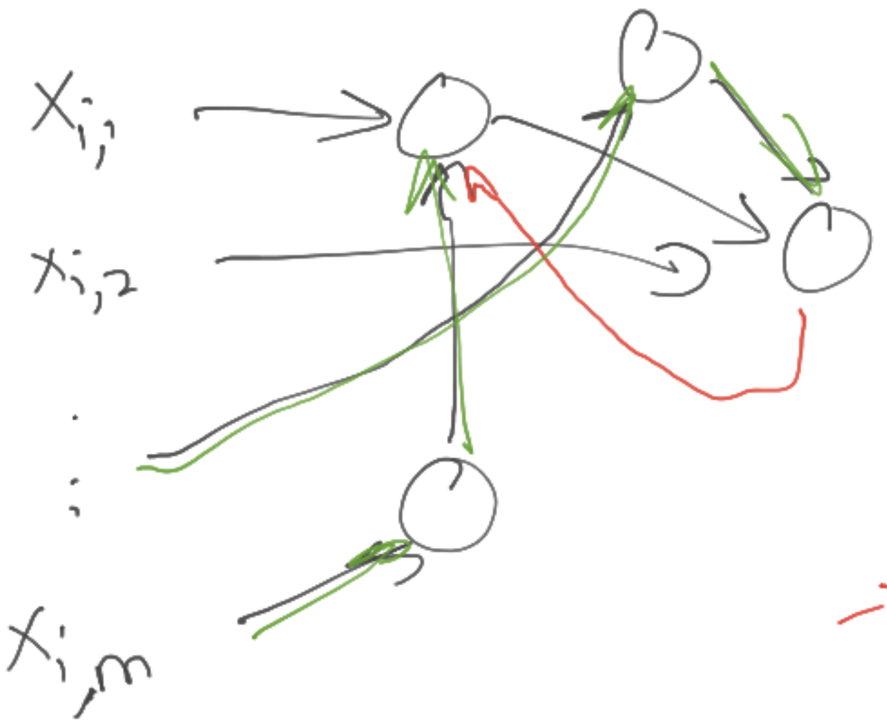
Artificial Neural Networks (ANN)

Lecture 8
-ANNs!



$\rightarrow f_{w_k}(x_i)$

"Network architecture" is how perceptrons are wired together. (and how many).

→ recurrent network: when the graph is cyclic.

- We consider non-recurrent networks.

○ → perceptron
→ unit
→ "node"
→ vertex.

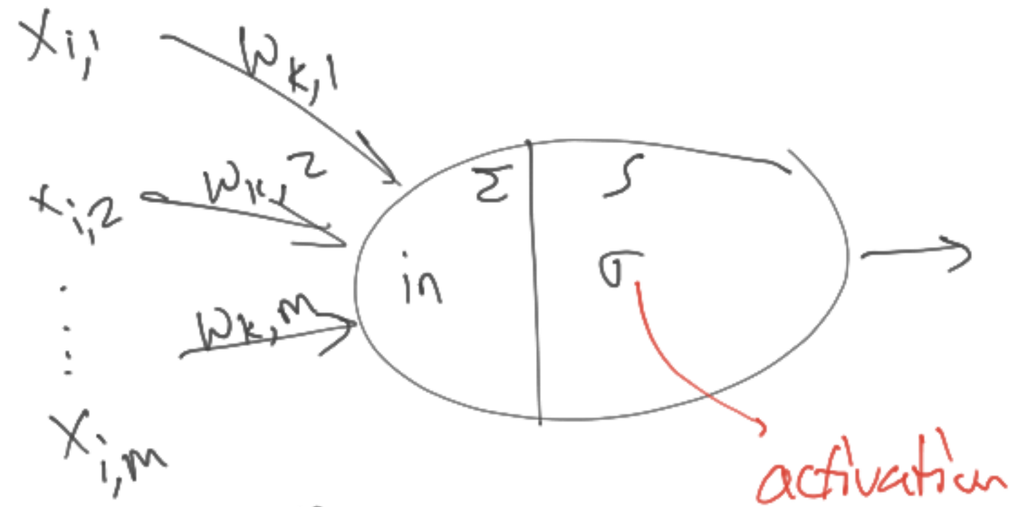
Perceptron

$$(x_i, y_i)_{i=1}^{n}$$

$$\ell(w_k) = \sum_{i=1}^{n} (y_i - f_{w_k}(x_i))^2$$

$$\forall j : w_{k+1,j} := w_{k,j} - \alpha \frac{\partial \ell(w_k)}{\partial w_{k,j}}$$

$$\frac{\partial \ell(w_k)}{\partial w_{k,j}} = 2(y_i - f_{w_k}(x_i)) \boxed{\frac{\partial f_w(x_i)}{\partial w_{k,j}}}$$

