So far - regression → gradient descent
→ Normal equations
→ polynomial feature + over fitting
→ bias variance trade off
→ regularization (Best subset selection
LASSO , Ridge)

Supervised learning

- classification
  - least squares (binary + multi class)
  - → probabilistic classifiers
    - → discriminative classifiers (logistic regression)
    - → generative classifiers
      (Gaussian Discriminant Analysis)
  - → perceptron (activation = step function)

Neural Networks

→ Motivation (XOR classification)

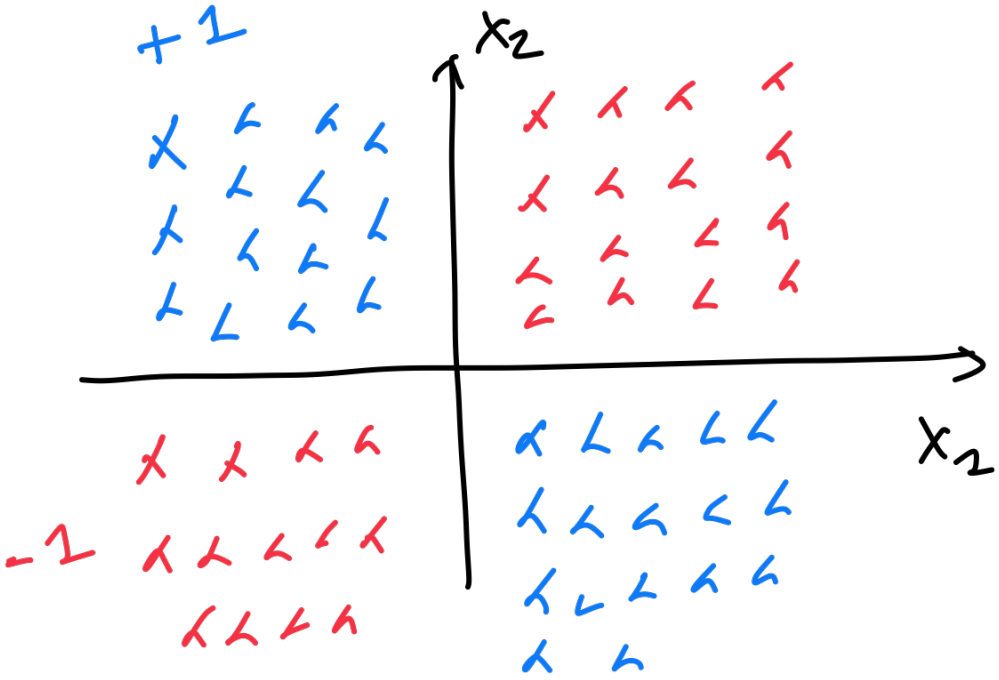
→ general architecture

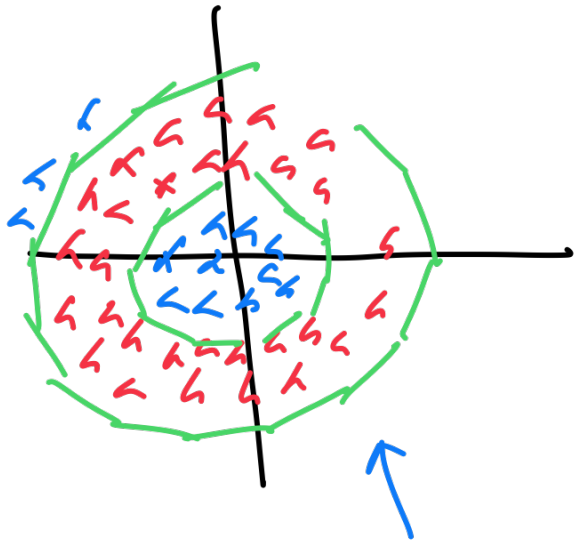
→ Training through backpropagation

linear classifier $y(x) = \sigma(\beta^T \tilde{x})$ $\sigma(x) = x$

logistic regression $y(x) = \sigma(\beta^T \tilde{x})$ $\sigma(x) = \frac{1}{1+e^{-x}}$

perceptron $y(x) = \sigma(\beta^T \tilde{x})$ $\sigma(x) = \begin{cases} +1 & x \geq 0 \\ -1 & x < 0 \end{cases}$

