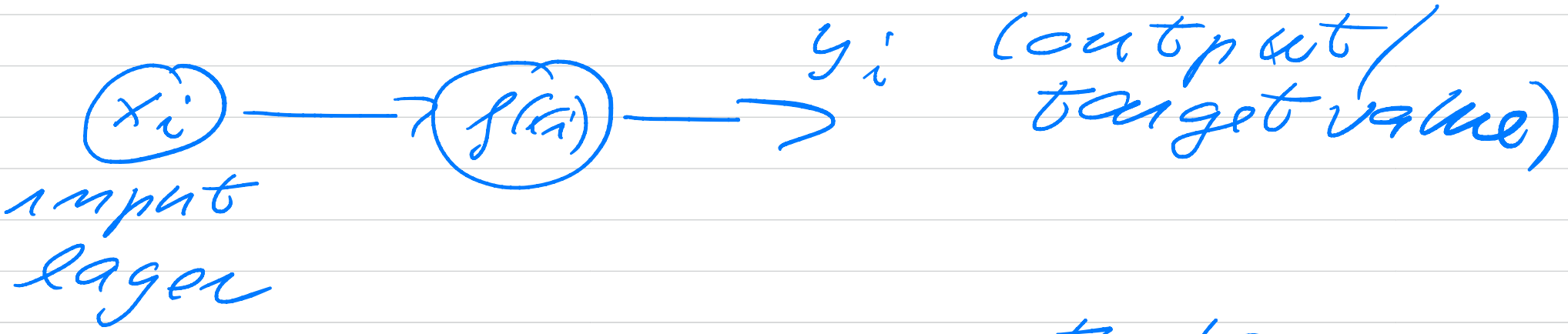
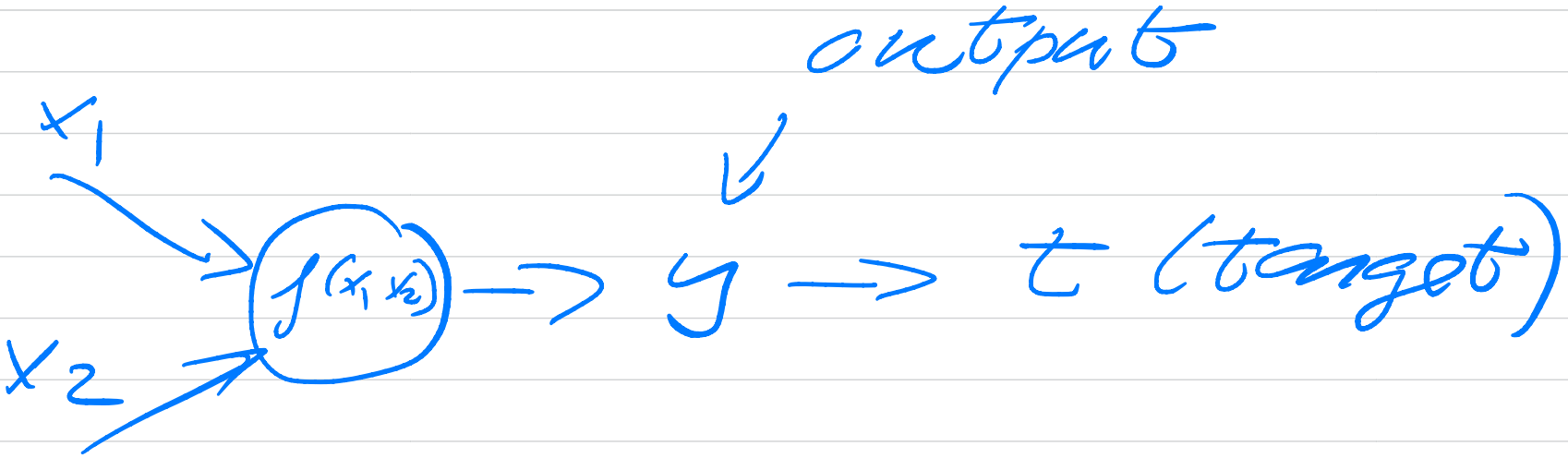
Erasmus+ course on Machine Learning, lecture November 20, 2023

