Soundness Proof of the Four Fundamental Equations

Let us now show how to derive equations `BP1` through `BP4` .

In the following we will make heavy use of the **chain rule for multivariate calculus**.

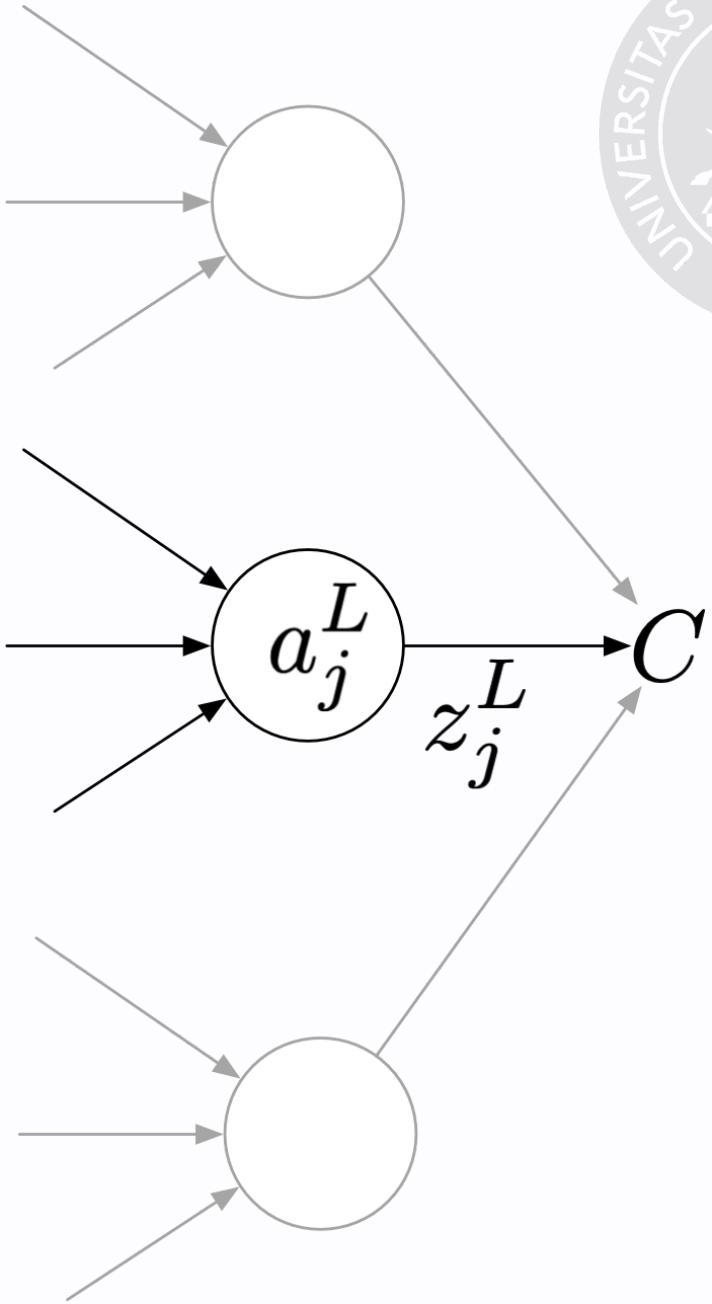
It states that if f is a function of g and h which themselves are a function of x, i.e., if $f(x) = f(g(x), h(x))$, then:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial x} + \frac{\partial f}{\partial h}\frac{\partial h}{\partial x}$$

Proof of BP1

We want to show that:

$$\delta_j^L \equiv \frac{\partial C}{\partial a_j^L} = \frac{\partial C}{\partial z_j^L} \sigma'(a_j^L)$$



By using the fact that C is a function of all output units z_j^L, and applying the chain rule for multivariate functions, we obtain:

$$\delta_j^L = \frac{\partial C}{\partial a_j^L} = \sum_k \frac{\partial C}{\partial z_k^L} \frac{\partial z_k^L}{\partial a_j^L}$$