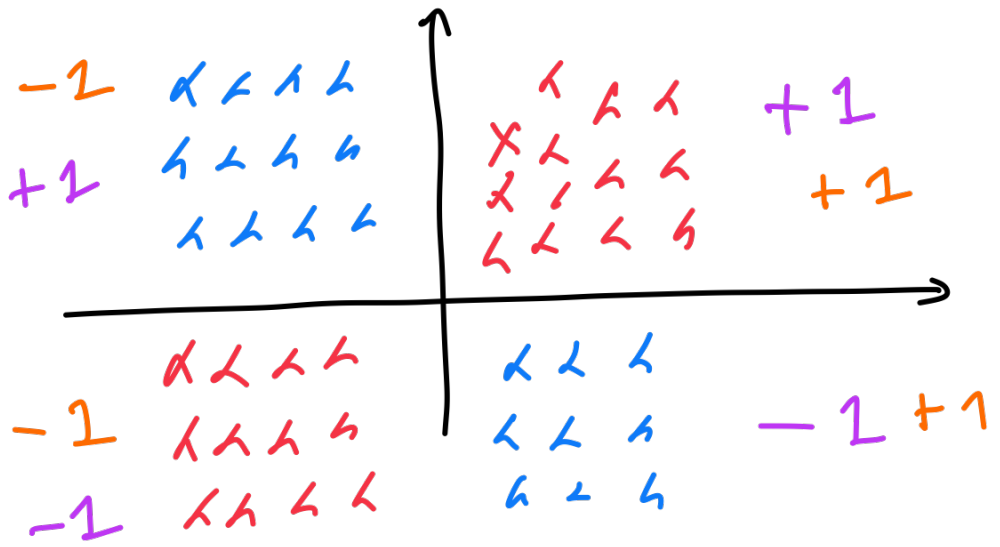




general form $\sigma(\beta^T x)$

classifier 1 $\sigma(x_1) = \begin{cases} +1 & \text{if } x_1 \geq 0 \\ -1 & x_1 < 0 \end{cases}$

classifier 2 $\sigma(x_2) = \begin{cases} +1 & x_2 \geq 0 \\ -1 & x_2 < 0 \end{cases}$



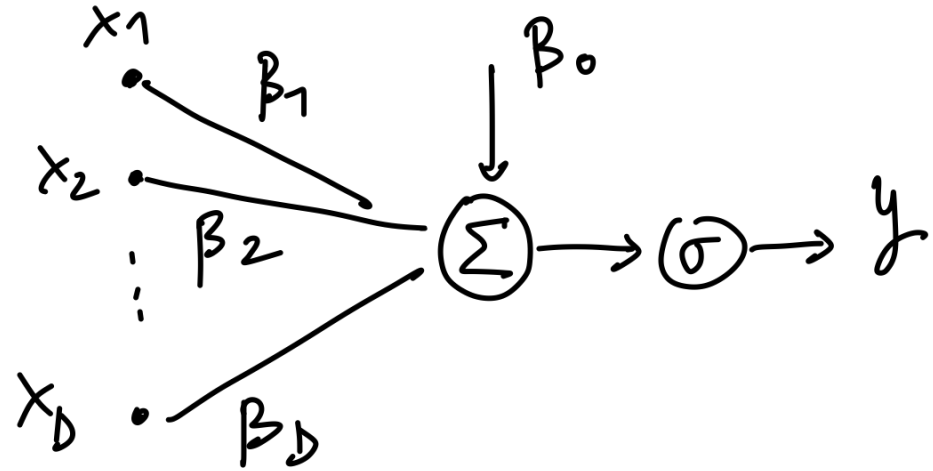
$\frac{1}{2}\widetilde{\sigma}(\sigma(x_1)+\sigma(x_2)) = \begin{cases} 1 & \text{if} \\ 0 & \text{if} \end{cases}$

