sum_{i,j} \frac{1}{|\nabla U_{ij}|_2} \left[\frac{\partial U_{ij+1}}{\partial U_{mn}} (U_{ij+1} - U_{ij}) \right] + \sum_{i,j} \frac{1}{|\nabla U_{ij}|_2} \left[\frac{\partial U_{i+1j}}{\partial U_{mn}} (U_{i+1j} - U_{ij}) \right] - \\
& \sum_{i,j} \frac{1}{|\nabla U_{ij}|_2} \left[\frac{\partial U_{ij}}{\partial U_{mn}} (U_{ij+1} - U_{ij}) \right] - \sum_{i,j} \frac{1}{|\nabla U_{ij}|_2} \left[\frac{\partial U_{ij}}{\partial U_{mn}} (U_{i+1j} - U_{ij}) \right] \\
=& \frac{(U_{mn} - U_{mn-1})}{\left((U_{mn} - U_{mn-1})^2 + (U_{m+1n-1} - U_{mn-1})^2 \right)^{1/2}} + \\
& \frac{(U_{mn} - U_{m-1n})}{\left((U_{m-1n+1} - U_{m-1n})^2 + (U_{mn} - U_{m-1n})^2 \right)^{1/2}} - \\
& \frac{(U_{mn+1} - U_{mn})}{\left((U_{mn+1} - U_{mn})^2 + (U_{m+1n} - U_{mn})^2 \right)^{1/2}} - \\
& \frac{(U_{m+1n} - U_{mn})}{\left((U_{mn+1} - U_{mn})^2 + (U_{m+1n} - U_{mn})^2 \right)^{1/2}}
\end{aligned} \tag{9}$$

5 Matrix functions

5.1 Gram matrix

The (un-normalized) Gram matrix is computed as

$$G(X) = XX^T, \tag{10}$$

where X is a $C \times N$ matrix. 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

$$G_{ij} = \langle X_i, X_j \rangle = \sum_{k=0}^{N-1} X_{ik} X_{jk}, \tag{11}$$

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

$$\frac{\partial E}{\partial X_{cn}} = \sum_{i,j} \delta_{ij} \frac{\partial G_{ij}}{\partial X_{cn}}. \tag{12}$$

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.

$$\begin{aligned}
\frac{\partial G_{ij}}{\partial X_{cn}} =& \frac{\partial \left(\sum_{k=0}^{N-1} X_{ik} X_{jk} \right)}{\partial X_{cn}} \\
=& \sum_{k=0}^{N-1} \left[\frac{\partial X_{ik}}{\partial X_{cn}} X_{jk} + \frac{\partial X_{jk}}{\partial X_{cn}} X_{ik} \right] \\
=& \sum_{k=0}^{N-1} \frac{\partial X_{ik}}{\partial X_{cn}} X_{jk} + \sum_{k=0}^{N-1} \frac{\partial X_{jk}}{\partial X_{cn}} X_{ik} \\
=& [X_{jn}]_{c=i} + [X_{in}]_{c=j}
\end{aligned} \tag{13}$$

So now we can write the backprop as

$$\begin{aligned}
\frac{\partial E}{\partial X_{cn}} =& \sum_{i,j} \delta_{ij} \left([X_{jn}]_{c=i} + [X_{in}]_{c=j} \right) \\
=& \sum_{i,j} \delta_{ij} [X_{jn}]_{c=i} + \sum_{i,j} \delta_{ij} [X_{in}]_{c=j} \\
=& \sum_{j} \delta_{cj} X_{jn} + \sum_{i} \delta_{ic} X_{in} \\
=& \sum_{i} \delta_{ci} X_{in} + \sum_{i} \delta_{ic} X_{in} \\
=& \sum_{i} (\delta_{ci} + \delta_{ic}) X_{in}
\end{aligned} \tag{14}$$

and as a matrix expression

$$\frac{\partial E}{\partial X} = (\delta + \delta^T) X \tag{15}$$