Statistical Methods in AI (CSE 471)
Lecture14: Neural Networks

Vineet Gandhi
Centre for Visual Information Technology (CVIT)

Artificial Neural Networks

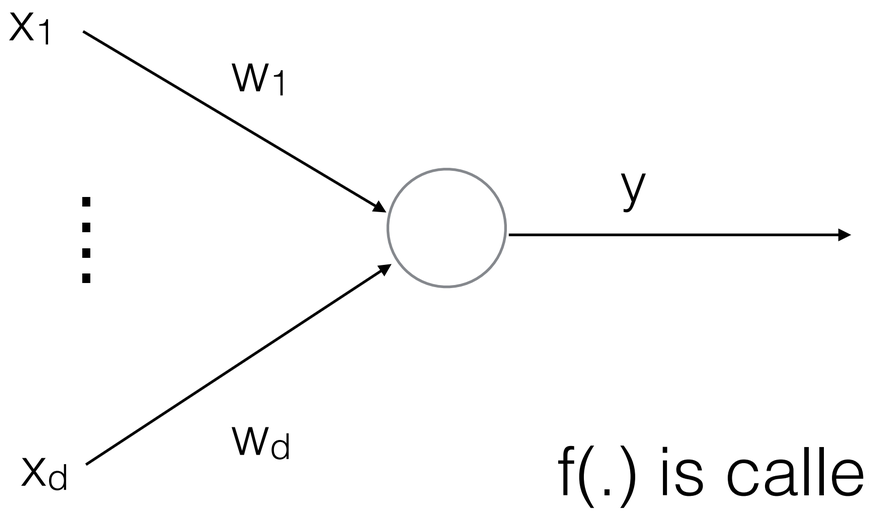
- A parallel distributed information processor made up of simple processing units that has a propensity for acquiring problem solving knowledge through experience
  - Large number of inter connected units
  - Each unit implements simple function, non linear
  - The knowledge resides in the interconnection strengths

Artificial Neural Networks

- Historical development of ANN was motivated by attempts to understand how human brain works
  - Neuron- basic computing unit
  - Human Brain: 10^{11} neurons, 10000 synapses per neutron
  - Total synapses: 10^{15}

Single Neuron model

$$\text{net input: } \eta = \sum_j w_j x_j$$

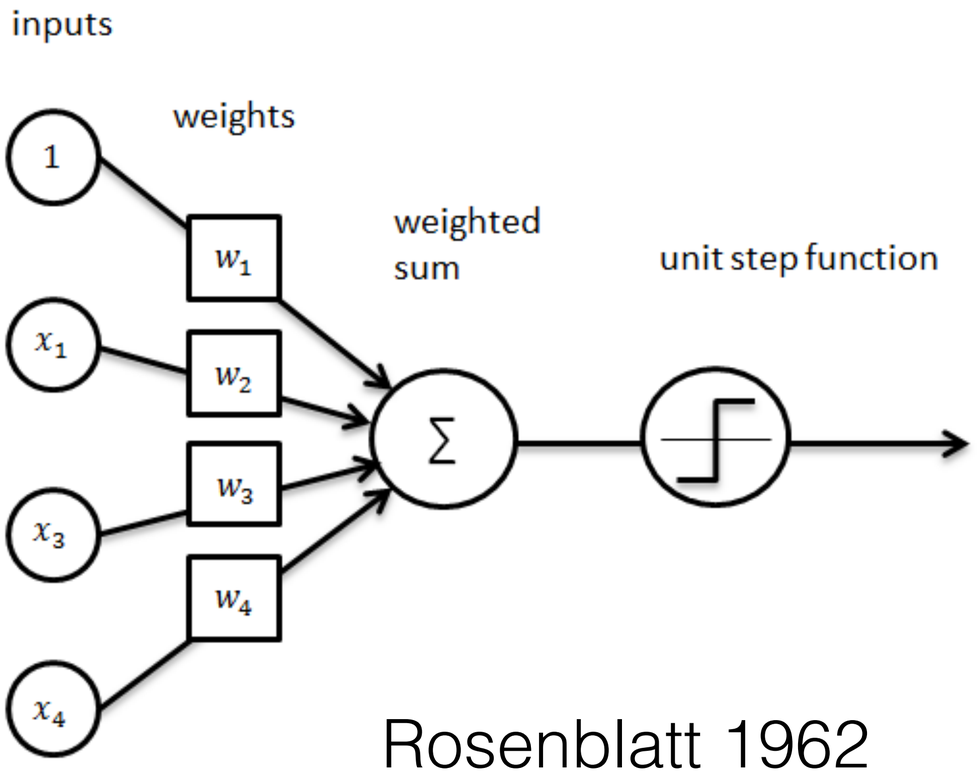


$$\text{output: } f(\eta)$$

f(.) is called the activation function

Perceptron

- Perceptron algorithm is one of the earliest algorithms for learning linear discriminant functions
- Finds a separating hyperplane if it exists



