Draft version February 5, 2018
Typeset using LaTeX **modern** style in AASTeX61

THE MATHEMATICS OF DEEP LEARNING*

Vishal P. Kasliwal[1]

[1] *Wave Computing*
42 W. Campbell Ave, # 301
Campbell, CA 95008, USA

(Received December 28, 2017; Revised December 28, 2017; Accepted February 5, 2018)

ABSTRACT

Deep Learning is a branch of Machine Learning in which 'deep' neural networks are used for various purposes.

Keywords: neural networks, training

Corresponding author: Vishal Kasliwal
vishal@wavecomp.com, vishal.kasliwal@gmail.com

* Draft prepared for personal use.

1. PRELIMINARIES

2. NOTATION

Tensors are associated with layers. The $p \times q$ tensor $\mathbf{A}$ when associated with layer l is denoted $\mathbf{A}^{[l]}_{p\times q}$. The transpose of this tensor is denoted by $(\mathbf{A}^{[l]}_{p\times q})^\top$. If the tensor is an input tensor to a layer, it may additionally have a minibatch index, m, and instance within the minibatch, i, associated with it. Thus $(\mathbf{A}^{[l]\{m\}(i)}_{p\times q})^\top$ is the transpose of the i-th training example from the m-th minibatch in the l-th layer from the input tensor $\mathbf{A}$ which is p rows high by q columns long. Furthermore, layers associated with convolutional neural networks may have *channels* associated with them. Channels add extra dimensions to tensors and so $(\mathbf{A}^{[l]\{m\}(i)}_{p\times q\times k_1\times k_2})^\top$ is the transpose of the i-th training example from the m-th minibatch in the l-th layer from the input tensor $\mathbf{A}$ which is p rows high by q columns long and has two sets of channels of depth k_1 and k_2 respectively.

The $\mathrm{n}^{[l]}$ operator returns the number of nodes, i.e. the width, of the l-th layer. Traditionally, the depth of the neural network is denoted by L, and hence we must have $0 \leqslant l \leqslant L$.

We transform the $l-1$-th layer of activations $\mathbf{A}^{[l-1]\{m\}(i)}_{p\times q}$, using neural network operations such as convolution etc... in the l-th layer to produce the l-th layer of activations i.e. $\mathbf{A}^{[l]\{m\}(i)}_{p\times q}$.

Neural networks use several different types of products. We shall denote the standard scalar product of two numbers with *no* special symbol i.e. the product of the scalars a & b shall be denoted by ab.

Note that we distinguish between the tensor $\mathbf{T}$ and the i-, j-th component of that tensor T_{ij}. The tensors encountered in Deep Learning can have lots of large dimensions. Hence, we shall write all operations using components as opposed to whole tensors. There is no downside to this choice since we never perform coordinate transformations in Deep Learning and hance writing tensor equations buys us no benefit.

3. OPERATIONS

3.1. *ReLU Activation Function*

The ReLU (Rectified Linear Unit) activation function takes the form

$$g(z) = \begin{cases} 0 & \text{if } z < 0 \\ z & \text{otherwise,} \end{cases} \tag{1}$$

making it discontinuous at $z = 0$. Deep Learning cheats around this discontinuity by defining the derivative so that it is either right-continuous or left-continuous. In this work, we shall choose to make the derivative right-continuous. Therefore, the derivative of the ReLU activation function is

$$\frac{\mathrm{d}g(z)}{\mathrm{d}z} = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{otherwise,} \end{cases} \tag{2}$$

3.1.1. *Derivation*

It is evident that for $z < 0$,

$$\frac{\mathrm{d}g(z)}{\mathrm{d}z} = 0,$$

since $g(z) = 0 \forall z < 0$. Similarly, since $g(z) = 1 \forall z \geqslant 0$,

