Lecture 4
Neural Network and Backpropagation algorithm

Lê Anh Cường

TDTU - 2020

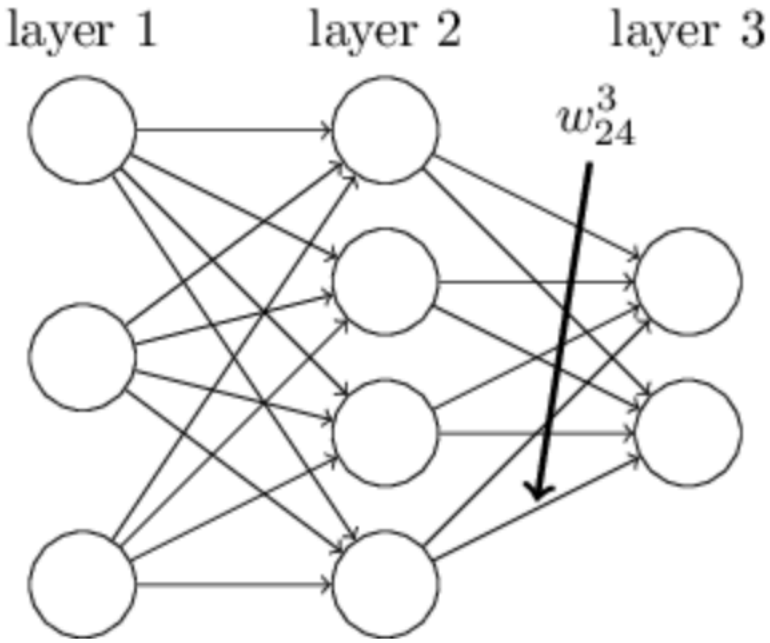
References

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

Objective

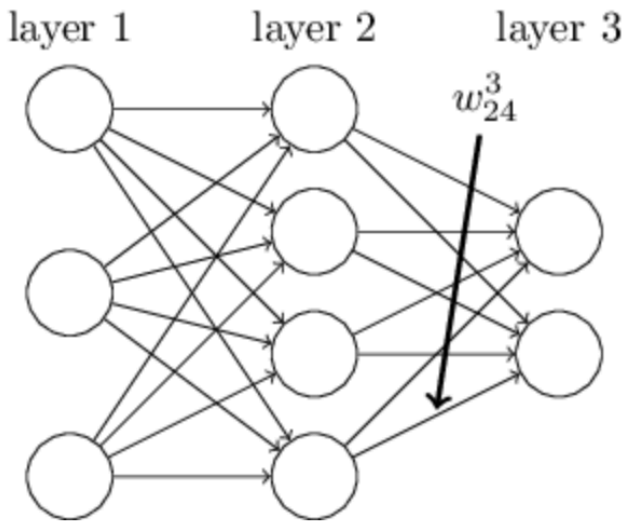
- Neural Networks architecture
- The Backpropagation algorithm

Notations:



w^l_{jk} is the weight from the k^{th} neuron in the $(l-1)^{\text{th}}$ layer to the j^{th} neuron in the l^{th} layer

Formulae:



w^l_{jk} is the weight from the k^{th} neuron in the $(l-1)^{\text{th}}$ layer to the j^{th} neuron in the l^{th} layer