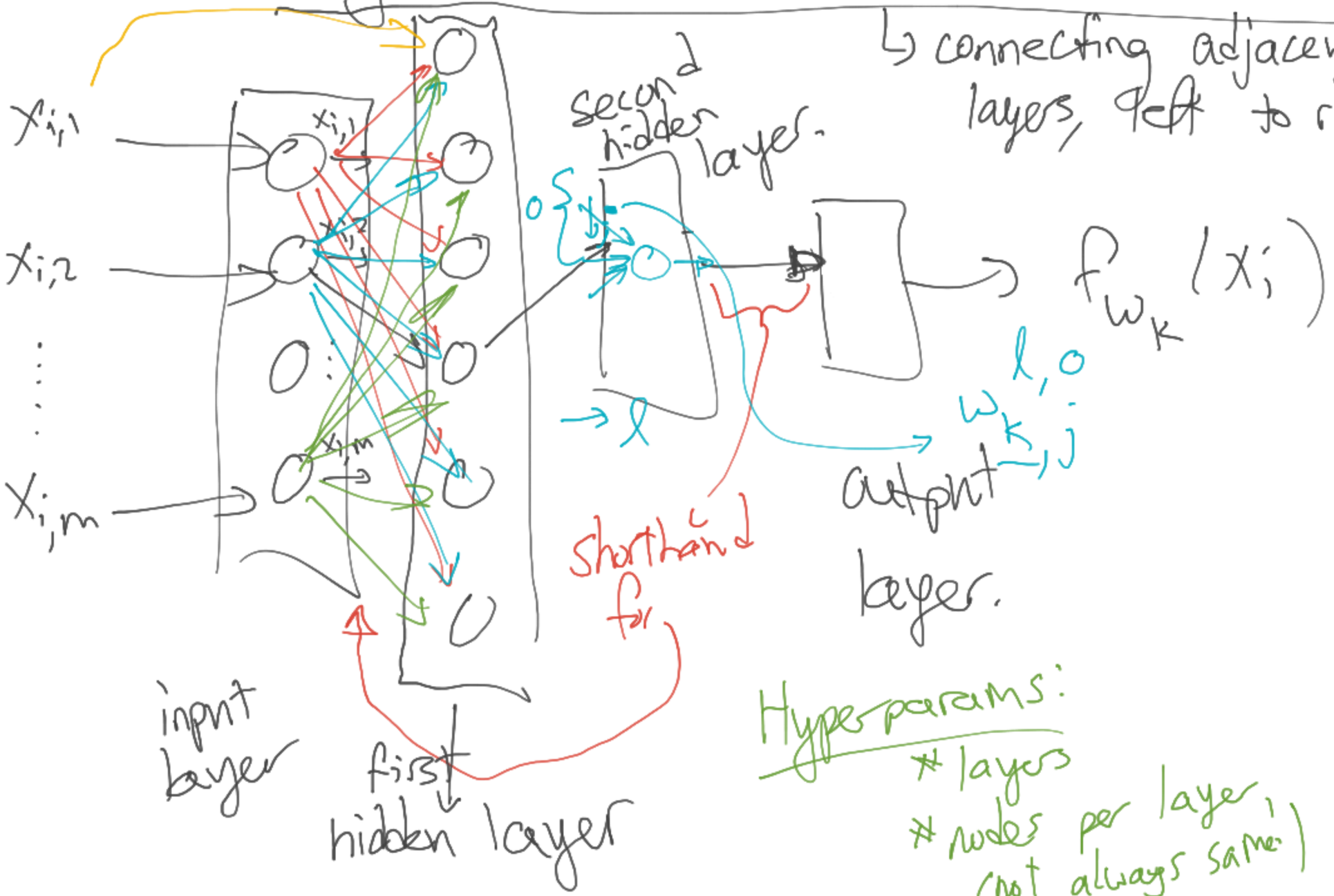


$$in = \sum_{j=1}^{m} x_{i,j}\, w_{k,j}$$

$$f_{w_k}(x_i) = \sigma(in)$$

Fully Connected Feedforward Network.

↳ connecting adjacent layers, left to right.