Rosenblatt 1962

Perceptron Learning algorithm

$$\Delta W(k) = W(k+1) - W(k)$$

$$\Delta W(k) = 0 \begin{cases} \text{if } W(k)^T X(k) > 0 \ \& \ y(k) = 1, \\ \text{or } W(k)^T X(k) < 0 \ \& \ y(k) = 0. \end{cases}$$

$$\Delta W(k) = \begin{cases} X(k) & \text{if } W(k)^T X(k) \leq 0 \ \& \ y(k) = 1, \\ -X(k) & \text{if } W(k)^T X(k) \geq 0 \ \& \ y(k) = 0. \end{cases}$$

- Error correcting algorithm: at every incorrect classification we perform ‘local’ correction

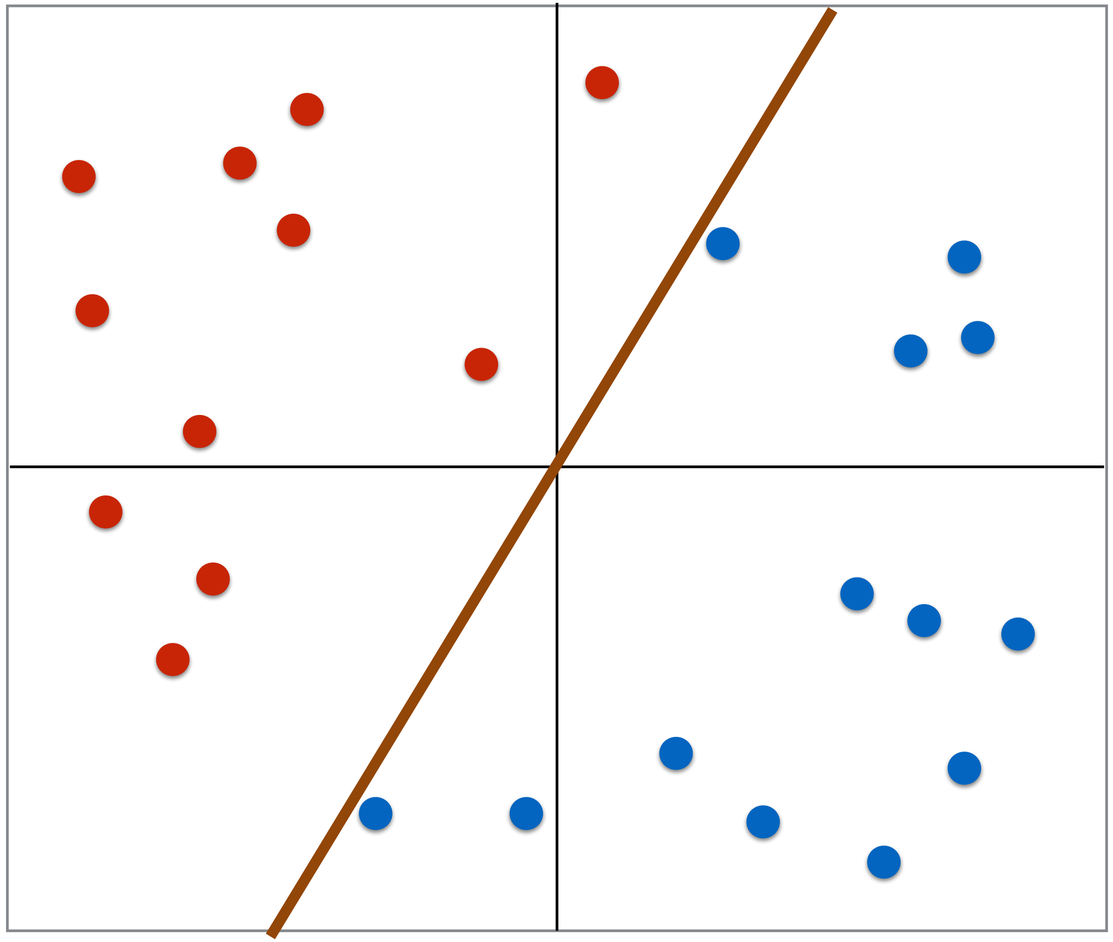
Intuition behind corrections

$$W(k)^T X(k) \leq 0 \ \& \ y(k) = 1$$

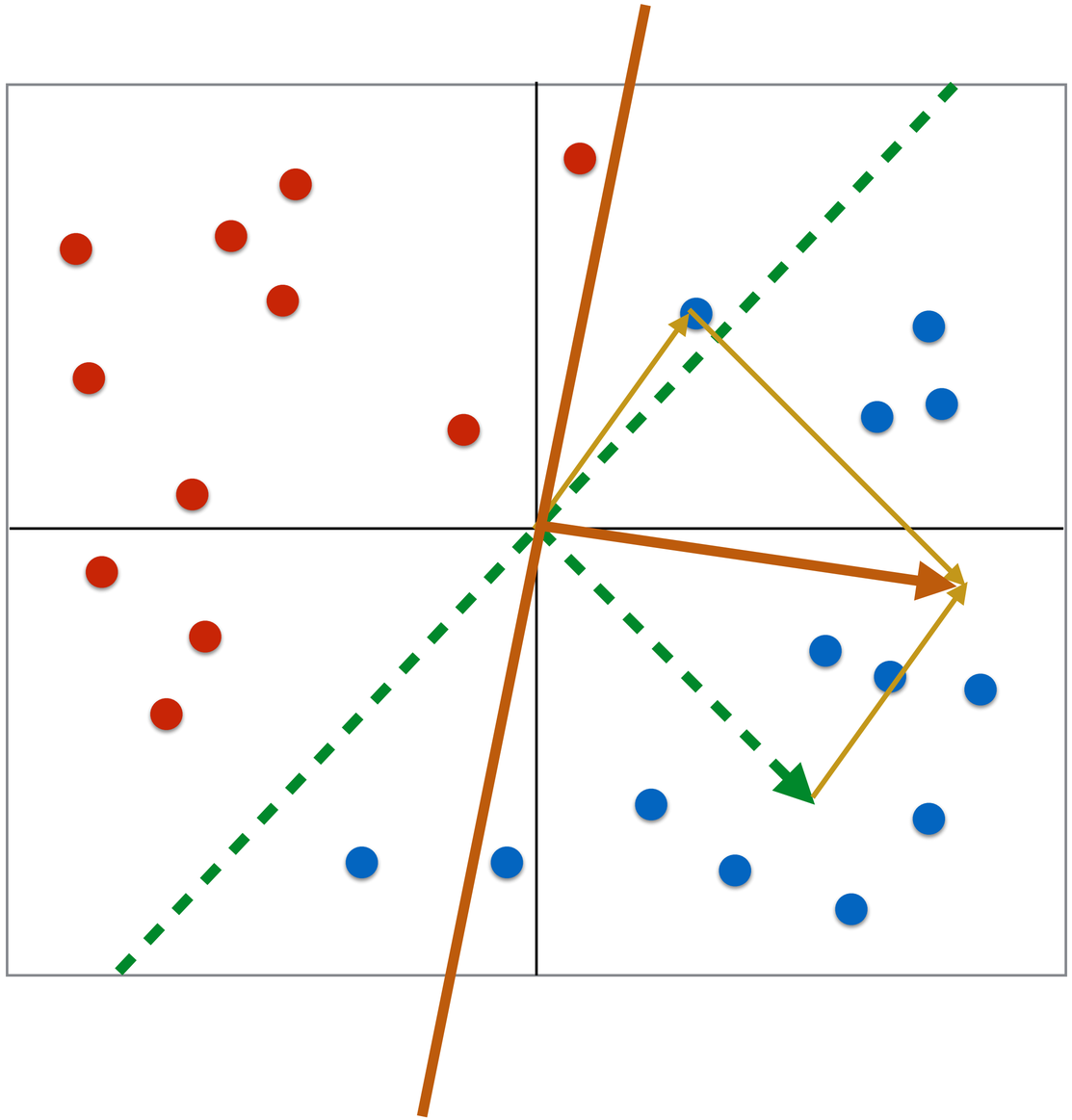
$$W(k+1)^T X(k) = W(k)^T X(k) + X(k)^T X(k) \geq W(k)^T X(k)$$

Similarly for other type of error….