$$a^l_j = \sigma\left(\sum_k w^l_{jk} a^{l-1}_k + b^l_j\right),$$

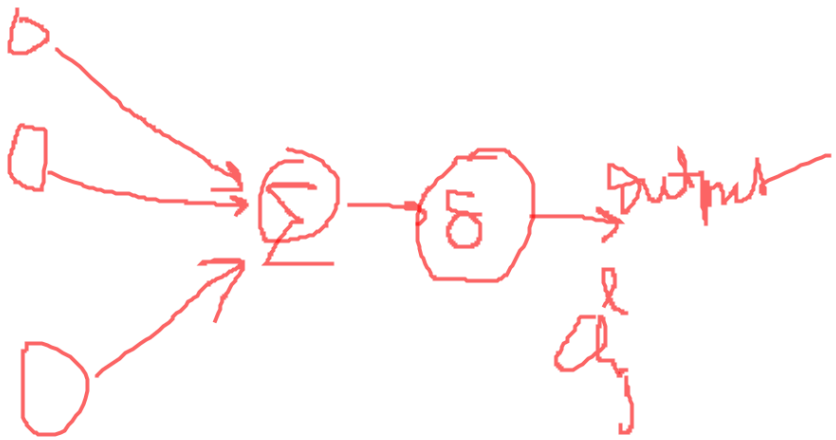
$$a^l = \sigma(w^l a^{l-1} + b^l).$$

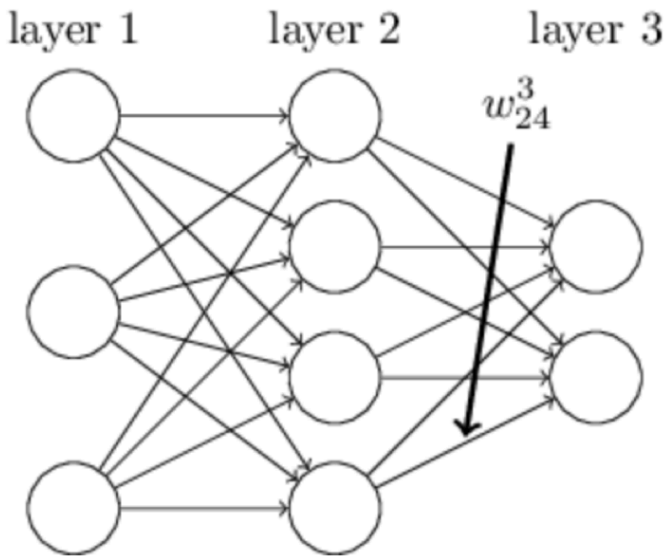
$$z^l_j = \sum_k w^l_{jk} a^{l-1}_k + b^l_j$$

$$z^l \equiv w^l a^{l-1} + b^l$$

$$a^l = \sigma(z^l)$$

Formulae:





w^l_{jk} is the weight from the k^{th} neuron in the $(l-1)^{\text{th}}$ layer to the j^{th} neuron in the l^{th} layer

