

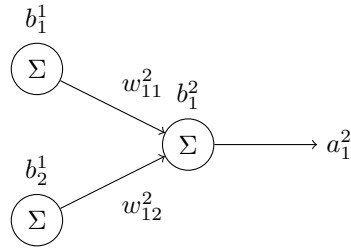
# Fully connected NN

Remi Chassagnol

March 5, 2025

## 1 Notations

- $w_{jk}^l$  is the weight of the  $k^{th}$  neuron in the layer  $(l - 1)$  to the  $j^{th}$  neuron in the layer  $l$
- $b_j^l$  is the bias of the  $j^{th}$  neuron on the layer  $l$ .
- $a_j^l$  is the activation of the  $j^{th}$  neuron on the layer  $l$ .
- $act$  is the activation function.
- $cost$  is the cost function.
- $\delta_j^l$  is the error of the  $j^{th}$  neuron on the layer  $l$ .
- $L$  is the last layer.



## 2 Definitions

Transition from the layer  $(l - 1)$  to the layer  $l$ :

$$z_j^l = \sum_k w_{jk}^l a_k^{l-1} + b_j^l \quad (1)$$

$$a_j^l = act(z_j^l) \quad (2)$$

The error of a neuron can be define as:

$$\delta_j^l \equiv \frac{\partial cost}{\partial z_j^l} \quad (3)$$

We define  $\delta_j^L$  the error on the output layer as the following:

$$\delta_j^L = \frac{\partial cost}{\partial a_j^L} act'(z_j^L) \quad (4)$$

Then we have  $\delta^L$  which is a vector that contains the errors of all the output neurons:

$$\delta^L = \nabla_a cost \odot act'(z^L) \quad (5)$$

Here, we can deduce the value of  $\delta^l$ :

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot act'(z^l) \quad (6)$$

Now that we have this, we can define the error in terms of  $w$  and  $b$ :

$$\frac{\partial cost}{\partial b_j^l} = \delta_j^l \quad (7)$$

$$\frac{\partial cost}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l \quad (8)$$

### 3 Algorithms

First we want to apply the model to the input by making the data flow through the network using the feed forward algorithm.

---

**Algorithm 1** FeedForward

---

```
procedure FEEDFORWARD(weights, biases, input)
  a = input
  as = [a]
  zs = []

  for w, b in (weights, biases) do
    z = w × a + b
    append(zs, z)
    a = act(z)
    append(as, a)
  end for
  return as, zs
end procedure
```

---

Then we compute the gradient of all the nodes using the back propagation (propagate the result error back in the network).

---

**Algorithm 2** Back propagation algorithm

---

```
procedure BACKPROPAGATE(as, zs, weights, biases)
  deltas = cost(as[len(as) - 1], expected_solution) ⊙ act'(zs[len(zs) - 1])
  grads_b[L - 1] = deltas
  grads_w[L - 1] = matmul(deltas, as[len(as) - 2])

  for l in 2.. ≤ L do
    deltas = matmul(T(weights[L - l + 1]), deltas) ⊙ act'(zs[len(zs) - l])
    grads_b[L - l] = deltas
    grads_w[L - l] = matmul(deltas, as[len(as) - l - 1])
  end for
  return grads_w, grads_b
end procedure
```

---

The last component we need is the optimize function that will update the weights and the biases. There are various ways to do this, here is an example for a stochastic gradient descent:

---

**Algorithm 3** Optimization for a stochastic gradient descent

---

```
procedure OPTIMIZE(weights, biases, grads_w, grads_b, l_rate)
  for l in 0.. < L do
    weights[l] -= l_rate × grads_w[l]
    biases[l] -= l_rate × grads_b[l]
  end for
  return weights, biases
end procedure
```

---

To train, we usually use minibatches. This means that we compute the average error on all the minibatches before updating the weights and biases.