Geometric view



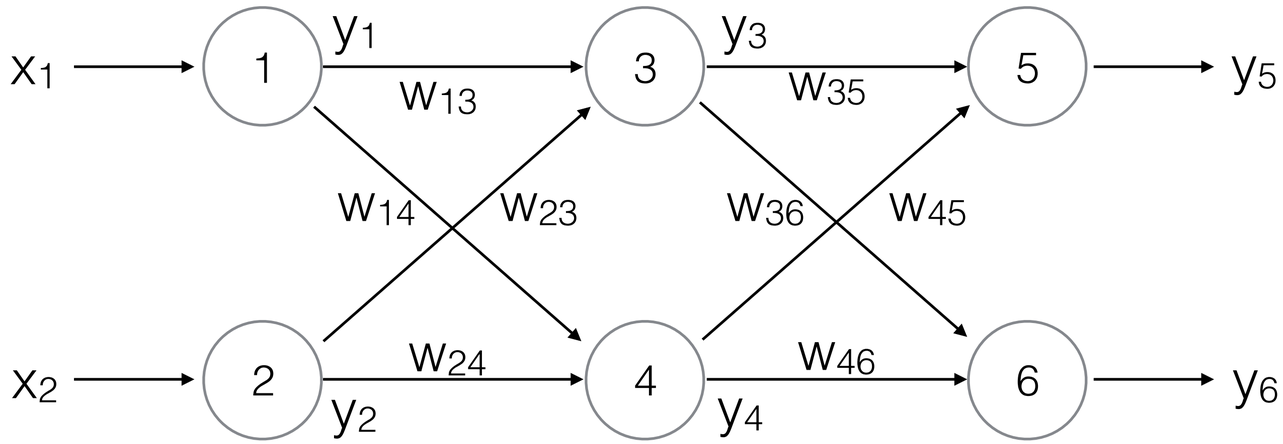
Geometric view



More on perceptrons

- The intuition behind the algorithm is clear, however, it is unclear how such a simple algorithm will work
- There is no guarantee that $W(k+1)^T X(k)$ has correct sign
- When we correct W(k) to take care of some X(k), we may misclassify another input feature vector that W(k) classified correctly
- Hence, it is remarkable that the algorithm works (we will show that soon)