Neural Networks

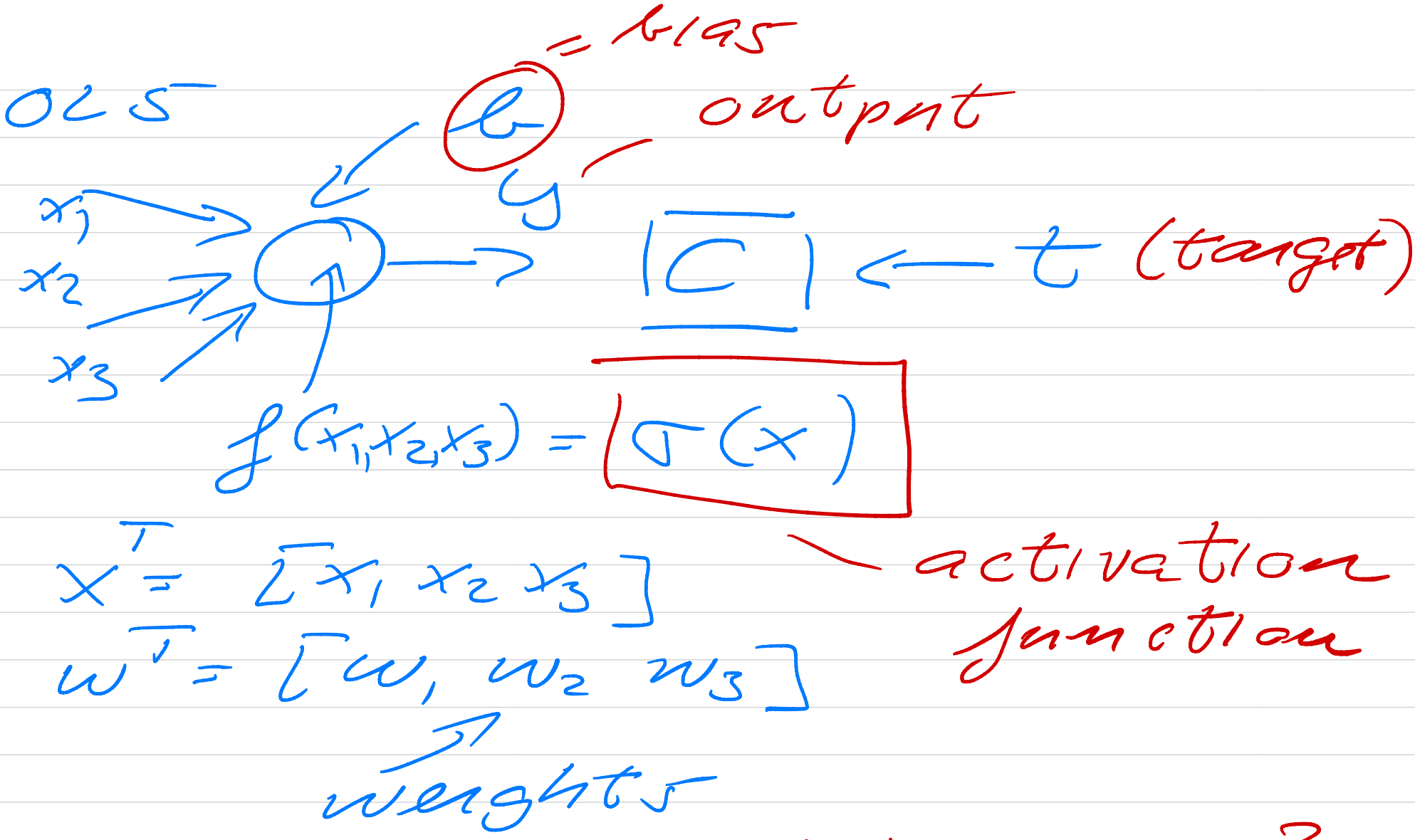




Example: $y = x_1 w_1 + x_2 w_2 + b$

$w^T = [w_1 \; w_2]$

$x^T = [x_1 \; x_2]$



$$x^T = [x_1 \; x_2 \; x_3]$$

$$w^T = [w_1 \; w_2 \; w_3]$$

weights

$$C(w, b) = \frac{1}{2} \sum_{i=0}^{n-1} (t_i - y_i)^2$$

$$\frac{\partial C}{\partial w} = 0 \quad \wedge \quad \frac{\partial C}{\partial b} = 0$$

classification problem (Binary)