$$a^l_j = \sigma\left(\sum_k w^l_{jk} a^{l-1}_k + b^l_j\right),$$

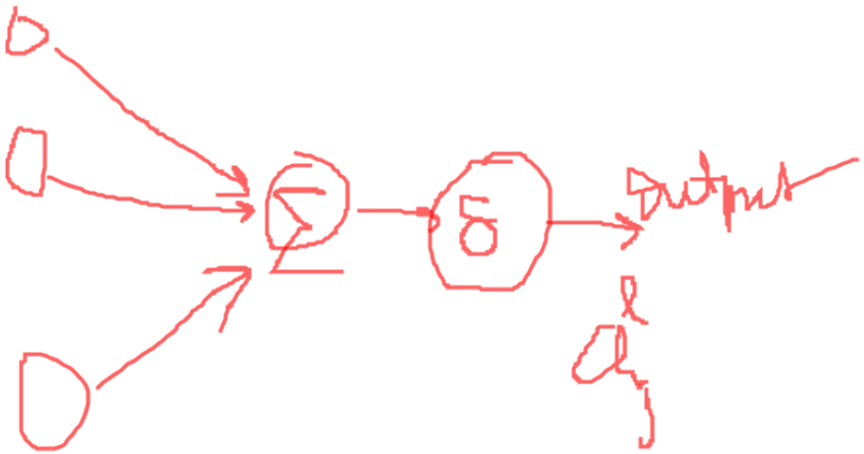
$$a^l = \sigma(w^l a^{l-1} + b^l).$$

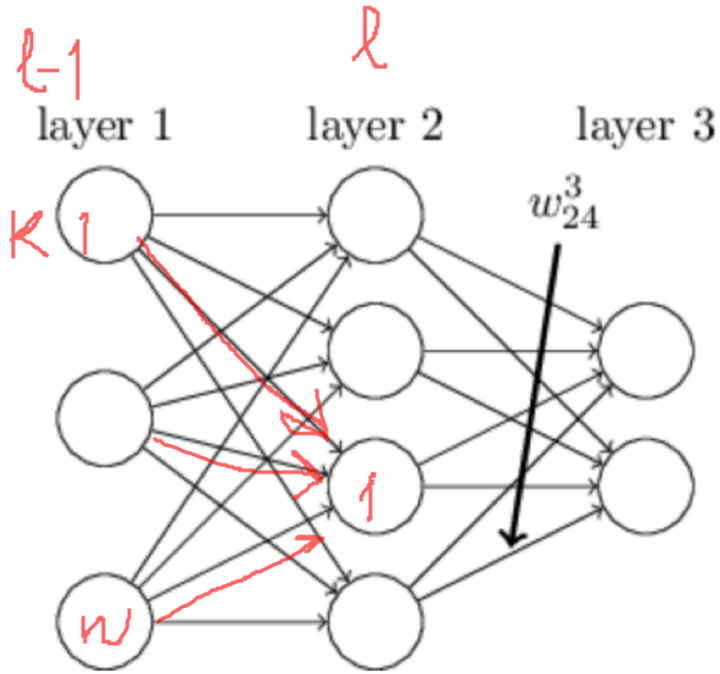
$$z^l_j = \sum_k w^l_{jk} a^{l-1}_k + b^l_j$$

$$z^l \equiv w^l a^{l-1} + b^l$$

$$a^l = \sigma(z^l)$$

Formulae:





w^l_{jk} is the weight from the k^{th} neuron in the $(l-1)^{\text{th}}$ layer to the j^{th} neuron in the l^{th} layer

$$a^l_j = \sigma\left(\sum_k w^l_{jk} a^{l-1}_k + b^l_j\right),$$

$$a^l = \sigma(w^l a^{l-1} + b^l).$$

$$z^l_j = \sum_k w^l_{jk} a^{l-1}_k + b^l_j$$

$$z^l \equiv w^l a^{l-1} + b^l$$

$$a^l = \sigma(z^l)$$

Loss/Cost function

$$C = \frac{1}{2n} \sum_x \|y(x) - a^L(x)\|^2.$$

$$C = \frac{1}{2}\|y - a^L\|^2 = \frac{1}{2}\sum_j (y_j - a_j^L)^2,$$

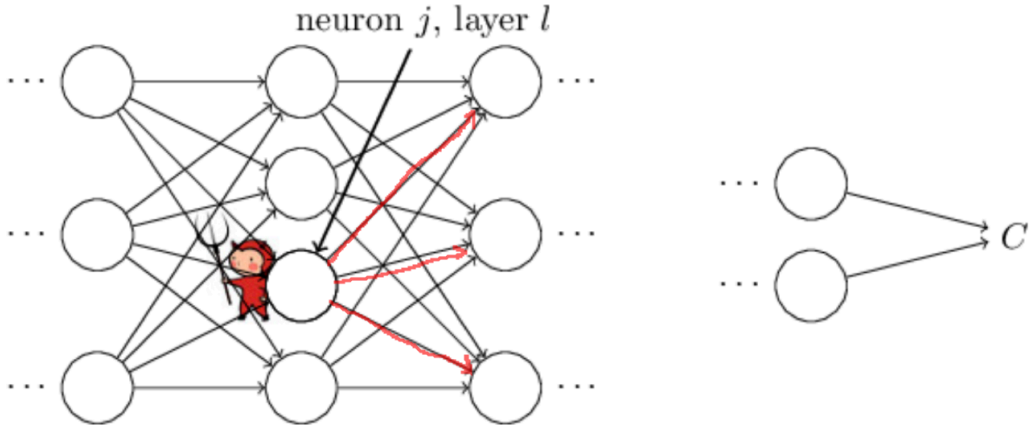
the cost for a single training example is $C_x = \frac{1}{2}\|y - a^L\|^2$.