Hyperparams:

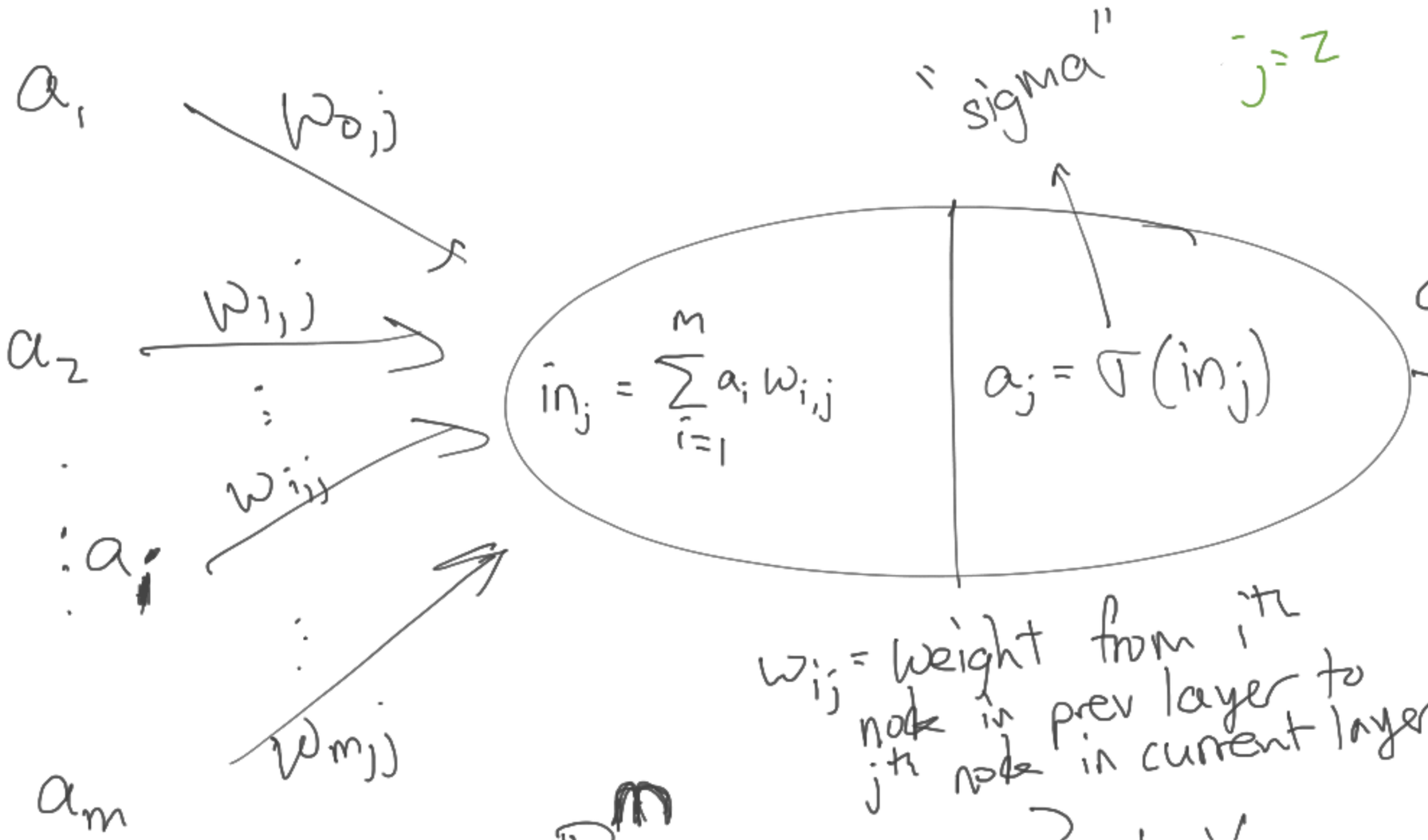
* layers
* nodes per layer (not always same)
- activation function.

New Notation

Trick: Assume "current layer" → j

"previous layer" → i

"next layer" → k



i j k

prev → current → next.

$w_{i,j}$ = weight from i^{th} node in prev layer to j^{th} node in current layer.

a_i : output of i^{th} node in prev layer.

a_j : output of j^{th} node in current layer.

$x \in \mathbb{R}^m$ ↳ input to network. } not X_i any more.

Forward Pass

// load input layer

for each input x_j in X → input to network.

$a_j \leftarrow x_j$ // a_i from input layer.

// Run hidden layers.

for each hidden layer

for each node j in current hidden layer do:

$$in_j = \sum_i a_i w_{ij}$$

$$a_j = \sigma(in_j)$$

// Return: output of output layer.

weights 4-D.