$$\frac{\mathrm{d}g(z)}{\mathrm{d}z} = 1,$$

$\forall z \geqslant 0$. Since

$$\lim_{z \uparrow 0} \frac{\mathrm{d}g(z)}{\mathrm{d}z} = 0,$$

and

$$\lim_{z \downarrow 0} \frac{\mathrm{d}g(z)}{\mathrm{d}z} = 1,$$

we can arbitarily choose to define

$$\frac{\mathrm{d}g(z)}{\mathrm{d}z}\Big|_{z=0} = 1,$$

3.2. *Sigmoid Activation Function*

The sigmoid activation function takes the form

$$g(z) = \sigma(z) = \frac{1}{1 + \mathrm{e}^{-z}}. \tag{3}$$

The derivative of the sigmoid activation function is

$$\frac{\mathrm{d}\sigma}{\mathrm{d}z} = \sigma(1 - \sigma). \tag{4}$$

3.2.1. *Derivation*

By definition

$$\sigma(z) = \frac{1}{1 + \mathrm{e}^{-z}} = (1 + \mathrm{e}^{-z})^{-1},$$

and so

$$\frac{\mathrm{d}\sigma(z)}{\mathrm{d}z} = -(1 + \mathrm{e}^{-z})^{-2} \frac{\mathrm{d}(1 + \mathrm{e}^{-z})}{\mathrm{d}z}.$$

But this is just

$$-(1 + \mathrm{e}^{-z})^{-2}(-\mathrm{e}^{-z}) = \sigma^2 \mathrm{e}^{-z}.$$

Now $\mathrm{e}^{-z} = \frac{1}{\sigma} - 1 = \frac{1-\sigma}{\sigma}$ and so

$$\frac{\mathrm{d}\sigma(z)}{\mathrm{d}z} = \sigma^2 \mathrm{e}^{-z} = \sigma^2 \frac{1 - \sigma}{\sigma} = \sigma(1 - \sigma),$$

Q.E.D.

3.3. *Bias (Convolutional Layer)*

Convolutional layers usually include a bias term of the following form -

$$\mathbf{Z}^{(m)} = \mathbf{Y}^{(m)} + \boldsymbol{b}, \tag{5}$$

i.e.

$$Z^{(m)}_{hwc} = Y^{(m)}_{hwc} + b_c, \tag{6}$$

where the same bias is applied to each pixel within the same channel i.e. equation (6) expands to $H \times W$ equations by Einstein's summation convention. The derivative of the loss with respect to the bias terms are given by

$$\frac{\mathrm{d}\mathscr{L}}{\mathrm{d}b_c} = \sum_{a,i,j=1}^{M,H,W} \frac{\partial \mathscr{L}}{\partial Z^{(a)}_{ijc}}, \tag{7}$$

while the derivative of the loss with respect to the inputs are given by

$$\frac{\mathrm{d}\mathscr{L}}{\mathrm{d}Y^{(m)}_{hwc}} = \frac{\partial \mathscr{L}}{\partial Z^{(m)}_{hwc}} \tag{8}$$

3.3.1. *Derivation*

First, consider the derivative of the loss with respect to the bias. By the chain rule, we have

$$\frac{\mathrm{d}\mathscr{L}}{\mathrm{d}b_c} = \sum_{a,i,j,k=1}^{M,H,W,C} \frac{\partial \mathscr{L}}{\partial Z^{(a)}_{ijk}} \frac{\mathrm{d}Z^{(a)}_{ijk}}{\mathrm{d}b_c}.$$

Notice that

$$\frac{\mathrm{d}Z^{(a)}_{ijk}}{\mathrm{d}b_c} = \delta_{kc},$$

so that

$$\frac{\mathrm{d}\mathscr{L}}{\mathrm{d}b_c} = \sum_{a,i,j,k=1}^{M,H,W,C} \delta_{ck} \frac{\partial \mathscr{L}}{\partial Z^{(a)}_{ijk}} = \sum_{a,i,j=1}^{M,H,W} \frac{\partial \mathscr{L}}{\partial Z^{(a)}_{ijc}},$$

Q.E.D.

Now consider the derivative of the loss with respect to the input derivatives from the previous operation. Again, applying the chain rule we get

$$\frac{\mathrm{d}\mathscr{L}}{\mathrm{d}Y^{(m)}_{hwc}} = \sum_{a,i,j,k=1}^{M,H,W,C} \frac{\partial \mathscr{L}}{\partial Z^{(a)}_{ijk}} \frac{\mathrm{d}Z^{(a)}_{ijk}}{\mathrm{d}Y^{(m)}_{hwc}}.$$

But notice that

$$\frac{\mathrm{d}Z^{(a)}_{ijk}}{\mathrm{d}Y^{(m)}_{hwc}} = \delta_{am}\delta_{ih}\delta_{jw}\delta_{kc},$$

so that

$$\frac{\mathrm{d}\mathscr{L}}{\mathrm{d}Y^{(m)}_{hwc}} = \sum_{a,i,j,k=1}^{M,H,W,C} \delta_{am}\delta_{ih}\delta_{jw}\delta_{kc} \frac{\partial \mathscr{L}}{\partial Z^{(a)}_{ijk}} = \frac{\partial \mathscr{L}}{\partial Z^{(m)}_{hwc}},$$

Q.E.D.