$$C = \frac{1}{n}\sum_x C_x$$

$$y = y(x)$$

$$a^L = a^L(x)$$

Error propagation

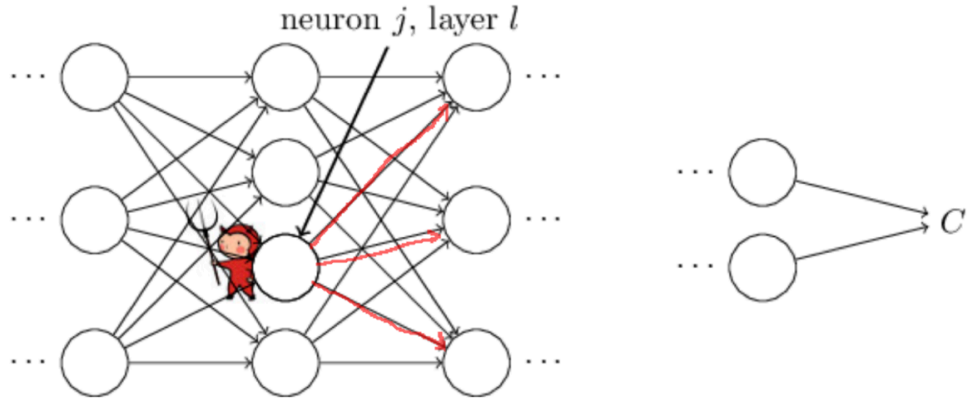


The demon sits at the j^{th} neuron in layer l. As the input to the neuron comes in, the demon messes with the neuron's operation. It adds a little change Δz_j^l to the neuron's weighted input, so that instead of outputting $\sigma(z_j^l)$, the neuron instead outputs $\sigma(z_j^l + \Delta z_j^l)$. This change propagates through later layers in the network, finally causing the overall cost to change by an amount $\frac{\partial C}{\partial z_j^l} \Delta z_j^l$.

$$\frac{\sigma(z_j^l + \Delta z_j^l) - \sigma(z_j^l)}{\Delta z_j^l} \approx \frac{\partial C}{\partial z_j^l}$$

the overall cost to change by an amount $\frac{\partial C}{\partial z_j^l} \Delta z_j^l$.

Error propagation



The demon sits at the j^{th} neuron in layer l. As the input to the neuron comes in, the demon messes with the neuron's operation. It adds a little change Δz_j^l to the neuron's weighted input, so that instead of outputting $\sigma(z_j^l)$, the neuron instead outputs $\sigma(z_j^l + \Delta z_j^l)$. This change propagates through later layers in the network, finally causing the overall cost to change by an amount $\frac{\partial C}{\partial z_j^l} \Delta z_j^l$.

$\sigma(z_j^l + \Delta z_j^l) - \sigma(z_j^l)$

Δz_j^l.

$= \frac{\partial C}{\partial z_j^l}$,

the overall cost to change by an amount $\frac{\partial C}{\partial z_j^l} \Delta z_j^l$.

partial derivatives $\partial C/\partial w^l_{jk}$ and $\partial C/\partial b^l_j$

we define the error δ^l_j of neuron j in layer l

$$\delta^l_j \equiv \frac{\partial C}{\partial z^l_j}.$$

partial derivatives $\partial C/\partial w^l_{jk}$ and $\partial C/\partial b^l_j$

we define the error δ^l_j of neuron j in layer l

$$\delta^l_j \equiv \frac{\partial C}{\partial z^l_j}.$$

$$w^l_{jk} = w^l_{jk} - \mu * \frac{\partial C}{\partial w^l_{jk}}$$

partial derivatives $\partial C/\partial w^l_{jk}$ and $\partial C/\partial b^l_j$

Backpropagation is about understanding how changing the weights and biases in a network changes the cost function. Ultimately, this means computing the partial derivatives $\partial C/\partial w^l_{jk}$ and $\partial C/\partial b^l_j$. But to compute those, we first introduce an intermediate quantity, δ^l_j, which we call the *error* in the j^{th} neuron in the l^{th} layer. Backpropagation will give us a procedure to compute the error δ^l_j, and then will relate δ^l_j to $\partial C/\partial w^l_{jk}$ and $\partial C/\partial b^l_j$.

partial derivatives $\partial C/\partial w^l_{jk}$ and $\partial C/\partial b^l_j$

we define the error δ^l_j of neuron j in layer l

$$\delta^l_j \equiv \frac{\partial C}{\partial z^l_j}.$$

$$a^l = \sigma(z^l)$$

$$a^l_j = \sigma(z^l_j)$$

$$\delta^L_j = \frac{\partial C}{\partial a^L_j}\sigma'(z^L_j).$$

$C = \frac{1}{2}\sum_j (y_j - a^L_j)^2$, and so $\partial C/\partial a^L_j = (a^L_j - y_j)$