e.g with $\widetilde{\sigma}(y) = |y|$

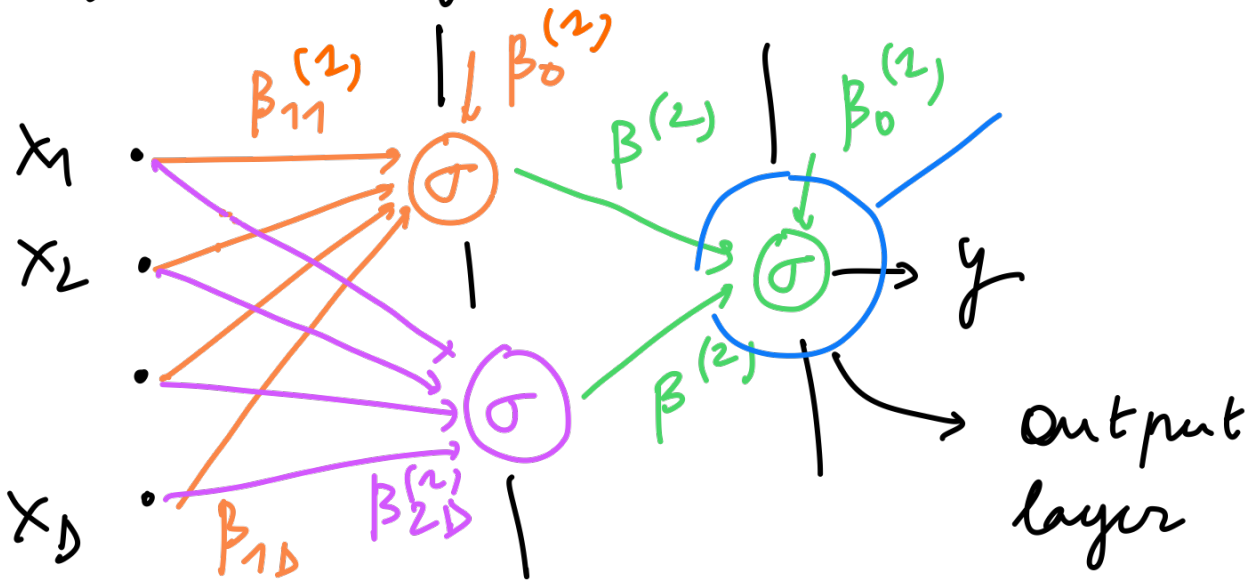
→ We defined a general classifier for a non linearly separable dataset by cascading linear models of the form $\sigma(\beta^T \tilde{x})$

$\sigma(\beta^T \tilde{x})$

$x \in \mathbb{R}^D$



How about our general $\tilde{\sigma}(\sigma(x_1) + \sigma(x_2))$? $x \in \mathbb{R}^D$

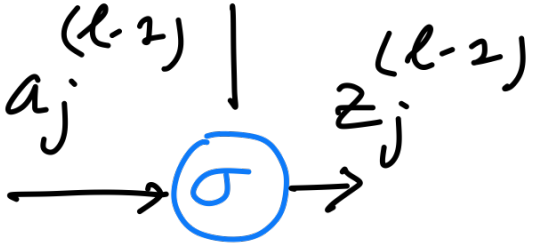


Let $w_{ij}^{(l)}$ be the j^{th} weight of neuron / unit i from the l^{th} layer.

Let $z_j^{(l-2)}$ be the output to the j^{th} neuron from $l-2$ th layer.