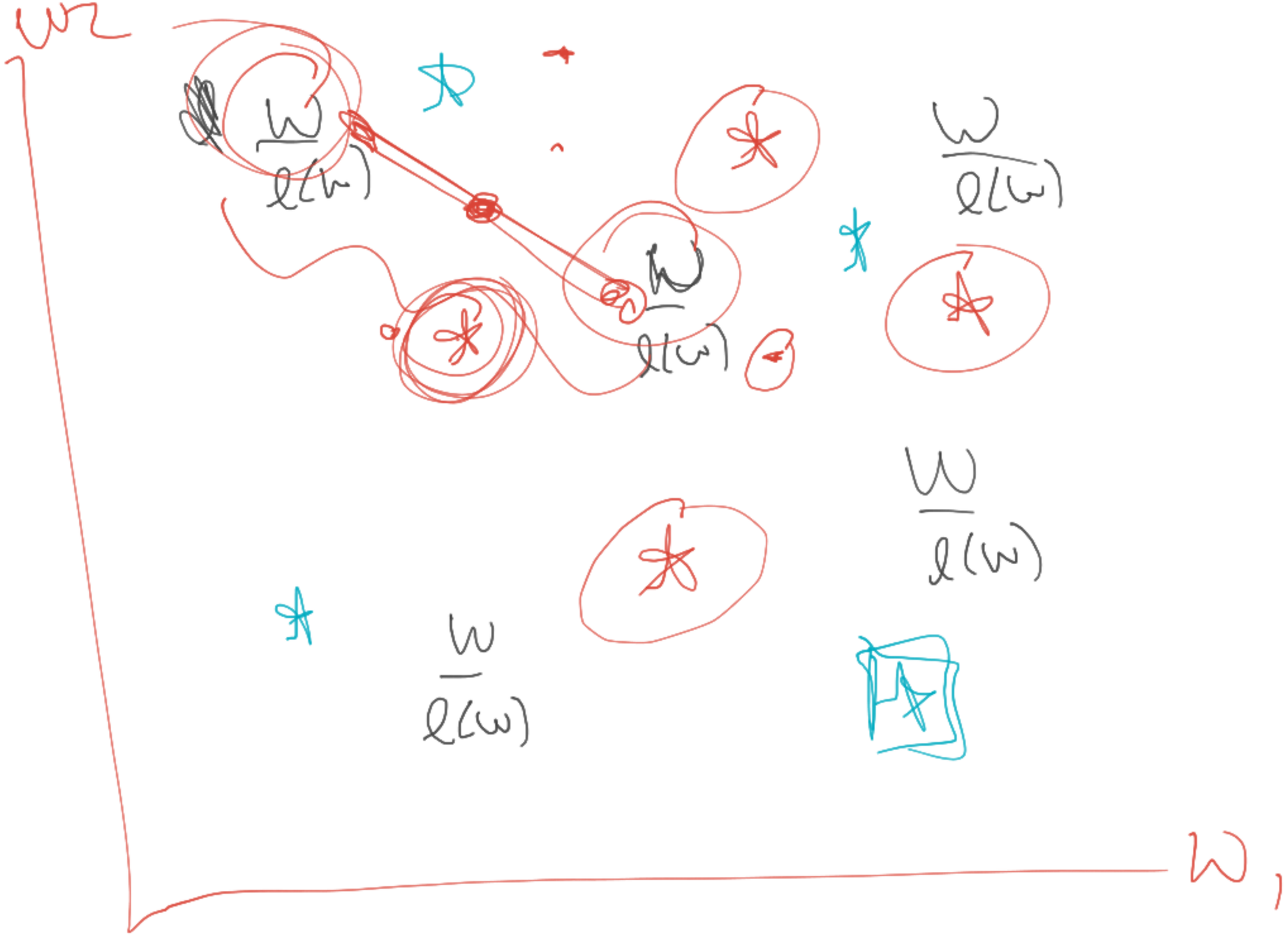
$w[l][j][i]$

- l → layer
- j → node
- i → i-th input.

$f(x)$

$x_{k+1} = x_k - \alpha \nabla f(x_k)$

$x \leftarrow x - \alpha \nabla f(x)$


w_2
$\frac{w}{\ell(w)}$
$\frac{w}{\ell(w)}$
$\frac{w}{\ell(w)}$
$\frac{w}{\ell(w)}$
$\frac{w}{\ell(w)}$
w_1