$$\delta^L = (a^L - y) \odot \sigma'(z^L).$$

partial derivatives $\partial C/\partial w^l_{jk}$ and $\partial C/\partial b^l_j$

we define the error δ^l_j of neuron j in layer l

$$\delta^l_j \equiv \frac{\partial C}{\partial z^l_j}.$$

$$a^l = \sigma(z^l)$$

$$a^l_j = \sigma(z^l_j)$$

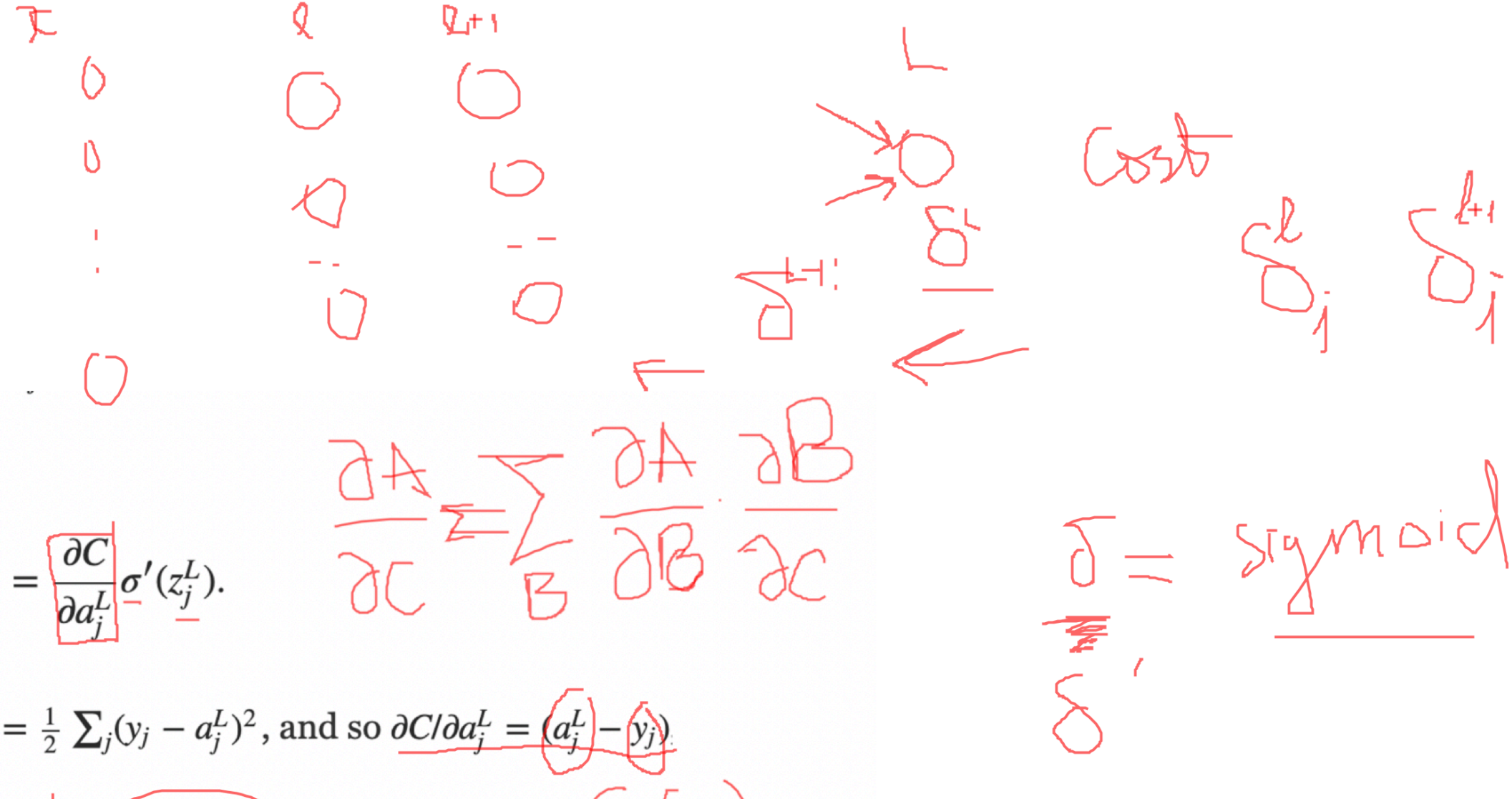
$$\delta^L_j = \frac{\partial C}{\partial a^L_j}\sigma'(z^L_j).$$