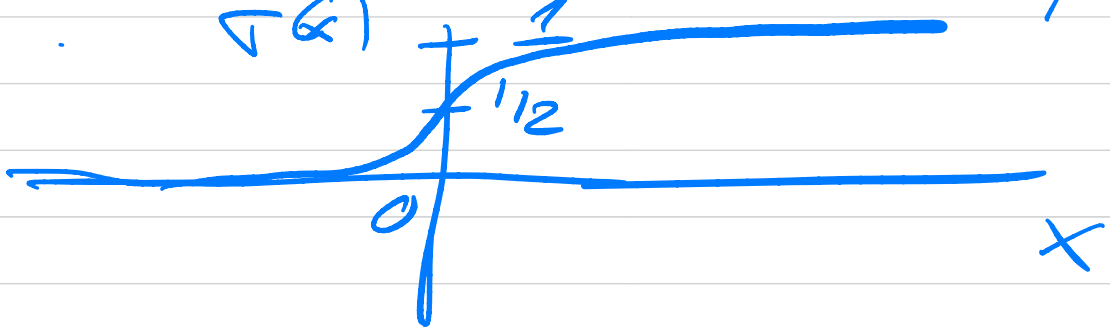
$t = \{0, 1\}$

cross-entropy

$$C(w, b) = -\sum_{i=0}^{n-1} \left(y_i' \log p_i' + (1 - y_i') \log (1 - p_i') \right)$$

$f(x) \rightarrow p(x) \quad \{ p_i' = p(x_i') \}$

$$p(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$$



$$y = x^T w + b$$

$$f(x_1, x_2) = y = x_1 w_1 + x_2 w_2 + b$$

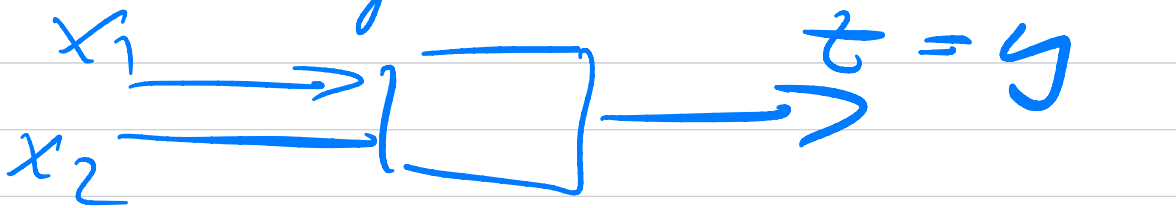
$$f(x_1, x_2) \Rightarrow \sigma(x_1, w_1, x_2, w_2, b)$$

$$= \sigma(x; \theta)$$

$$\theta = \{w, b\}$$

New example:

Gates from electronics



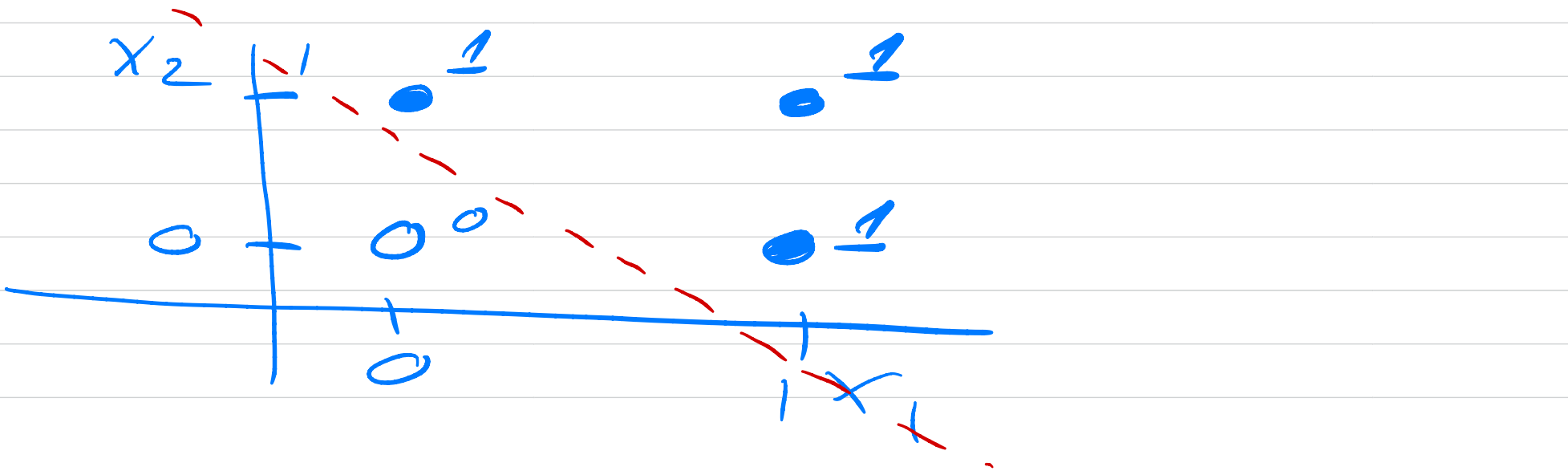
OR-gate

	x_1	x_2	t
4 values $n=4$	0	0	0
	0	1	1
	1	0	1
	1	1	1

2 features

$p = 2$

$$X^T = \{ [0,0], [0,1], [1,0], [1,1] \}$$



$$\theta = \begin{bmatrix} b \\ w_1 \\ w_2 \end{bmatrix} \qquad \{ \mathbb{R} \}$$

$$y = x^T w + b$$

$$\begin{bmatrix} 0 & 0 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} + b = 0$$

$$\begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} + b = \underline{1}, \text{ etc}$$

include intercept

$$X = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$X^T X = \begin{bmatrix} 4 & 2 & 2 \\ 2 & 2 & 1 \\ 2 & 1 & 2 \end{bmatrix}$$