---

**Algorithm 4** Update the weights using a minibatch

---

```
procedure UPDATEMINIBATCH(weights, biases, minibatches, l_rate)  
  grads_w_total = 0  
  grads_b_total = 0  
  
  for minibatch in minibatches do  
    as, zs = FeedForward(weights, biases, minibatch)  
    grads_w, grads_b = BackPropagate(as, zs, weights, biases)  
    grads_w_total += grads_w  
    grads_b_total += grads_b  
  end for  
  grads_w = grads_w_total / size(minibatches)  
  grads_b = grads_b_total / size(minibatches)  
  weights, biases = Optimize(weights, biases, grads_w, grads_b, l_rate)  
  
  return weights, biases  
end procedure
```

---

## 4 Proofs

### 4.0.1 Error on the output layer $\delta_j^L$

Let's prove the following:

$$\delta_j^L = \frac{\partial cost}{\partial a_j^L} act'(z_j^L) \quad (9)$$

$$= \frac{\partial cost}{\partial a_j^L} \frac{\partial act}{\partial z_j^L} \quad (10)$$

The definition of the error is:

$$\delta_j^L = \frac{\partial cost}{\partial z_j^L} \quad (11)$$

$$= \sum_k \frac{\partial cost}{\partial a_k^L} \frac{\partial a_k^L}{\partial z_j^L} \quad (12)$$

Since  $a_k^L = act(z_k^L)$ , if  $j \neq k$ , then  $\frac{\partial a_k^L}{\partial z_j^L} = 0$ . Which gives the result expression:

$$\delta_j^L = \frac{\partial cost}{\partial a_j^L} \frac{\partial a_j^L}{\partial z_j^L} \quad (13)$$

### 4.0.2 Error on internal layers $\delta_j^l$

$\delta_j^l$  is defined as:

$$\delta_j^l = \frac{\partial cost}{\partial z_j^l} \quad (14)$$

$$= \sum_k \frac{\partial cost}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial z_j^l} \quad (15)$$

$$= \sum_k \delta_k^{l+1} \frac{\partial z_k^{l+1}}{\partial z_j^l} \quad (16)$$

We can then define  $z_k^{l+1}$  as:

$$z_k^{l+1} = \sum_j w_{kj}^{l+1} act(z_j^l) + b_k^{l+1} \quad (17)$$

so the derivative is:

$$\frac{\partial z_k^{l+1}}{\partial z_j^l} = w_{kj}^{l+1} act'(z_j^l) \quad (18)$$

Substituting back to the original formula:

$$\delta_j^l = \sum_k w_{kj}^{l+1} \delta_k^{l+1} act'(z_j^l) \quad (19)$$

#### 4.0.3 Bias gradient $\frac{\partial cost}{\partial b_j^l}$

We prove the following:

$$\frac{\partial cost}{\partial b_j^l} = \delta_j^l \quad (20)$$

We can rewrite it like the following:

$$\frac{\partial cost}{\partial b_j^l} = \sum_k \frac{\partial cost}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial z_j^l} \frac{\partial z_j^l}{\partial b_j^l} \quad (21)$$

since  $\frac{\partial z_j^l}{\partial b_j^l} = 1$ , then:

$$\frac{\partial cost}{\partial b_j^l} = \sum_k \frac{\partial cost}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial z_j^l} \quad (22)$$

$$= \delta_j^l \quad (23)$$

#### 4.0.4 Bias gradient $\frac{\partial cost}{\partial w_{jk}^l}$

We prove the following:

$$\frac{\partial cost}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l \quad (24)$$

We can rewrite it like the following:

$$\frac{\partial cost}{\partial w_{jk}^l} = \sum_k \frac{\partial cost}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial z_j^l} \frac{\partial z_j^l}{\partial w_{jk}^l} \quad (25)$$

since  $\frac{\partial z_j^l}{\partial w_{jk}^l} = a_k^{l-1}$ , then:

$$\frac{\partial cost}{\partial w_{jk}^l} = \sum_k \frac{\partial cost}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial z_j^l} a_k^{l-1} \quad (26)$$

$$= \delta_j^l a_k^{l-1} \quad (27)$$

## 5 References

- <http://neuralnetworksanddeeplearning.com/chap2.html>