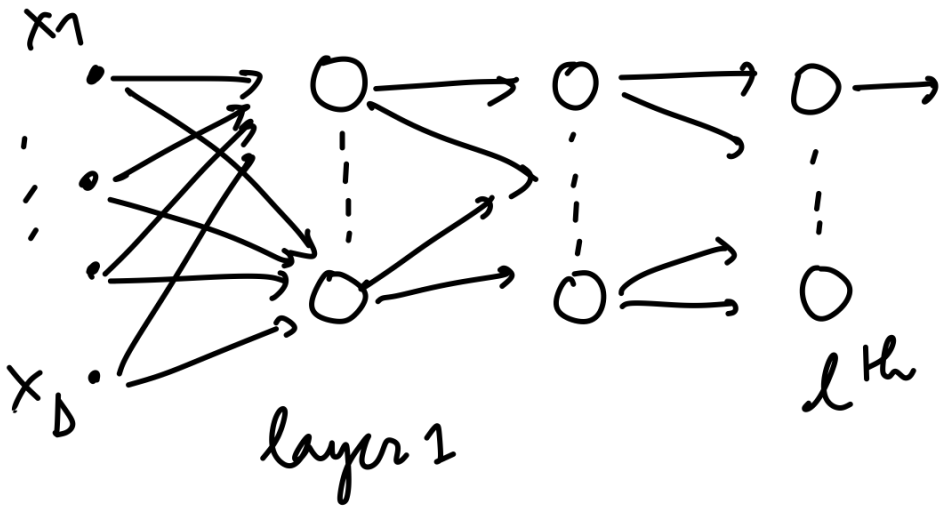
Let $a_j^{(l-1)}$ to be the input to the activation of the j^{th} neuron from layer $l-1$

$$z_j^{(l-2)} = \sigma(a_j^{(l-2)})$$



$$a_i^{(l)} = \sum_j w_{ij}^{(l)} z_j^{(l-2)}$$

) result of i^{th} neuron



$$\sigma\left(a_i^{(l)}\right) = \sigma\left(\sum_j w_{ij}^{(l)} z_j^{(l-1)} + w_{0i}^{(l)}\right)$$

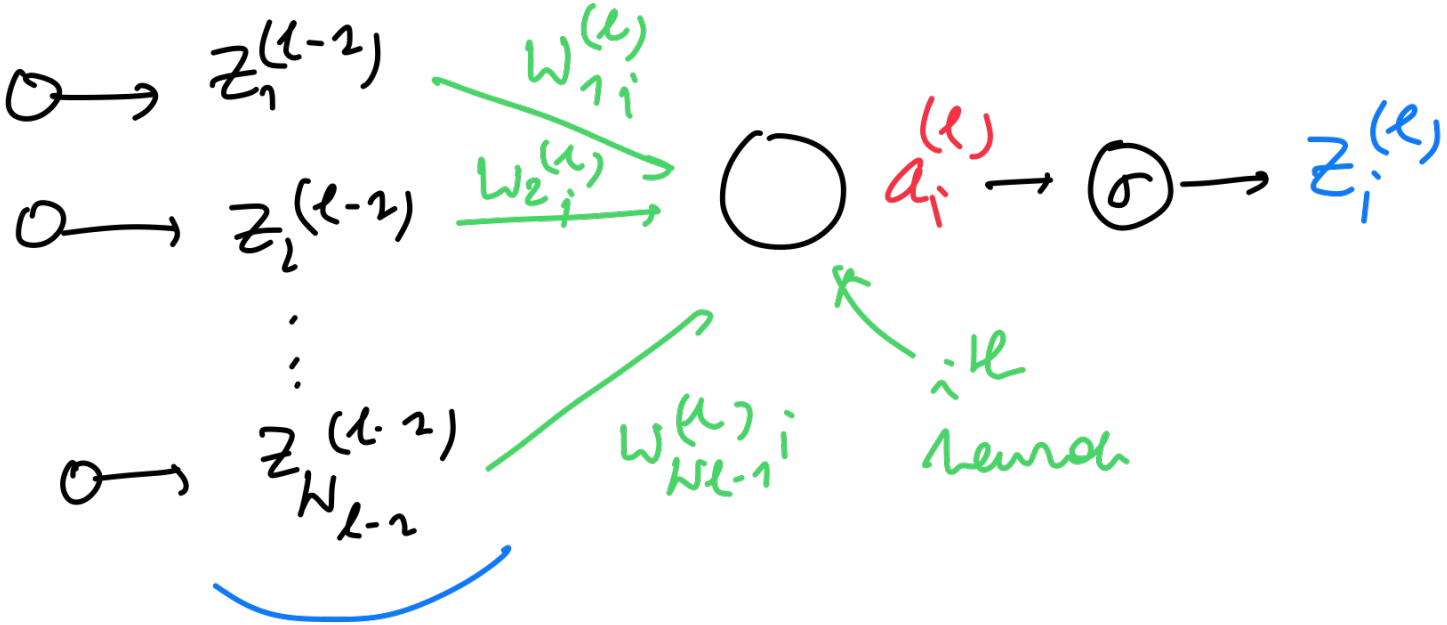
$$\sigma\left(W^{(2)}\left(\sigma\left(W^{(1)}X + w_0^{(0)}\right) + w_0^{(1)}\right)\right)$$

$$y(x) = \sigma\left(w_0^{(L)} + \sum_{j=1}^{N_L} w_{ij}^{(L)} z_j^{(L-1)}\right)$$