$$\hat{\theta} = (X^T X)^{-1} X^T t$$

$$y = X\hat{\theta} = \begin{bmatrix} 1/4, 3/4, 3/4, 3/4 \end{bmatrix}$$

$$t = \begin{bmatrix} 0, 1, 1, 1 \end{bmatrix}$$

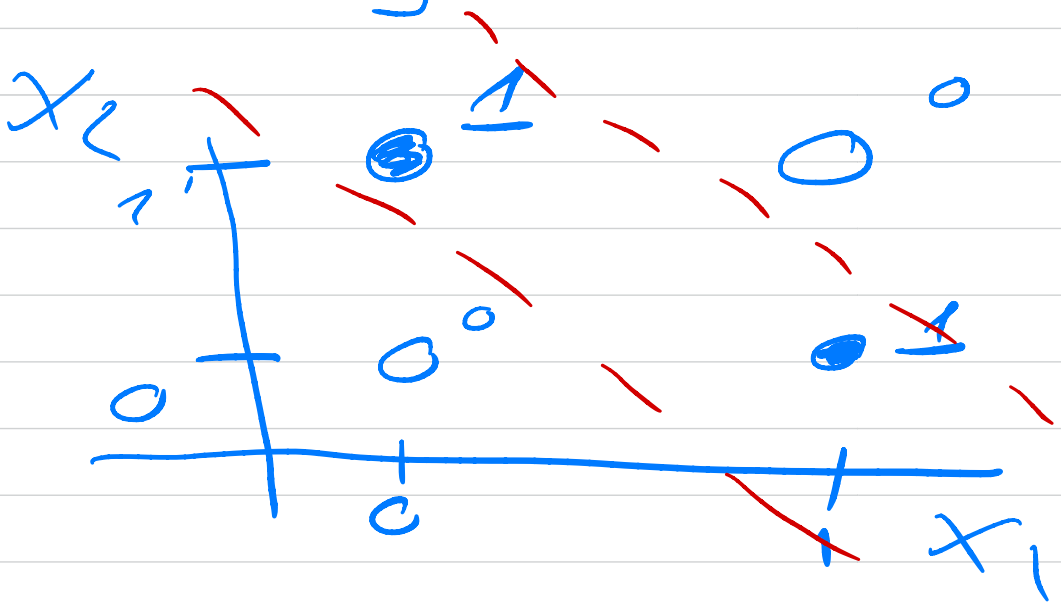
if output $y \geq 1/2$ then $y = 1$

else $y \leq 1/2$ then $y = 0$

$\bar{e}^T = [\,1/4,\ 1/2,\ 1/2\,]$

XOR

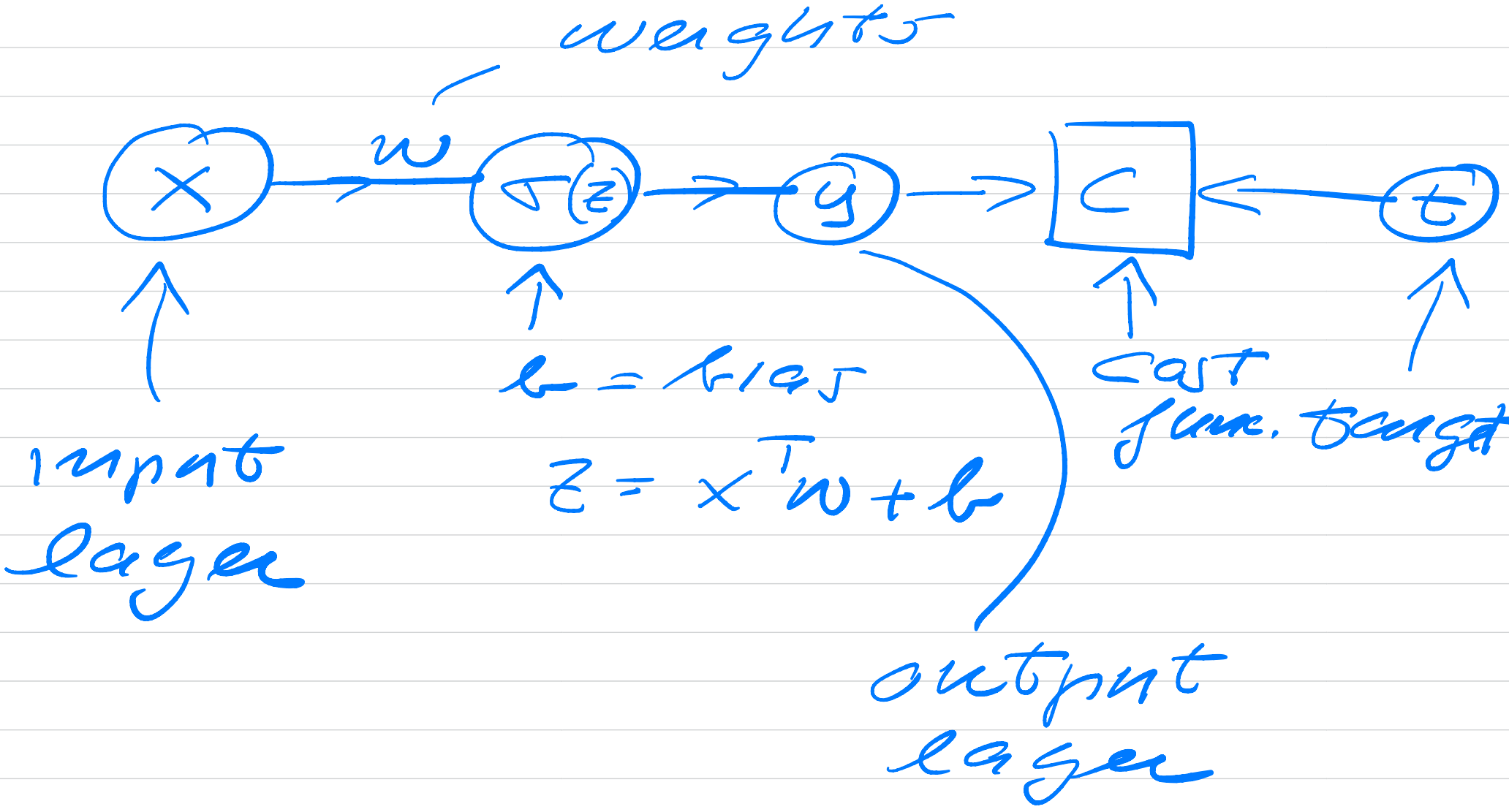
x_1	x_2	t
0	0	0
0	1	1
1	0	1
1	1	0



$y = [\,\frac{1}{2}\ \frac{1}{2}\ \frac{1}{2}\ \frac{1}{2}\,]$

$t = [\,0\ 1\ 1\ 0\,]$

with one input and one output layer, we cannot reproduce the XOR gate



The universal approximation theorem states that given a function $F \in [0,1]^d$ and error $\varepsilon > 0$, there exists a one-layer (one hidden) neural network $f(x;\theta)$ with $\theta = \{W, b\}$ $W \in \mathbb{R}^{m \times m}$ and $b \in \mathbb{R}^m$ for which

$$|f(x;\theta) - F(x)| < \varepsilon$$

for all $x \in [0,1]^d$

any continuous function $y = F(x)$ supported on unit cube can be approximated with a given error ε with one hidden layer.

Model (architecture of network)

- a given number of hidden layers
- a given number of nodes in a hidden layer
- a specific activation function $\sigma(z)$

- Sigmoid $\sigma(z) = \frac{1}{1+e^{-z}}$
- tanh $\sigma(z) = \tanh(z)$
- ELU $\sigma(z) \sim \exp(z)$
- ReLU $\sigma(z) = \max(0, z)$

⋮

Bounded, continuous / with continuous derivative.

Optimization

- gradient descent methods

- simple GD with constant learning rate η
- Stochastic GD with # epochs and mini-batches
- ADAM
- RMSprop
- Adagrad

⋮

- Cost function $C(\Theta) = C(W, B)$
- Hyperparameter $\lambda \|w\|_2^2$

$\lambda \|w\|_1$

$$\frac{\partial C}{\partial w} = 0 \quad 1 \quad \frac{\partial C}{\partial b} = 0$$

Back propagation algo:

$$\theta = \{w, b\}$$

General structure

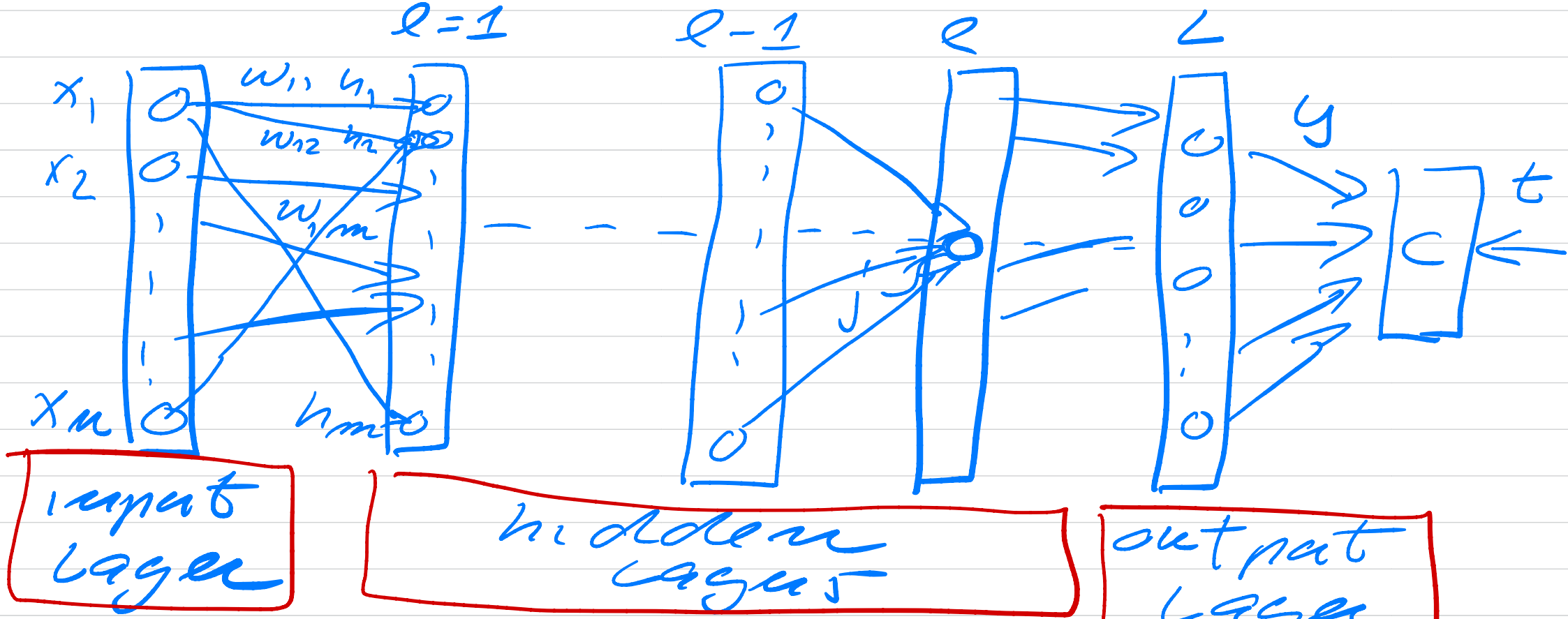
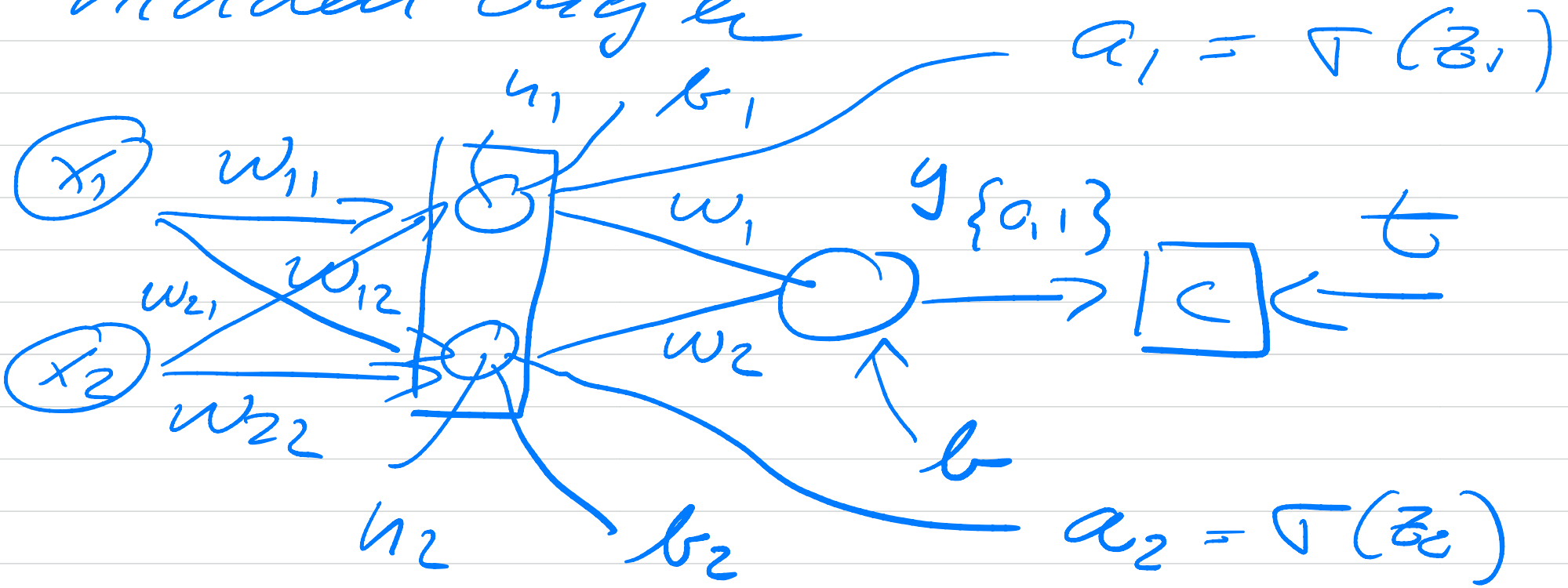