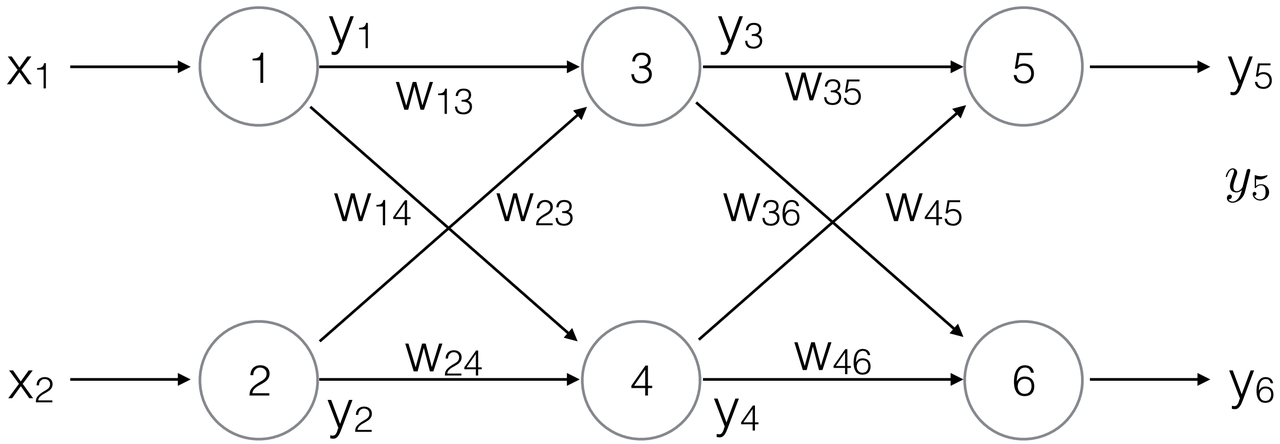
Network



$$y_5 = f_5(w_{35}y_3 + w_{45}y_4)$$

$$= f_5(w_{35}f_3(w_{13}y_1 + w_{23}y_2) + w_{45}f_4(w_{14}y_1 + w_{24}y_2))$$

Network



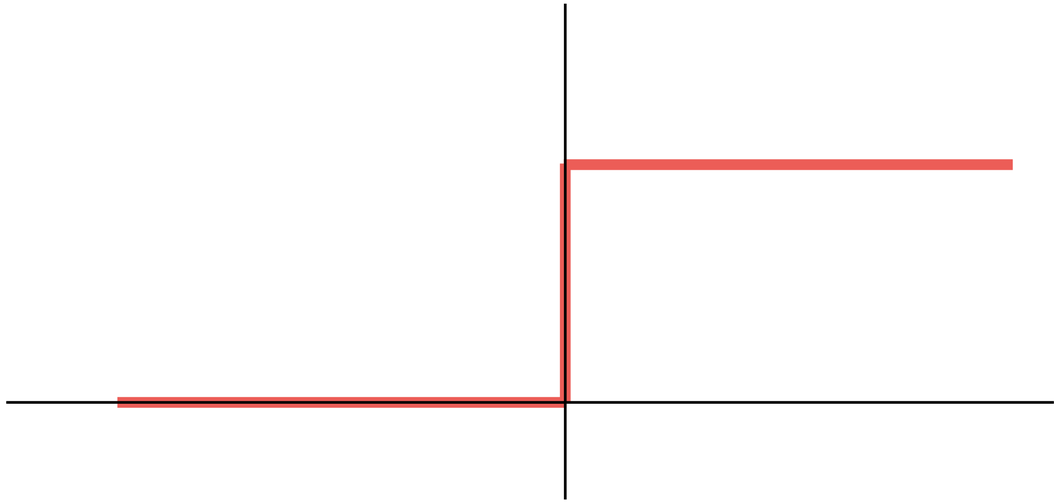
$$y_5 = f_5(w_{35}y_3 + w_{45}y_4)$$

$$= f_5(w_{35}f_3(w_{13}y_1 + w_{23}y_2) + w_{45}f_4(w_{14}y_1 + w_{24}y_2))$$

To form meaningful networks, non linearity of activation functions is important

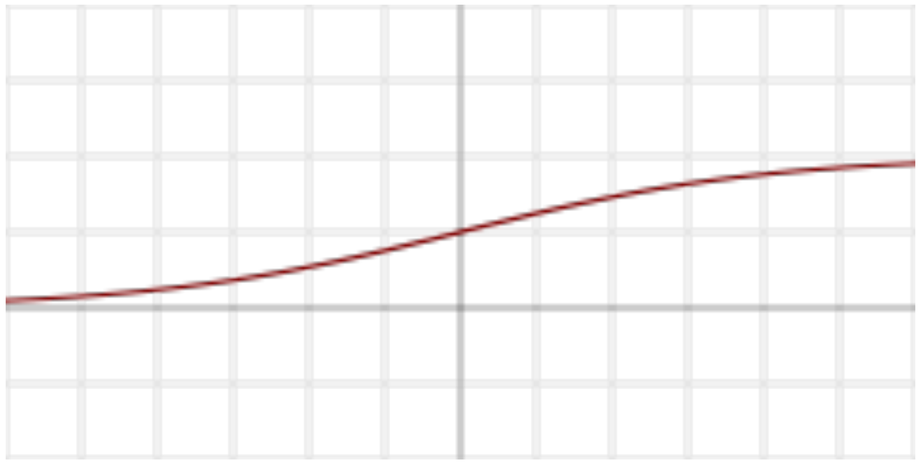
Without non linearity, it can do no more than a single neuron model