But since only z_j^L depends on a_j^L all the terms but one are zero, yielding:

$$\delta_j^L = \frac{\partial C}{\partial a_j^L} = \frac{\partial C}{\partial z_j^L} \frac{\partial z_j^L}{\partial a_j^L} = \frac{\partial C}{\partial z_j^L} \sigma'(a_j^L)$$

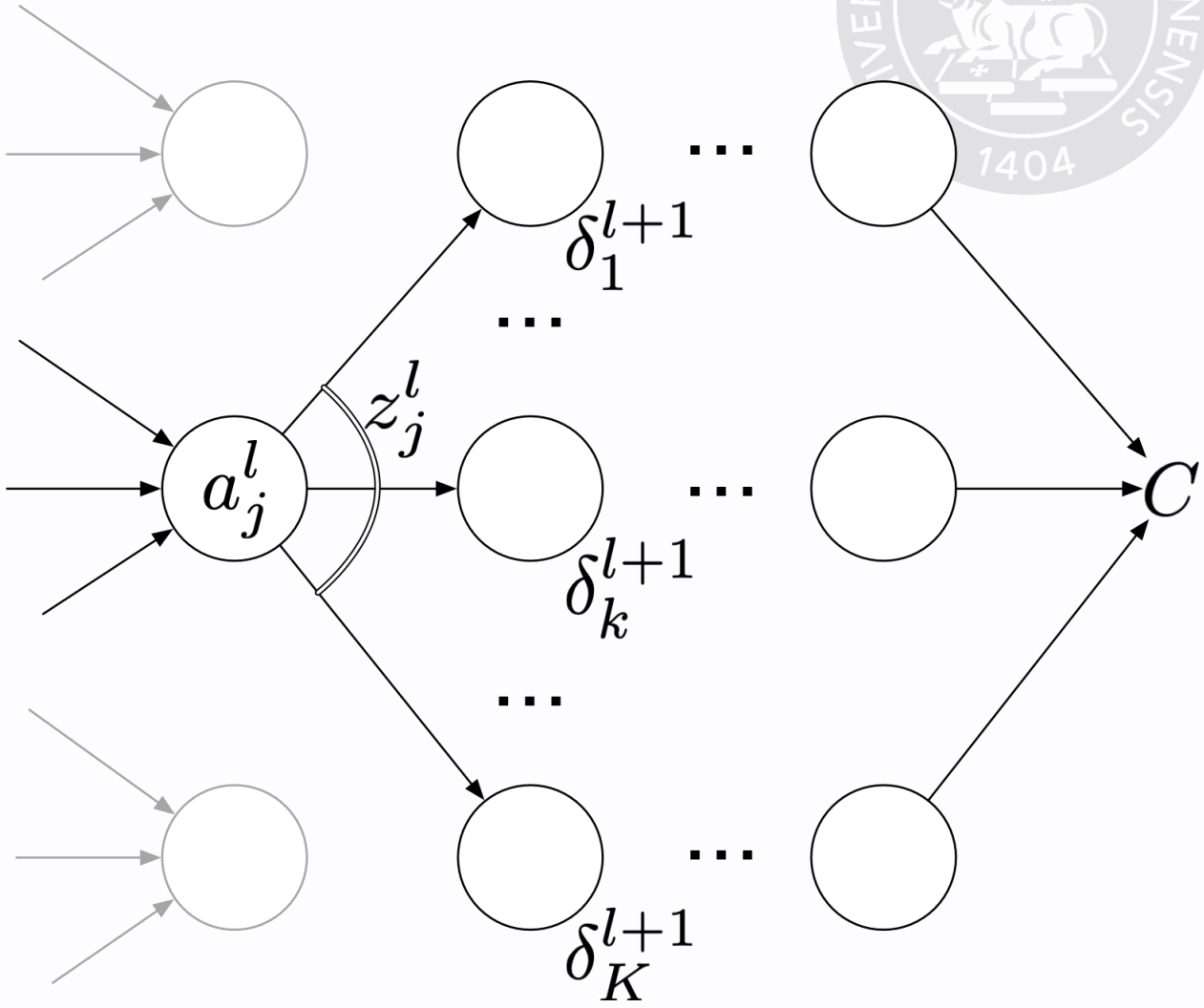
where the last equality derives from remembering that $z_j^L = \sigma(a_j^L)$.

