# Richard Mathematical Implementation

Rob Jinman

January 14, 2024

## Dense Layers

### Forward pass

The weighted sums $z_i^l$ for layer $l$ are given by

$$\boxed{z_i^l = \sum_j w_{i,j}^l a_j^{l-1} + b_i^l} \tag{1}$$

and we obtain the activations by applying the sigmoid function

$$\boxed{a_i^l = \sigma(z_i^l)} \tag{2}$$

where

$$\boxed{\sigma(x) = \frac{1}{1 + e^{-1}}} \tag{3}$$

In vectorized form, the forward pass can be written as

$$\boxed{\begin{aligned} \boldsymbol{z}^l &= \boldsymbol{W}^l \boldsymbol{a}^{l-1} + \boldsymbol{b}^l && (4) \\ \boldsymbol{a}^l &= \sigma(\boldsymbol{z}^l) && (5) \end{aligned}}$$

### Backward pass

Given $\partial C / \partial a_i^l$ back-propagated from layer $l + 1$, we calculate the layer delta

$$\boxed{\delta_i^l := \frac{\partial C}{\partial z_i^l}} \tag{6}$$

Applying the chain rule gives

$$\frac{\partial C}{\partial z_i^l} = \frac{\partial C}{\partial a_i^l} \frac{\partial a_i^l}{\partial z_i^l}$$

where $\partial a_i^l / \partial z_i^l$ is the derivative of the sigmoid function, so

$$\boxed{\delta_i^l = \frac{\partial C}{\partial a_i^l} \sigma'(z_i^l)} \tag{7}$$

which in vectorized form is the hadamard product

$$\boxed{\boldsymbol{\delta}^l = \nabla_{\boldsymbol{a}^l} C \odot \sigma'(\boldsymbol{z}^l)} \tag{8}$$

We use this value to compute the cost gradient both with respect to the layer parameters and the layer inputs.

### Cost gradient with respect to parameters

For the weights $w^l_{i,j}$, applying the chain-rule gives

$$\frac{\partial C}{\partial w^l_{i,j}} = \frac{\partial C}{\partial z^l_i}\frac{\partial z^l_i}{\partial w^l_{i,j}} = \delta^l_i \frac{\partial z^l_i}{\partial w^l_{i,j}} = \delta^l_i \frac{\partial \sum_k w^l_{i,k} a^{l-1}_k + b^l_i}{\partial w^l_{i,j}} = \delta^l_i a^{l-1}_j$$

which in vectorized form is the outer product between $\boldsymbol{\delta}^l$ and $\boldsymbol{a}^{l-1}$

$$\boxed{\nabla_{W^l} C = \boldsymbol{\delta}^l \otimes \boldsymbol{a}^{l-1}} \tag{9}$$

For the bias $b^l_i$ we again use the chain rule

$$\frac{\partial C}{\partial b^l_i} = \frac{\partial C}{\partial z^l_i}\frac{\partial z^l_i}{\partial b^l_i} = \delta^l_i \frac{\partial z^l_i}{\partial b^l_i} = \delta^l_i \frac{\partial \sum_k w^l_{i,k} a^{l-1}_k + b^l_i}{\partial b^l_i} = \delta^l_i$$

which in vectorized form is

$$\boxed{\nabla_{b^l} C = \boldsymbol{\delta}^l} \tag{10}$$

Then, given a learning rate $\lambda$, we update the weights and biases

$$\boxed{\begin{aligned} \boldsymbol{W}^l &\leftarrow \boldsymbol{W}^l - \lambda \nabla_{W^l} C & (11)\\ \boldsymbol{b}^l &\leftarrow \boldsymbol{b}^l - \lambda \nabla_{b^l} C & (12)\end{aligned}}$$