3.4. *Convolution*

Convolution operations convolve a set of $C \times F \times F \times C'$-dimensional filters with a $H' \times W' \times C'_{\text{i}}$-dimensional input tensor $\mathbf{X}$ producing the $H \times W \times C$-dimensional output tensor $\mathbf{Y}$ via

$$\mathbf{Y}^{(m)} = \mathbf{W} * \mathbf{X}^{(m)}. \tag{9}$$

$\mathbf{X}$ is often padded with 0s in the H & W dimensions in order to make the size of the tensor a multiple of the filter size. Padding $\mathbf{X}$ (dimensions $H' \times W' \times C'$) with p zeros produces the padded tensor $\tilde{\mathbf{X}}$ (dimensions $\tilde{H}' \times \tilde{W}' \times C' \equiv H' + 2p \times W' + 2p \times C$). Additionally, the *stride* s of the convolution is not always 1. Given non-zero padding p and non-unit stride s, the convolution operation in equation (9) can be computed as

$$Y^{(m)}_{abc} = \sum_{f_1, f_2 = \frac{1-F}{2}, c'=1}^{\frac{F-1}{2}, C'} W_{cf_1f_2c'} \tilde{X}^{(m)}_{a'+f_1, b'+f_2, c'}, \tag{10}$$

with $a' = \frac{F-1}{2} + (a-1)s$ and $b' = \frac{F-1}{2} + (b-1)s$.

The derivative of the loss with respect to the filter $\mathbf{W}$ is given by

$$\frac{\mathrm{d}\mathscr{L}}{\mathrm{d}W_{cf_1f_2c'}} = \sum_{mab}^{MHW} \tilde{X}^{(m)}_{a'+f_1, b'+f_2, c'} \frac{\partial L}{\partial Y^{(m)}_{abc}}, \tag{11}$$

with $a' = \frac{F-1}{2} + (a-1)s$ and $b' = \frac{F-1}{2} + (b-1)s$.

3.4.1. *Derivation*

We wish to compute the total derivative of the loss $\mathscr{L}$ with respect to the filter $\mathbf{W}$ and also the total derivative of $\mathscr{L}$ with respect to the input tensor $\mathbf{X}$ given the derivative of $\mathscr{L}$ with respect to the output tensor $\mathbf{Y}$.

First lets compute the total derivative of the loss $\mathscr{L}$ with respect to the filter $\mathbf{W}$. Using the chain rule, we have

$$\frac{\mathrm{d}\mathscr{L}}{\mathrm{d}W_{cf_1f_2c'}} = \sum_{mabk}^{MHWC} \frac{\partial \mathscr{L}}{\partial Y^{(m)}_{abk}} \frac{\mathrm{d}Y^{(m)}_{abk}}{\mathrm{d}W_{cf_1f_2c'}}.$$

Equation (10) gives

$$\frac{\mathrm{d}Y^{(m)}_{abk}}{\mathrm{d}W_{cf_1f_2c'}} = \delta_{kc} \tilde{X}^{(m)}_{a'+f_1, b'+f_2, c'},$$

with $a' = \frac{F-1}{2} + (a-1)s$ and $b' = \frac{F-1}{2} + (b-1)s$. So

$$\frac{\mathrm{d}\mathscr{L}}{\mathrm{d}W_{cf_1f_2c'}} = \sum_{mabk}^{MHWC} \delta_{kc} \tilde{X}^{(m)}_{a'+f_1, b'+f_2, c'} \frac{\partial L}{\partial Y^{(m)}_{abk}} = \sum_{mab}^{MHW} \tilde{X}^{(m)}_{a'+f_1, b'+f_2, c'} \frac{\partial L}{\partial Y^{(m)}_{abc}}$$

Similarly, we may compute the total derivative of $\mathscr{L}$ with respect to the inputs $\mathbf{X}$ using the chain rule as follows

$$\frac{\mathrm{d}\mathscr{L}}{\mathrm{d}X^{(m)}_{a'b'c'}} = \sum_{lijk}^{MHWC} \frac{\partial \mathscr{L}}{\partial Y^{(l)}_{ijk}} \frac{\mathrm{d}Y^{(l)}_{ijk}}{\mathrm{d}X^{(m)}_{a'b'c'}}.$$