$$\frac{\partial A}{\partial C} = \sum_B \frac{\partial A}{\partial B}\cdot\frac{\partial B}{\partial C}$$

$$C = \tfrac{1}{2}\sum_j (y_j - a^L_j)^2, \text{ and so } \partial C/\partial a^L_j = (a^L_j - y_j)$$

$$\delta^L = (a^L - y) \odot \sigma'(z^L).$$

$$C = \delta(z)$$



$$\delta^L_j = \frac{\partial C}{\partial a^L_j} \sigma'(z^L_j).$$

$$C = \tfrac{1}{2} \sum_j (y_j - a^L_j)^2, \text{ and so } \partial C / \partial a^L_j = (a^L_j - y_j).$$

$$\delta^L = (a^L - y) \odot \sigma'(z^L).$$

An equation for the error δ^l in terms of the error in the next layer, δ^{l+1}

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

Suppose we know the error δ^{l+1} at the $l + 1^{\text{th}}$ layer. When we apply the transpose weight matrix, $(w^{l+1})^T$, we can think intuitively of this as moving the error *backward* through the network, giving us some sort of measure of the error at the output of the l^{th} layer. We then take the Hadamard product $\odot\sigma'(z^l)$. This moves the error backward through the activation function in layer l, giving us the error δ^l in the weighted input to layer l.

$$z_k^{l+1} = \sum_j w_{kj}^{l+1} a_j^l + b_k^{l+1} = \sum_j w_{kj}^{l+1} \sigma(z_j^l) + b_k^{l+1}. \tag{43}$$

Differentiating, we obtain

$$\frac{\partial z_k^{l+1}}{\partial z_j^l} = w_{kj}^{l+1} \sigma'(z_j^l). \tag{44}$$

$$\begin{aligned}
\delta_j^l &= \frac{\partial C}{\partial z_j^l} \\
&= \sum_k \frac{\partial C}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial z_j^l} \\
&= \sum_k \frac{\partial z_k^{l+1}}{\partial z_j^l} \delta_k^{l+1},
\end{aligned}$$

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

compute the error δ^l

$$\delta^L = (a^L - y) \odot \sigma'(z^L). \tag{BP1}$$

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l), \tag{BP2}$$

By combining (BP2) with (BP1) we can compute the error δ^l for any layer in the network. We start by using (BP1) to compute δ^L, then apply Equation (BP2) to compute δ^{L-1}, then Equation (BP2) again to compute δ^{L-2}, and so on, all the way back through the network.

An equation for the rate of change of the cost with respect to any weight in the network:

$$\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j$$

An equation for the rate of change of the cost with respect to any weight in the network:

$$\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j$$

$$w^l_{jk} = w^l_{jk} - \mu \cdot \frac{\partial C}{\partial w^l_{jk}}$$

Summary: the equations of backpropagation

$$\delta^L = \nabla_a C \odot \sigma'(z^L) \tag{BP1}$$

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l) \tag{BP2}$$

$$\frac{\partial C}{\partial b^l_j} = \delta^l_j \tag{BP3}$$

$$\frac{\partial C}{\partial w^l_{j\,k}} = a^{l-1}_k \delta^l_j \tag{BP4}$$

$$\delta^L = \nabla_a C \odot \sigma'(z^L).$$

$$\delta^L = (a^L - y) \odot \sigma'(z^L)$$

$$a_j^l = \sigma\left(\sum_k w_{jk}^l a_k^{l-1} + b_j^l\right)$$

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

$$a^l = \sigma(w^l a^{l-1} + b^l)$$

$$z^l \equiv w^l a^{l-1} + b^l$$

$$a^l = \sigma(z^l)$$

$$C = \frac{1}{2n} \sum_x \|y(x) - a^L(x)\|^2$$

$$C = \frac{1}{2} \|y - a^L\|^2 = \frac{1}{2} \sum_j (y_j - a_j^L)^2$$

$$C = \frac{1}{n} \sum_x C_x$$

$$C_x = \frac{1}{2} \|y - a^L\|^2$$

$$a^L = a^L(x)$$

$$y = y(x)$$