image of 8x8 pixel
64 input nodes

Think of XOR gate with one hidden layer



Design a input matrix

$$X = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix} \in \mathbb{R}^{4 \times 2}$$

$$W = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} \in \mathbb{R}^{2\times 2}$$

$$X W \in \mathbb{R}^{4\times 2}$$

$$X W + b$$

$$z_1 = X W + b_1$$

$$a_1 = \sigma(z_1)$$

$$a_2 = \sigma(z_2)$$

in more detail for layer ℓ :

$$\boxed{z_j^{\ell}} = \sum_{i=1}^{N_{\ell-1}} w_{ij}^{\ell} a_i^{\ell-1} + b_j^{\ell}$$

output from node - j - in layer - l -

$$a_j^l = \sigma(z_j^l) = \frac{1}{1+e^{-z_j^l}}$$

$$\left(\sigma = \frac{1}{1+e^{-z}}\right)$$

intermediate quantities -

$$z_j^l = \sum_i w_{ij}^l a_i^{l-1} + b_j^l$$

$$a_i^{l-1} = \sigma(z_i^{l-1})$$

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

$$\frac{\partial a_j^e}{\partial z_j^e} =$$

$$a_j^e = \sigma(z_j^e) = \frac{1}{1+e^{-z_j^e}}$$

$$= \sigma(z_j^e)(1 - \sigma(z_j^e))$$

$$= a_j^e(1 - a_j^e)$$

$$C = C(x; \epsilon)$$

$$= C(x; \epsilon^1, \epsilon^2, \ldots \epsilon^L)$$

$$\frac{\partial C(\epsilon)}{\partial w_{jk}^{L}} = ? \qquad \frac{\partial C}{\partial b_j^{L}} = ?$$

assume that $C = \frac{1}{2}\sum_j (t_j - y_j)^2$

note implicit dependence on ϵ in $y_j = a_j^L$

$$\frac{\partial C}{\partial a_j^{L}} \frac{\partial a_j^{L}}{\partial z_j^{L}} \frac{\partial z_j^{L}}{\partial w_{jk}^{L}} = \frac{\partial C}{\partial w_{jk}^{L}}$$

$\frac{\partial C}{\partial a_j^{L}} = (a_j^L - t_j)$

$\frac{\partial a_j^{L}}{\partial z_j^{L}} = a_j^L(1 - a_j^L)$

$\frac{\partial z_j^{L}}{\partial w_{jk}^{L}} = a_k^{L-1}$

$$\delta_j^L = \underbrace{a_j^L(1-a_j^L)}_{\sigma'(z_j^L)} \underbrace{(a_j^L - t_i')}_{\frac{\partial C}{\partial a^L}}$$

$$\frac{\partial C}{\partial w_{jk}^L} = \delta_j^L \cdot a_k^{L-1}$$

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

collect all results a to output layer $l = L$

$$\frac{\partial C}{\partial w_{jk}^{l}} = \delta_j^{l} a_k^{l-1}$$

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

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

$l \to e$

$$\delta_j^{e} = ?$$

$$\delta_j^l = \frac{\partial C}{\partial z_j^l} = \sum_k \frac{\partial C}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial z_j^l}$$

$$= \sum_k \delta_k^{l+1} \frac{\partial z_k^{l+1}}{\partial z_j^l}$$

$$z_j^{l+1} = \sum_i w_{ij}^{l+1} a_i^l + b_j^{l+1}$$

$$\frac{\partial z_k^{l+1}}{\partial z_j^l} = \sigma'(z_j^l)$$

$$\rightarrow w_{kj}^{l+1} \sigma'(z_j^l)$$

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

Set up gradients:

$$w_{jk}^{\ell} \leftarrow w_{jk}^{\ell} - \eta\, \delta_j^{\ell}\, a_k^{\ell-1}$$

$$b_j^{\ell} \leftarrow b_j^{\ell} - \eta \frac{\partial C}{\partial b_j^{\ell}}$$

$$= b_j^{\ell} - \eta\, \delta_j^{\ell}$$

$\ell = 1, 2, 3, \ldots L-1, L$

Next week: set up algorithm + code for Neural networks

Link Back propagation with automatic differentiation.