□

Proof of BP2

We want to show that:

$$\begin{aligned}\delta_j^l &= \frac{\partial C}{\partial a_j^l} \\ &= \left((\mathbf{W}^{l+1})^T \boldsymbol{\delta}^{l+1} \odot \sigma'(\mathbf{a}^l)\right)_j \\ &= \sum_k w_{kj}^{l+1} \delta_k^{l+1} \sigma'(a_j^l)\end{aligned}$$



We start by rewriting δ_j^l in terms of δ_k^{l+1}. Again we exploit the chain rule for multivariate functions by observing that C can be thought of as a function of the a_k^{l+1} which in turn can be thought of as functions of a_j^l.

$$\begin{aligned}\delta_j^l &= \frac{\partial C}{\partial a_j^l} \\ &= \sum_k \frac{\partial C}{\partial a_k^{l+1}} \frac{\partial a_k^{l+1}}{\partial a_j^l}\end{aligned}$$

Let us now note that $\frac{\partial C}{\partial a_k^{l+1}} = \delta_k^{l+1}$ by definition, yielding:

$$\delta_j^l = \sum_k \frac{\partial a_k^{l+1}}{\partial a_j^l} \delta_k^{l+1}$$

Let us now recall that:

$$a_k^{l+1} = \sum_j w_{kj}^{l+1} z_j^l + b_k^{l+1} = \sum_j w_{kj}^{l+1} \sigma(a_j^l) + b_k^{l+1}$$

implying that:

$$\frac{\partial a_k^{l+1}}{\partial a_j^l} = w_{kj}^{l+1} \sigma'(a_j^l)$$

Putting all together we get:

$$\delta_j^l = \sum_k w_{kj}^{l+1} \sigma'(a_j^l) \delta_k^{l+1} = \sum_k w_{kj}^{l+1} \delta_k^{l+1} \sigma'(a_j^l)$$

□

Proof of BP3

We want to prove:

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l$$

We start again by interpreting C as a function of the a_j^l and applying the chain rule for multivariate calculus:

$$\frac{\partial C}{\partial b_j^l} = \sum_k \frac{\partial C}{\partial a_k^l} \frac{\partial a_k^l}{\partial b_j^l}$$

By noticing that the only a_k^l that depends on b_j^l is a_j^l, we can simplify it as:

$$\frac{\partial C}{\partial b_j^l} = \frac{\partial C}{\partial a_j^l} \frac{\partial a_j^l}{\partial b_j^l} = \delta_j^l \frac{\partial a_j^l}{\partial b_j^l}$$

Now: $a_j^l = \sum_k w_{jk}^l z_k^{l-1} + b_j^l$ implying that $\frac{\partial a_j^l}{\partial b_j^l} = 1$.

□

Proof of BP4

We want to prove:

$$\frac{\partial C}{\partial w_{jk}^{l}} = z_{k}^{l-1}\delta_{j}^{l}$$

As usual via the chain rule:

$$\frac{\partial C}{\partial w^l_{jk}} = \sum_i \frac{\partial C}{\partial a^l_i}\frac{\partial a^l_i}{\partial w^l_{jk}} = \frac{\partial C}{\partial a^l_j}\frac{\partial a^l_j}{\partial w^l_{jk}} = \delta^l_j \frac{\partial a^l_j}{\partial w^l_{jk}}$$

and the proof is completed by showing that $\frac{\partial a^l_j}{\partial w^l_{jk}} = z^{l-1}_k$.