Now

$$\frac{\mathrm{d}Y^{(l)}_{ijk}}{\mathrm{d}X^{(m)}_{a'b'c'}} = \delta_{lm}$$

4. OUTPUT LAYERS

5. FEED-FORWARD LAYERS

6. CONVOLUTIONAL LAYERS

6.1. 1×1 *Convolution*

1×1 convolution is a misnomer. Recall that both the input and output activation tensors may have more than 1 channels. 1×1 convolution essentially convolves over all the channels in the input layer and thus there is an implicit hidden dimension in the term 1×1 convolution i.e. the reader should really understand the term 1×1 convolution to mean $1 \times 1 \times D$ convolution where D is the depth of the input tensor.

By convention, the size of the filter in the 3$^{\mathrm{rd}}$ dimension always matches the depth of the input layer. Visualize 1×1 convolution as follows: Imagine the input tensor to be a block of numbers with height, width, and depth equal to $H \times W \times D$. The 1×1 convolution filter is a pencil of numbers that is $1 \times 1 \times D$. Starting at the top right corner of the input tensor, scan the pencil across the height and width of the tensor while keeping the depth of the pencil aligned with the depth of the tensor. The output of 1×1 convolution is just the inner product of the pencil with the vector of numbers from the input tensor that the pencil overlaps with. It is readily apparent that this operation will produce an output tensor of the same height and width as the input (assuming a stride of 1) with depth equal to 1 i.e. a normal matrix.

It is rarely the case that a single $1 \times 1 \times \mathrm{n}^{[l-1]}$ convolution is applied to the input tensor. More often than not, multiple filters i.e. a *filter bank* is applied to the input tensor. If the filter bank contains C filters, the filter bank has dimensions $1 \times 1 \times D \times C$ and produces an output tensor of dimensions $H \times W \times C$ (again assuming unit stride).

Mathematically, 1×1 convolution performs the following operations–

$$\begin{aligned} Z^{[l](m)}_{hwc} &= \sum_{i}^{D} W^{[l]}_{ic} A^{[l-1](m)}_{h'w'i} + b^{[l]}_{c}, \\ A^{[l](m)}_{hwc} &= g^{[l]}(Z^{[l](m)}_{hwc}), \end{aligned} \tag{12}$$

with $h^{'} = hs$ & $w^{'} = ws$ where s is the stride. Notice that since $b^{[l]}_{c}$ carries just a channel index, we have a single value of b for each output channel. As is evident from the Einstein summation convention, equation is really a full set of equations, one per output pixel. In the PYTHON programming language, we would say that we *broadcast* b to the shape of $Z^{[l](m)}_{hwc}$.

The derivatives of the loss function $\mathscr{L}$ for $1 \times 1 \times D$ convolution are given by

$$\frac{\mathrm{d}\mathscr{L}}{\mathrm{d}W^{[l]}_{dc}} = \sum_{m,i,j}^{M,H,W} A^{[l](m)}_{i'j'd} \frac{\partial \mathscr{L}}{\partial Z^{[l](m)}_{ijc}}, \tag{13}$$

$$\frac{\mathrm{d}\mathscr{L}}{\mathrm{d}b_c^{[l]}} = \sum_{m,i,j}^{M,H,W} \frac{\partial \mathscr{L}}{\partial Z_{ijc}^{[l](m)}}, \tag{14}$$

and

$$\frac{\mathrm{d}\mathscr{L}}{\mathrm{d}A_{h'w'd}^{[l-1](m)}} = \sum_{i,j,k}^{H,W,C} W_{kd}^{[l]} \frac{\partial \mathscr{L}}{\partial Z_{ijk}^{[l](m)}} \tag{15}$$

with

$$\frac{\partial \mathscr{L}}{\partial Z_{ijk}^{[l](m)}} = \frac{\partial \mathscr{L}}{\partial A_{ijk}^{[l](m)}} \frac{\mathrm{d}g^{[l]}(Z_{ijk}^{[l](m)})}{\mathrm{d}Z_{ijk}^{[l](m)}}. \tag{16}$$