Activation functions



$$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$$

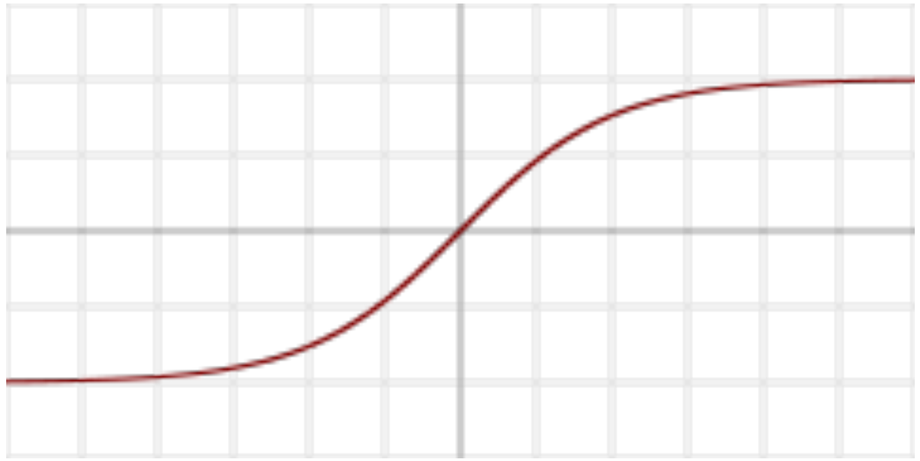
Activation functions



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

$$f'(x) = f(x)(1 - f(x))$$

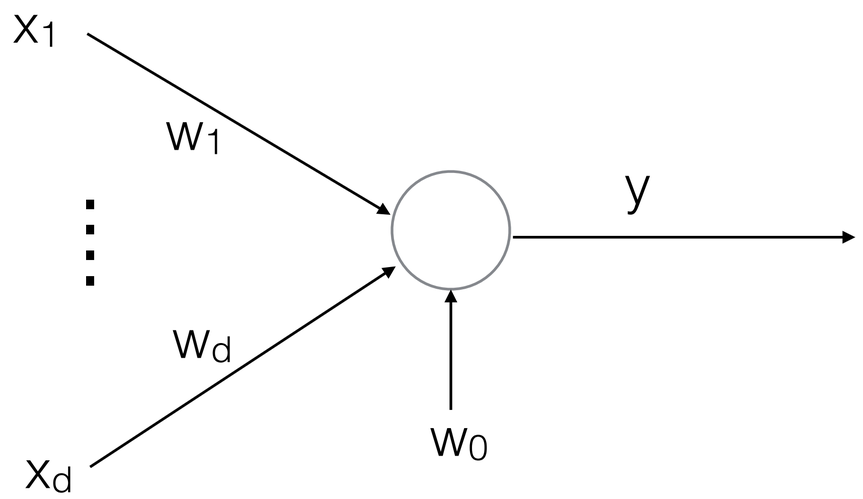
Activation functions