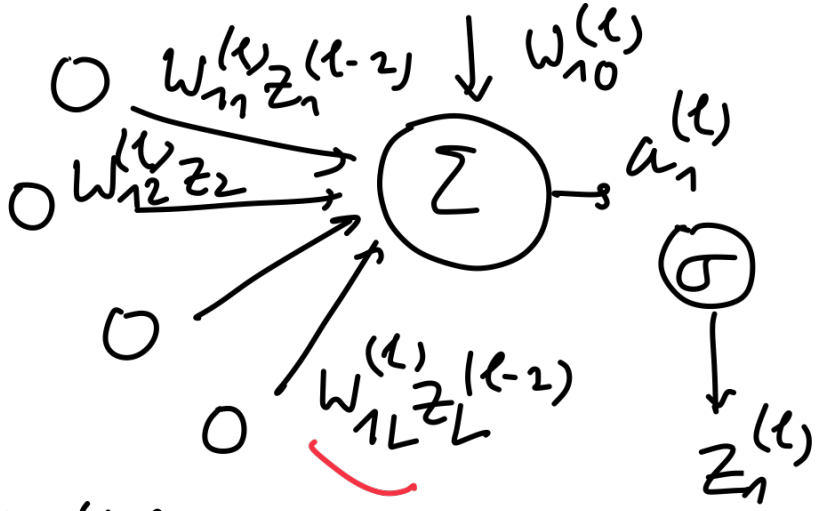
$$z_j^{(L-1)} = \sigma\left(w_{0j}^{(L-1)} + \sum_{k=1}^{N_{L-2}} w_{jk}^{(L-2)} z_k^{(L-2)}\right)$$

....

for any neuron



General Formulation of NNets



$$y(x) = \sigma\left(w_0^L + \sum_{j=1}^{N_L} w_{ij}^L \, z_j^{(L-1)} \right)$$

$$z_j^{(L-1)} = \sigma\left(w_{0j}^{(L-1)} + \sum_{k=1}^{N_{L-2}} w_{jk}^{(L-2)} \, z_k^{(L-2)} \right)$$