### Cost gradient with respect to layer inputs

The multi-variable chain rule gives us

$$\frac{\partial C}{\partial a^{l-1}_i} = \sum_j \frac{\partial C}{\partial z^l_j}\frac{\partial z^l_j}{\partial a^{l-1}_i} = \sum_j \delta^l_j \frac{\partial z^l_j}{\partial a^{l-1}_i} = \sum_j \delta^l_j \frac{\partial \sum_k w^l_{j,k} a^{l-1}_k + b^l_j}{\partial a^{l-1}_i} = \sum_j \delta^l_j w^l_{j,i}$$

which is the layer delta multiplied by the transposed weight matrix

$$\boxed{\nabla_{a^{l-1}} C = (\boldsymbol{W}^l)^T \boldsymbol{\delta}^l} \tag{13}$$

This value is propagated back to layer $l-1$.

# Output Layer

## Forward pass

The forward pass is the same as for ordinary dense layers (see above).

## Backward pass

It's at this final layer $L$ where backpropagation begins. As in equation (8), we have

$$\boxed{\boldsymbol{\delta}^L = \nabla_{a^L} C \odot \sigma'(\boldsymbol{z}^L)} \tag{14}$$

But rather than receive $\nabla_{a^L} C$ from the next layer (there is no next layer), we compute it directly. The cost function $C(\boldsymbol{a}^L, \boldsymbol{y})$ computes the squared error between the network output $\boldsymbol{a}^L$ and the expected network output $\boldsymbol{y}$ for the current sample.

$$\boxed{C(\boldsymbol{a}^L, \boldsymbol{y}) = \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{a}^L\|^2} \tag{15}$$

which in component form is

$$\frac{1}{2}\sum_i \left(a_i^L - y_i\right)^2 = \frac{1}{2}\sum_i \left(\left(a_i^L\right)^2 - 2a_i^L y_i + \left(y_i\right)^2\right)$$

Differentiating with respect to $a_i^L$ we get

$$\boxed{\frac{\partial C}{\partial a_i^L} = a_i^L - y_i} \tag{16}$$

which in vectorized form is

$$\boxed{\nabla_{a^L} C = \boldsymbol{a}^L - \boldsymbol{y}} \tag{17}$$

which we plug into equation (14) to obtain $\boldsymbol{\delta}^L$. Using this value we calculate the gradients for the layer parameters and inputs in the same way as for the dense layers.

# Convolutional Layers

The convolution between functions $f$ and $g$ is

$$\boxed{(f * g)(t) := \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau} \tag{18}$$

The discrete form in 2 dimensions is

$$\boxed{(f * g)(x,y) := \sum_i \sum_j f(i,j)g(x-i, y-j)} \tag{19}$$

which is equivalent to cross-correlation with a horizontally and vertically flipped kernel. The cross-correlation is defined as

$$\boxed{(f \star g)(x,y) := \sum_i \sum_j f(i,j)g(x+i, y+j)} \tag{20}$$

## Forward pass

A layer of depth $D$ generates $D$ feature maps using $D$ kernels and biases. During the forward pass we compute the elements of the feature map $z_{x,y}^{l,d}$ by cross-correlating the 3-dimensional block of input activations $\boldsymbol{A}^{l-1}$ with the 3-dimensional kernel $\boldsymbol{W}^{l,d}$ and adding the bias $b^{l,d}$. We then obtain the activations by applying the ReLU function.

$$\boxed{\begin{aligned} \boldsymbol{Z}^{l,d} &= (\boldsymbol{A}^{l-1} \star \boldsymbol{W}^{l,d}) + b^{l,d} && (21)\\ \boldsymbol{A}^{l,d} &= R(\boldsymbol{Z}^{l,d}) && (22)\end{aligned}}$$

where ReLU is defined as

$$\boxed{R(z) = max(0,z)} \tag{23}$$

In component form, the forward pass looks like

$$\boxed{\begin{aligned} z_{x,y}^{l,d} &= \sum_i \sum_j \sum_k w_{i,j,k}^{l,d} a_{x+i,y+j,k}^{l-1} + b^{l,d} && (24)\\ a_{x,y}^{l,d} &= R(z_{x,y}^{l,d}) && (25)\end{aligned}}$$

