The backpropagation algorithm is simply the *instantiation of the gradient descent* technique to the *case of neural networks*. Specifically, it provides iterative rules to compute partial derivatives of the loss function with respect to each parameter of the network. In the following, we shall discuss it in the case of dense networks and shall hint to extensions to convolutional networks and recurrent networks at proper places.

In the previous lesson we computed the gradient by hand for a simple linear net. But a neural network computes a complex function obtained by composition of many neural layers. How can we compute the gradient w.r.t. a specific parameter (weight) of the net? We need a mathematical rule known as the chain rule (for derivatives).

The chain rule

Given two derivable functions f, g with derivatives f' and g' , the derivative of the composite function $h(x) = f(g(x))$ is

$$h'(x) = f'(g(x)) * g'(x))$$

Equivalently, letting $y = g(x)$,

$$h'(x) = f'(g(x)) * g'(x) = f'(y) * g'(x) = \frac{df}{dy} * \frac{dg}{dx}$$

Given a multivariable function f (x, y) and two single variable functions x(t) and y(t)

$$\underbrace{\frac{d}{dt} f(x(t), y(t))}_{\text{derivative of composition}} = \frac{\partial f}{\partial x}\frac{dx}{dt} + \frac{\partial f}{\partial y}\frac{dy}{dt}$$

In vector notation: let ...

$a^l = \sigma(b^l + w^l \cdot x^l)$ a^l is the activation vector at layer

The derivative must be computed at the start of the backpropagation process.