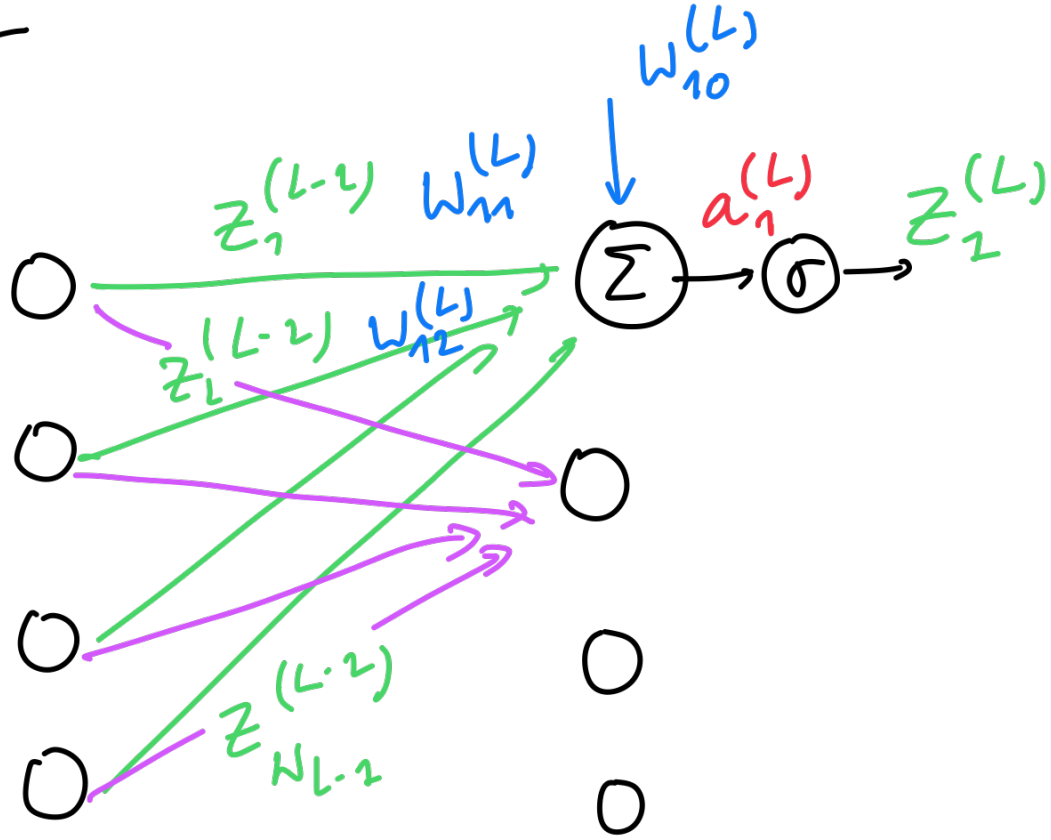
$$\vdots$$

for the ith neuron of layer L, we use $w_{i0}^{(L)}$ for the bias/intercept

We use $w_{ij}^{(L)}$ for the weights associated to that neuron

We use $a_i^{(L)}$ is the preactivation of the i^{th} neuron in layer L

We use $z_i^{(L)}$ is the postactivation of the i^{th} neuron in layer L



How do we train the network ?

just as for logistic regression, we define the probability to be in each class as

$$p(t(x^{(i)}) = 1 \mid x^{(i)}) = y(x^{(i)})$$

$$p(t(x^{(i)}) = 0 \mid x^{(i)}) = 1 - y(x^{(i)})$$

Write down the probability to observe the pairs $\{x^{(i)}, t^{(i)}\}_{i=1}^{N}$, take the log and get the binary cross entropy

$$\mathcal{L} = -\sum_{i=1}^{N} t^{(i)} \log(y(x^{(i)})) + (1 - t^{(i)}) \log(1 - y(x^{(i)}))$$

We then want to do gradient descent on the binary cross entropy loss.

gradient is given by $\left[\frac{\partial \mathcal{L}}{\partial w_{ij}^{(l)}}\right]_{ij}^{l}$

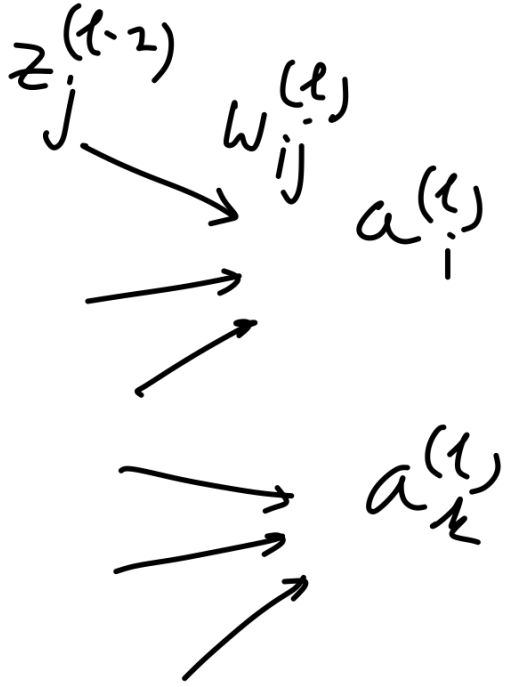
L = total number of layers

l = index of each layer

$$\frac{\partial \mathcal{L}}{\partial w_{ij}^{(l)}} = \frac{\partial \mathcal{L}}{\partial a_i^{(l)}} \cdot \frac{\partial a_i^{(l)}}{\partial w_{ij}^{(l)}}$$

