Loss function

can be sum of squared errors

$$L = \sum_{i=1}^{n} (y_i - \hat{y}_i)$$

y_i — given training value

$\hat{y}_i$ — prediction

chain rule

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z} \cdot \frac{\partial z}{\partial W}$$

, where $z = Wx + b$

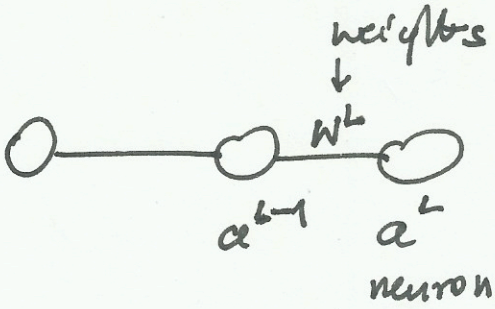
W — weights, b — biases

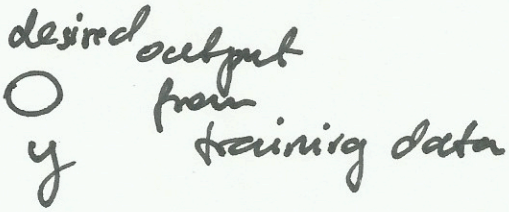
$\frac{\partial L}{\partial W}$ — derivative by weights of neural network layers

After transformation

$$\frac{\partial L}{\partial W} = 2(y - \hat{y}) \cdot \text{derivative\_sign}(z) \cdot \text{layer}^{L-1}$$

proof on the next page





cost function

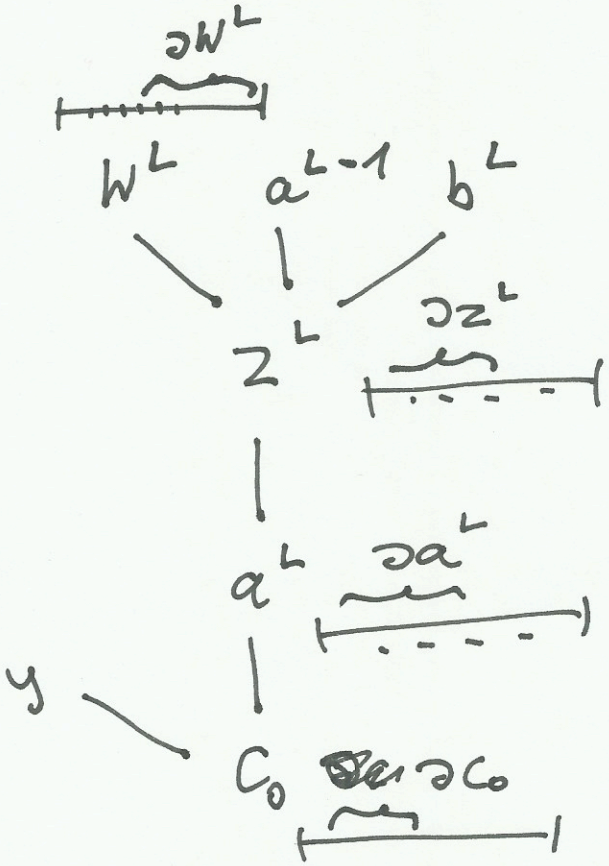
$$C_0 = (a^L - y)^2$$

value of neuron L $= z^L$

$$a^L = \sigma\left(\underbrace{W^L \cdot a^{L-1} + b^L}_{z^L}\right) \quad ; \quad W^L \cdot a^{L-1} + b^L = z^L$$

σ: activation function (here sigmoid)

b^L: bias



we want to know how changes in weights influence cost function (in other words prediction accuracy)

$$\frac{\partial C_0}{\partial W^L} = \frac{\partial z^L}{\partial W^L} \frac{\partial a^L}{\partial z^L} \frac{\partial C_0}{\partial a^L}$$

as per chain rule

$$\frac{\partial C_0}{\partial a^L} = \frac{\partial (a^L - y)^2}{\partial a^L} =$$

$$= 2(a^L - y)$$

$$\frac{\partial a^L}{\partial z^L} = \frac{\partial \sigma(z^L)}{\partial z^L} = \text{sigmoid\_deriv}(z^L)$$

$$\frac{\partial z^L}{\partial w^L} = \frac{\partial (w^L \cdot a^{L-1} + b^L)}{\partial w^L} = a^{L-1}$$

so

$$\frac{\partial C_0}{\partial w^L} = 2(a^L - y) \cdot \text{sigm\_deriv}(z^L) \cdot a^{L-1}$$

for refresh on page 4

A) output to hidden layers

a^{L-1} = layer I

$z^L = w^L a^{L-1} + b^L$, assuming that $b^L = 0$ we have

$z^L = w^L \cdot a^{L-1}$ = output

that's why

layer1.T o 2(output − y) · sigm_deriv.(output)

makes sense layer 1 position next to front

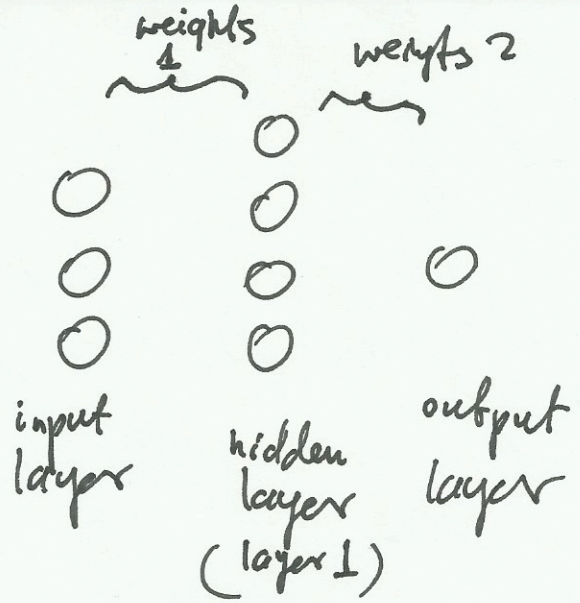
because of numpy dot function

B) hidden layer to input

a^{L-1} = input

by analogy z^L = layer 1

input.T o { difference between prediction and true value moved by one layer · sigmoid_deriv (layer 1) } → see p. 4



change of weights 2 :

$$d_2 = \text{layer}1.T \circ 2(y-\hat{y}) \cdot \text{sigmoid\_derivative}(\hat{y})$$

(slope)

↑ values kept in neurons at layer 1 (must be transposed to correctly do operation on vectors)

$$\text{weights}2 = \text{weights}2 + d_2$$

change of weights 1 :

$$d_1 = \text{input}.T \circ \left\{ \left[\overbrace{2(y-\hat{y}) \cdot \text{sigmoid\_derivative}(\hat{y})}^{\text{difference between network prediction and the true value}} \circ \text{weights}2.T \right] \cdot \text{sigmoid\_derivative}(\text{layer}1) \right\}$$

moved to layer 1 by weights 2 multiplication