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

□

The Backpropagation algorithm

Input $\mathbf{x}$: Set the corresponding activation $\mathbf{z}^1$ for the input layer.

1. **Feedforward**: For each $l = 2, 3, \ldots, L$ compute $\mathbf{a}_l = \mathbf{W}^l \mathbf{z}^{l-1} + \mathbf{b}^l$ and $\mathbf{z}^l = \sigma(\mathbf{a}^l)$.
2. **Output error** $\boldsymbol{\delta}^L$: Compute the vector $\boldsymbol{\delta}^L = \nabla_{\mathbf{z}} C \odot \sigma'(\mathbf{a}^L)$.
3. **Backpropagation of the error**: For each $l = L-1, L-2, \ldots, 2$ compute $\boldsymbol{\delta}^l = ((\mathbf{W}^{l+1})^T \boldsymbol{\delta}^{l+1}) \odot \sigma'(\mathbf{a}^l)$.

Output: The gradient of the cost function is given by $\frac{\partial C}{\partial w^l_{jk}} = z^{l-1}_k \delta^l_j$ and $\frac{\partial C}{\partial b^l_j} = \delta^l_j$.

(Stochastic) Gradient Descent + Backpropagation

Backpropagation gives us only the gradient of the error committed on a fixed input. To implement (stochastic) gradient descent we still have to iterate over all examples and average the results.

The following algorithm makes it explicit all the steps needed to iterate over the examples.

Note: to fully implement stochastic gradient descent, one need an additional loop that iterates over the minibatches and one additional loop to iterate through the epochs).

Gradient Descent + Backpropagation

Input a minibatch of training examples of length m.

1. **For each training example $\mathbf{x}$**: Set the corresponding input activation $\mathbf{z}^{\mathbf{x},1}$, and perform the following steps:
   - **Feedforward**: For each l=2,3,...,L compute $\mathbf{a}^{\mathbf{x},l} = \mathbf{W}^l \mathbf{z}^{\mathbf{x},l-1} + \mathbf{b}^l$ and $\mathbf{z}^{\mathbf{x},l} = \sigma(\mathbf{a}^{\mathbf{x},l})$.
   - **Output error $\boldsymbol{\delta}^{\mathbf{x},L}$**: Compute the vector $\boldsymbol{\delta}^{\mathbf{x},L} = \nabla_{\mathbf{z}} C_{\mathbf{x}} \odot \sigma'(\mathbf{a}^{\mathbf{x},l})$.
   - **Backpropagate the error**: For each $l = L-1, L-2, \ldots, 2$ compute $\boldsymbol{\delta}^{\mathbf{x},l} = ((\mathbf{W}^{l+1})^T \boldsymbol{\delta}^{\mathbf{x},l+1}) \odot \sigma'(\mathbf{a}^{\mathbf{x},l})$.