6.1.1. *Derivation*

We want the total derivative of $\mathscr{L}$ with respect to a given $W_{dc}^{[l]}$ i.e. $\mathrm{d}\mathscr{L}/\mathrm{d}W_{dc}^{[l]}$. Using the chain rule, we may write the total derivative as

$$\frac{\mathrm{d}\mathscr{L}}{\mathrm{d}W_{dc}^{[l]}} = \sum_{m,i,j,k=1}^{M,H,W,C} \frac{\partial \mathscr{L}}{\partial Z_{ijk}^{[l](m)}} \frac{\mathrm{d}Z_{ijk}^{[l](m)}}{\mathrm{d}W_{dc}^{[l]}}.$$

Notice that

$$\frac{\mathrm{d}Z_{ijk}^{[l](m)}}{\mathrm{d}W_{dc}^{[l]}} = \delta_{ck} A_{i'j'd}^{[l](m)},$$

and so

$$\frac{\mathrm{d}\mathscr{L}}{\mathrm{d}W_{dc}^{[l]}} = \sum_{m,i,j,k=1}^{M,H,W,C} \delta_{ck} A_{i'j'd}^{[l](m)} \frac{\partial \mathscr{L}}{\partial Z_{ijk}^{[l](m)}} = \sum_{m,i,j=1}^{M,H,W} A_{i'j'd}^{[l](m)} \frac{\partial \mathscr{L}}{\partial Z_{ijc}^{[l](m)}},$$

Q.E.D.

Similarly,

$$\frac{\mathrm{d}\mathscr{L}}{\mathrm{d}b_c^{[l]}} = \sum_{m,i,j,k=1}^{M,H,W,C} \frac{\partial \mathscr{L}}{\partial Z_{ijk}^{[l](m)}} \frac{\mathrm{d}Z_{ijk}^{[l](m)}}{\mathrm{d}b_c^{[l]}}.$$

Now,

$$\frac{\mathrm{d}Z_{ijk}^{[l](m)}}{\mathrm{d}b_c^{[l]}} = \delta_{ck},$$

and so

$$\frac{\mathrm{d}\mathscr{L}}{\mathrm{d}b_c^{[l]}} = \sum_{m,i,j,k=1}^{M,H,W,C} \delta_{ck} \frac{\partial \mathscr{L}}{\partial Z_{ijk}^{[l](m)}} = \sum_{m,i,j=1}^{M,H,W} \frac{\partial \mathscr{L}}{\partial Z_{ijc}^{[l](m)}},$$

Q.E.D.

Lastly,

$$\frac{\mathrm{d}\mathscr{L}}{\mathrm{d}A_{h'w'd}^{[l-1](m)}} = \sum_{n,i,j,k=1}^{M,H,W,C} \frac{\partial \mathscr{L}}{\partial Z_{ijk}^{[l](n)}} \frac{\mathrm{d}Z_{ijk}^{[l](n)}}{\mathrm{d}A_{h'w'd}^{[l-1](m)}}.$$

However,

$$\frac{\mathrm{d}Z^{[l](n)}_{ijk}}{\mathrm{d}A^{[l-1](m)}_{h'w'd}} = \delta_{mn} W^{[l]}_{kd},$$

and so

$$\frac{\mathrm{d}\mathscr{L}}{\mathrm{d}A^{[l-1](m)}_{h'w'd}} = \sum_{n,i,j,k=1}^{M,H,W,C} \delta_{mn} W^{[l]}_{kd} \frac{\partial \mathscr{L}}{\partial Z^{[l](n)}_{ijk}} = \sum_{i,j,k=1}^{H,W,C} W^{[l]}_{kd} \frac{\partial \mathscr{L}}{\partial Z^{[l](m)}_{ijk}},$$

Q.E.D.

In all three cases,

$$\frac{\partial \mathscr{L}}{\partial Z^{[l](m)}_{ijk}} = \frac{\partial \mathscr{L}}{\partial A^{[l](m)}_{ijk}} \frac{\mathrm{d}A^{[l](m)}_{ijk}}{\mathrm{d}Z^{[l](m)}_{ijk}} = \frac{\partial \mathscr{L}}{\partial A^{[l](m)}_{ijk}} \frac{\mathrm{d}g^{[l]}(Z^{[l](m)}_{ijk})}{\mathrm{d}Z^{[l](m)}_{ijk}}.$$

7. POOLING LAYERS

APPENDIX

REFERENCES