$\frac{\partial \mathcal{L}}{\partial a_i^{(l)}} = \delta_i^{(l)}$

$\frac{\partial a_i^{(l)}}{\partial w_{ij}^{(l)}} = z_j^{(l-1)}$



$\delta_{out} = \frac{\partial \mathcal{L}}{\partial a_{out}}$ where a_{out} = pre activation of last layer

$\frac{\partial \mathcal{L}}{\partial a_{out}} \Rightarrow$ let us do it in a Stochastic Setting (taking one sample at each iteration)

$$\mathcal{L} = -t^{(i)} \log\left(y(x^{(i)})\right) - (1 - t^{(i)}) \log\left(1 - y(x^{(i)})\right)$$

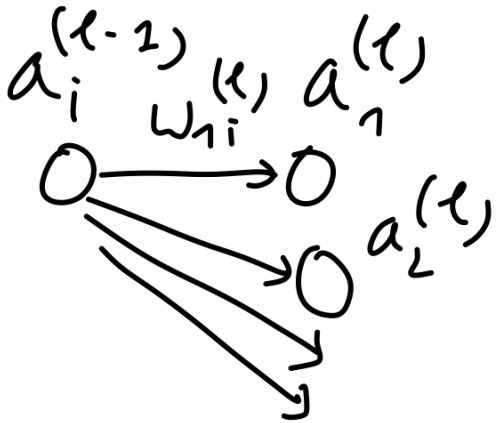
$$= -t^{(i)} \log\left(\sigma(a_{out})\right) - (1 - t^{(i)}) \log\left(1 - \sigma(a_{out})\right)$$

$$\frac{\partial \mathcal{L}}{\partial a_{out}} = -t^{(i)} \frac{\sigma(a_{out})(1 - \sigma(a_{out}))}{\sigma(a_{out})} + (1 - t^{(i)}) \frac{\sigma(1 - \sigma)}{(1 - \sigma)}$$

$$= -t^{(i)}(1 - \sigma) + (1 - t^{(i)})\sigma$$

$$\frac{\partial \mathcal{L}}{\partial a_{out}} = \sigma(a_{out}) - t^{(i)} = \delta_{out}$$

How can we get $\delta_i^{(l)}$ from δ_{out} ?



use $$\frac{\partial \mathcal{L}}{\partial a_i^{(l-1)}} = \sum_{j=1}^{N_l} \underbrace{\frac{\partial \mathcal{L}}{\partial a_j^{(l)}}}_{\delta_j^{(l)}} \frac{\partial a_j^{(l)}}{\partial a_i^{(l-1)}}$$

$$\frac{\partial a_j^{(l)}}{\partial a_i^{(l-1)}} \qquad a_j^{(l)} = \sum_{i=1}^{N_l} w_{ji}^{(l)} \cdot z_i^{(l-1)} + w_{j0}^{(1)}$$

$$a_j^{(l)} = \sum_{i=1}^{N_l} w_{ji}^{(l)} \underbrace{\sigma(a_i^{(l-1)})} + w_{j0}^{(l)}$$

$$\frac{\partial a_j^{(l)}}{\partial a_i^{(l-1)}} = w_{ji}^{(l)} \, \sigma'(a_i^{(l-1)})$$

$$\frac{\partial \mathcal{L}}{\partial a_i^{(l-1)}} = \sum_{j=1}^{N_l} \delta_j^{(l)} \, w_{ji}^{(l)} \, \sigma'(a_i^{(l-1)})$$

$$\delta_i^{(l-1)} = \sum_{j=1}^{N_l} \delta_j^{(l)} \, w_{ji}^{(l)} \, \sigma'(a_i^{(l-1)}) \quad (*)$$

Backprop:

1) forward propagate $x^{(i)}$ through NN and get all $z_i^{(l)}$, $a_i^{(l)}$, for all layers all neurons

2) Compute $\delta^{out} = y(x^{(i)}) - t^{(i)}$

3) back propagate to get $\delta_i^{(l)}$ for all l and i using (*)

4) get the gradient as $\frac{\partial \mathcal{L}}{\partial w_{ij}^{(l)}} = \delta_i^{(l)} z_j^{(l-1)}$