## Backward pass

As before, we calculate the layer delta

$$\delta^{l,d}_{x,y} := \frac{\partial C}{\partial z^{l,d}_{x,y}}$$

Applying the chain rule, we get

$$\delta^{l,d}_{x,y} = \frac{\partial C}{\partial a^{l,d}_{x,y}} R'(z^{l,d}_{x,y})$$

where $\partial C / \partial a^{l,d}_{x,y}$ is propagated back from layer $l+1$, and $R'$ is the ReLU derivative.

$$\boxed{R'(z) = \begin{cases} 0 & \text{if } z \leq 0 \\ 1 & \text{if } z > 0 \end{cases}} \tag{26}$$

### Cost gradient with respect to layer inputs

From the chain rule, we have

$$\frac{\partial C}{\partial a^{l-1}_{x,y,z}} = \sum_d \left( \sum_{x'} \sum_{y'} \frac{\partial C}{\partial z^{l,d}_{x',y'}} \frac{\partial z^{l,d}_{x',y'}}{\partial a^{l-1}_{x,y,z}} \right)$$

which is

$$\frac{\partial C}{\partial a^{l-1}_{x,y,z}} = \sum_d \left( \sum_{x'} \sum_{y'} \delta^{l,d}_{x',y'} \frac{\partial z^{l,d}_{x',y'}}{\partial a^{l-1}_{x,y,z}} \right)$$

Expanding the $z^{l,d}_{x',y'}$ term

$$\frac{\partial C}{\partial a^{l-1}_{x,y,z}} = \sum_d \left( \sum_{x'} \sum_{y'} \delta^{l,d}_{x',y'} \frac{\partial \sum_i \sum_j \sum_k w^{l,d}_{i,j,k} a^{l-1}_{x'+i,y'+j,k} + b^{l,d}}{\partial a^{l-1}_{x,y,z}} \right)$$

All terms of the differential vanish except where $x = x' + i$, $y = y' + j$, and $z = k$, or equivalently $i = x - x'$, $j = y - y'$, and $k = z$, therefore

$$\frac{\partial C}{\partial a^{l-1}_{x,y,z}} = \sum_d \left( \sum_{x'} \sum_{y'} \delta^{l,d}_{x',y'} w^{l,d}_{x-x',y-y',z} \right)$$

which amounts to a *full* 2-dimensional convolution between each 2-dimensional kernel slice $\boldsymbol{W}^{l,d}_z$ and feature map delta $\boldsymbol{\delta}^{l,d}$, all added together.

$$\boxed{\nabla_{\boldsymbol{a}^{l-1}_z} C = \sum_d (\boldsymbol{W}^{l,d}_z * \boldsymbol{\delta}^{l,d})} \tag{27}$$

### Cost gradient with respect to layer parameters

From the chain rule, we have

$$\frac{\partial C}{\partial w^{l,d}_{i,j,k}} = \sum_x \sum_y \frac{\partial C}{\partial z^{l,d}_{x,y}} \frac{\partial z^{l,d}_{x,y}}{\partial w^{l,d}_{i,j,k}} = \sum_x \sum_y \delta^{l,d}_{x,y} \frac{\partial z^{l,d}_{x,y}}{\partial w^{l,d}_{i,j,k}}$$

Expanding $z^{l,d}_{x,y}$, we get

$$\frac{\partial C}{\partial w^{l,d}_{i,j,k}} = \sum_x \sum_y \delta^{l,d}_{x,y} \frac{\partial \sum_{i'} \sum_{j'} \sum_{k'} w^{l,d}_{i',j',k'} a^{l-1}_{x+i',y+j',k'} + b^{l,d}}{\partial w^{l,d}_{i,j,k}}$$