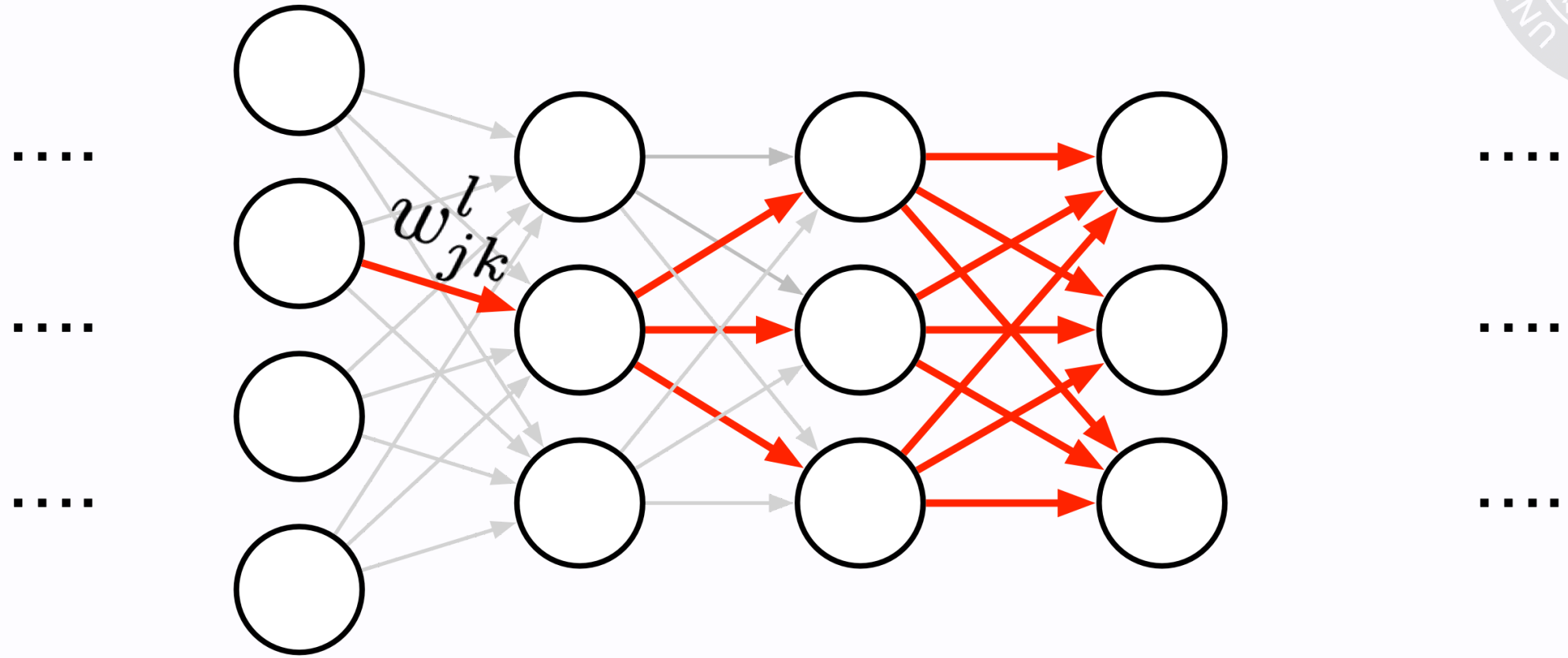
2. **Gradient Descent**: For each $l = L, L-1, \ldots, 2$ update the weights according to the rule $\mathbf{W}^l \leftarrow \mathbf{W}^l - \frac{\eta}{m} \sum_{\mathbf{x}} \boldsymbol{\delta}^{\mathbf{x},l} (\mathbf{z}^{\mathbf{x},l-1})^T$, and the biases according to the rule $\mathbf{b}^l \leftarrow \mathbf{b}^l - \frac{\eta}{m} \sum_{\mathbf{x}} \boldsymbol{\delta}^{\mathbf{x},l}$.

Backpropagation efficiency

To fully apreciate how efficient backpropagation is consider the following:

- A numerical approximation of the derivative based on computing $\frac{\partial C}{\partial w_j} \approx \frac{C(w+\epsilon e_j)-C(w)}{\epsilon}$, would require to compute the approximation for each possible weight (they can be millions) each time performing a forward pass to compute C with the new weighs.
- The computation of the exact derivative performed by backpropagation requires a single forward pass + a single backward pass.

- The algorithm specification makes it trivial to exploit dynamic programming to avoid an exponential number of computations: the influence of nodes on later layers onto nodes of the l-th layer is totally captured by the $\boldsymbol{\delta}^{l+1}$ vector.
- If this was not the case, to compute the derivative of the cost w.r.t. a weight w_{jk}^{l} one would need to integrate the effects over the exponential number of paths that connect that node k at level to $l-1$ to the output node.


w^l_{jk}