All terms in the differential vanish except where $i = i'$, $j = j'$, and $k = k'$, so finally

$$\frac{\partial C}{\partial w^{l,d}_{i,j,k}} = \sum_x \sum_y \delta^{l,d}_{x,y} a^{l-1}_{x+i,y+j,k}$$

which is the cross-correlation of 2-dimensional input slice $\boldsymbol{A}^{l-1}_k$ with 2-dimensional feature map delta $\boldsymbol{\delta}^{l,d}$

$$\boxed{\nabla_{\boldsymbol{W}^{l,d}_k} C = \boldsymbol{A}^{l-1}_k \star \boldsymbol{\delta}^{l,d}} \tag{28}$$

# Max Pooling Layers

Max pooling layers perform a downscale by allowing through only the largest values within each $M \times N$ window.

## Forward pass

For $0 \leq i < M$ and $0 \leq j < N$, the elements of the max pooling layer are given by

$$\boxed{\begin{aligned} z^l_{x,y,z} &= \max_{i,j}(a^{l-1}_{x+i,y+j,z}) && (29) \\ a^l_{x,y,z} &= z^l_{x,y,z} && (30) \end{aligned}}$$

and we don't apply an activation function—or you could say the activation function is just the identity function $f(z) = z$.

## Backward pass

As usual, the layer delta is

$$\delta^l_{x,y,z} := \frac{\partial C}{\partial z^l_{x,y,z}} = \frac{\partial C}{\partial a^l_{x,y,z}} \frac{\partial a^l_{x,y,z}}{\partial z^l_{x,y,z}}$$

There's no activation function, so $z^l_{x,y,z}$ and $a^l_{x,y,z}$ are identical, therefore

$$\boxed{\delta^l_{x,y,z} = \frac{\partial C}{\partial a^l_{x,y,z}}} \tag{31}$$

Or in vectorized form

$$\boxed{\boldsymbol{\delta^l} = \nabla_{A^l} C} \tag{32}$$

which is the value we receive from layer $l + 1$.

### Cost gradient with respect to layer inputs

Each element $a^{l-1}_{x,y,z}$ only contributes to a single element $z^{l}_{\lfloor x/M \rfloor, \lfloor y/N \rfloor, z}$ so there's no need for a summation.

$$\frac{\partial C}{\partial a^{l-1}_{x,y,z}} = \frac{\partial C}{\partial z^{l}_{\lfloor x/M \rfloor, \lfloor y/N \rfloor, z}} \frac{\partial z^{l}_{\lfloor x/M \rfloor, \lfloor y/N \rfloor, z}}{\partial a^{l-1}_{x,y,z}} = \delta^{l}_{\lfloor x/M \rfloor, \lfloor y/N \rfloor, z} \frac{\partial z^{l}_{\lfloor x/M \rfloor, \lfloor y/N \rfloor, z}}{\partial a^{l-1}_{x,y,z}}$$

$$\frac{\partial C}{\partial a^{l-1}_{x,y,z}} = \delta^{l}_{\lfloor x/M \rfloor, \lfloor y/N \rfloor, z} \frac{\partial \max_{i,j}(a^{l-1}_{\lfloor x/M \rfloor + i, \lfloor y/N \rfloor + j, z})}{\partial a^{l-1}_{x,y,z}}$$

$$\boxed{\frac{\partial C}{\partial a^{l-1}_{x,y,z}} = \begin{cases} \delta^{l}_{\lfloor x/M \rfloor, \lfloor y/N \rfloor, z} & \text{if } a^{l-1}_{x,y,z} = \max_{i,j}(a^{l-1}_{\lfloor x/M \rfloor + i, \lfloor y/N \rfloor + j, z}) \\ 0 & \text{otherwise} \end{cases}} \tag{33}$$

So each element $\partial C / \partial a^{l-1}_{x,y,z}$ is equal to $\delta^{l}_{\lfloor x/M \rfloor, \lfloor y/N \rfloor, z}$, but only if its corresponding value in $\boldsymbol{A}^{l-1}$ was the largest within its window during the forward pass.