$$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$$

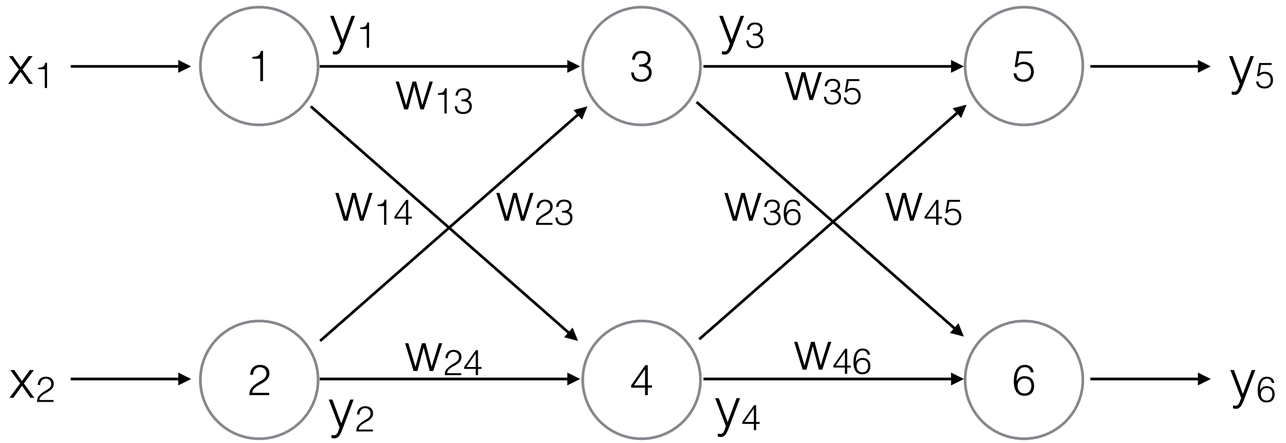
$$f'(x) = f(x)(1 - f(x))$$

Bias term

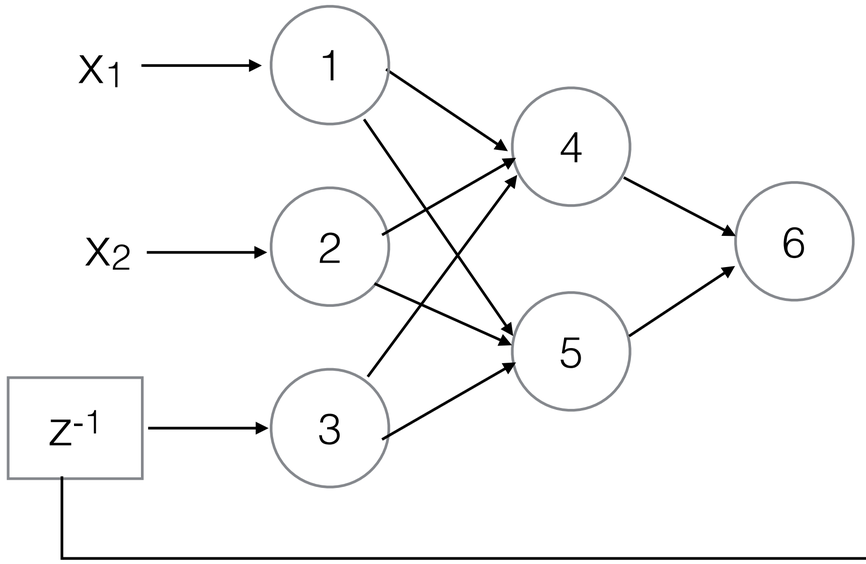


Each unit can also have a bias input

Feedforward vs Recurrent

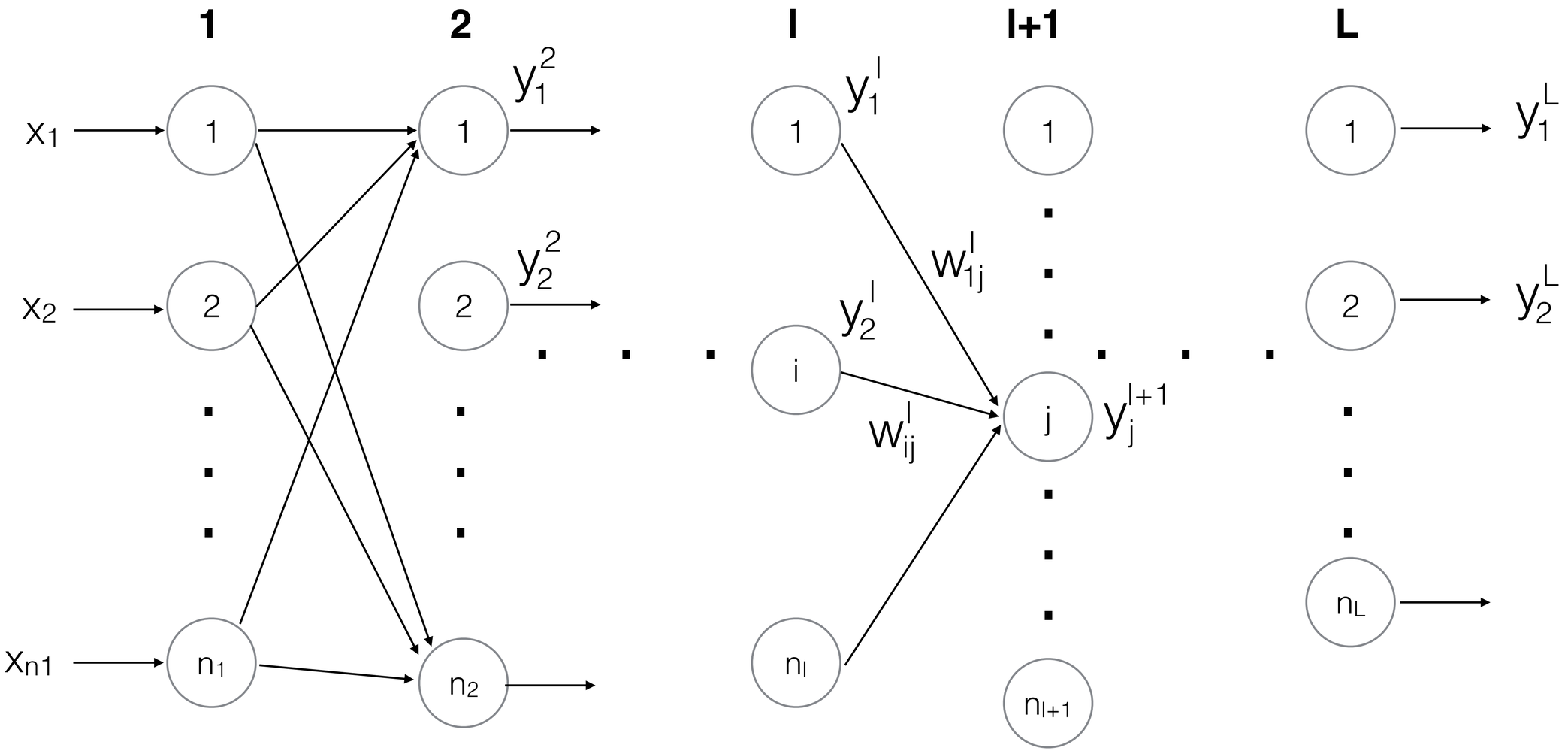


No feedback



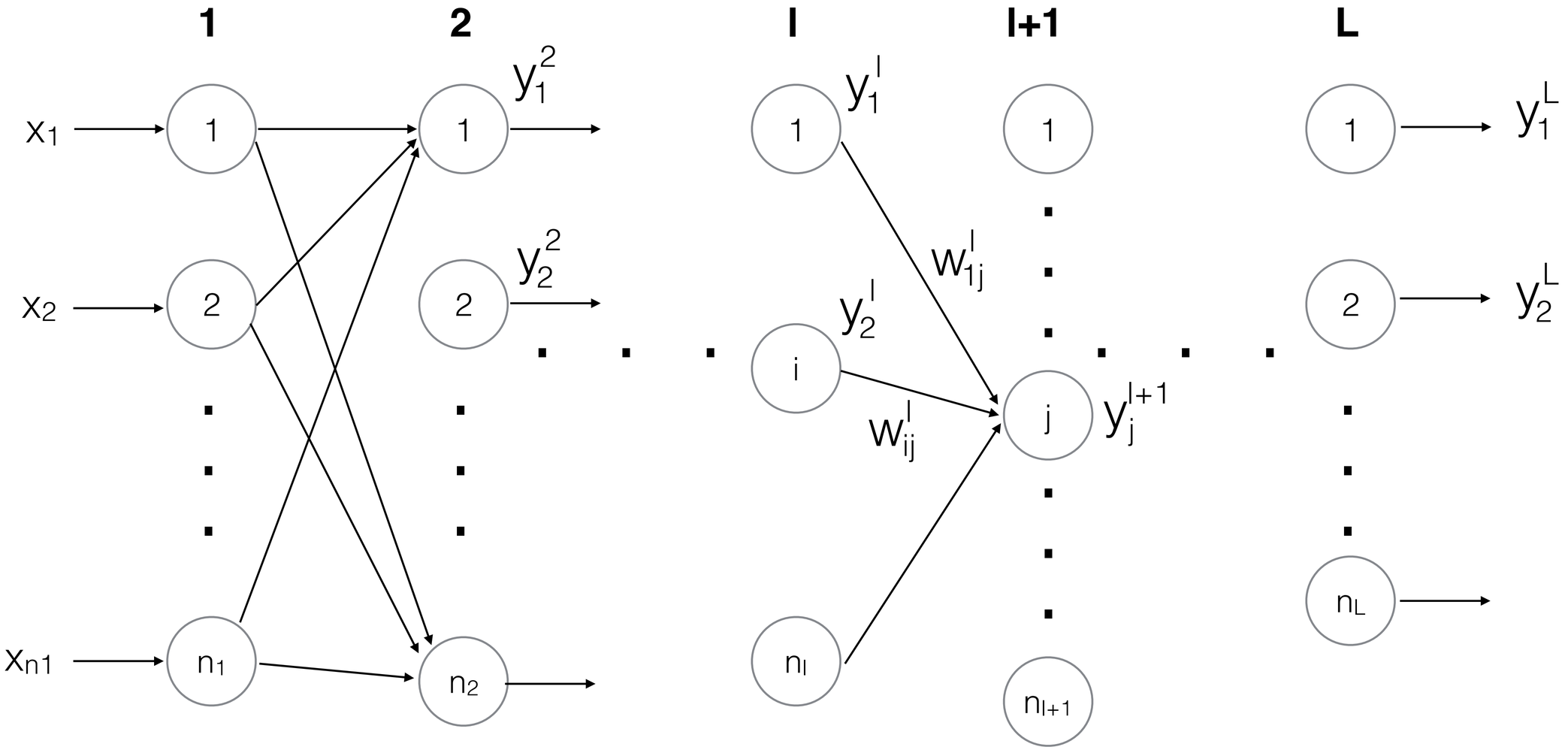
Previous output as feedback

Notations: multilayer feedforward network



Mapping from R^{n1} to R^{nL}

Hidden layers?



Cost function

$$J_i(W) = \frac{1}{2}\sum_{j=1}^{n_L}(y_j(W, X^i) - d_j^i)^2$$

$$J(W) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{n_L}(y_j(W, X^i) - d_j^i)^2$$

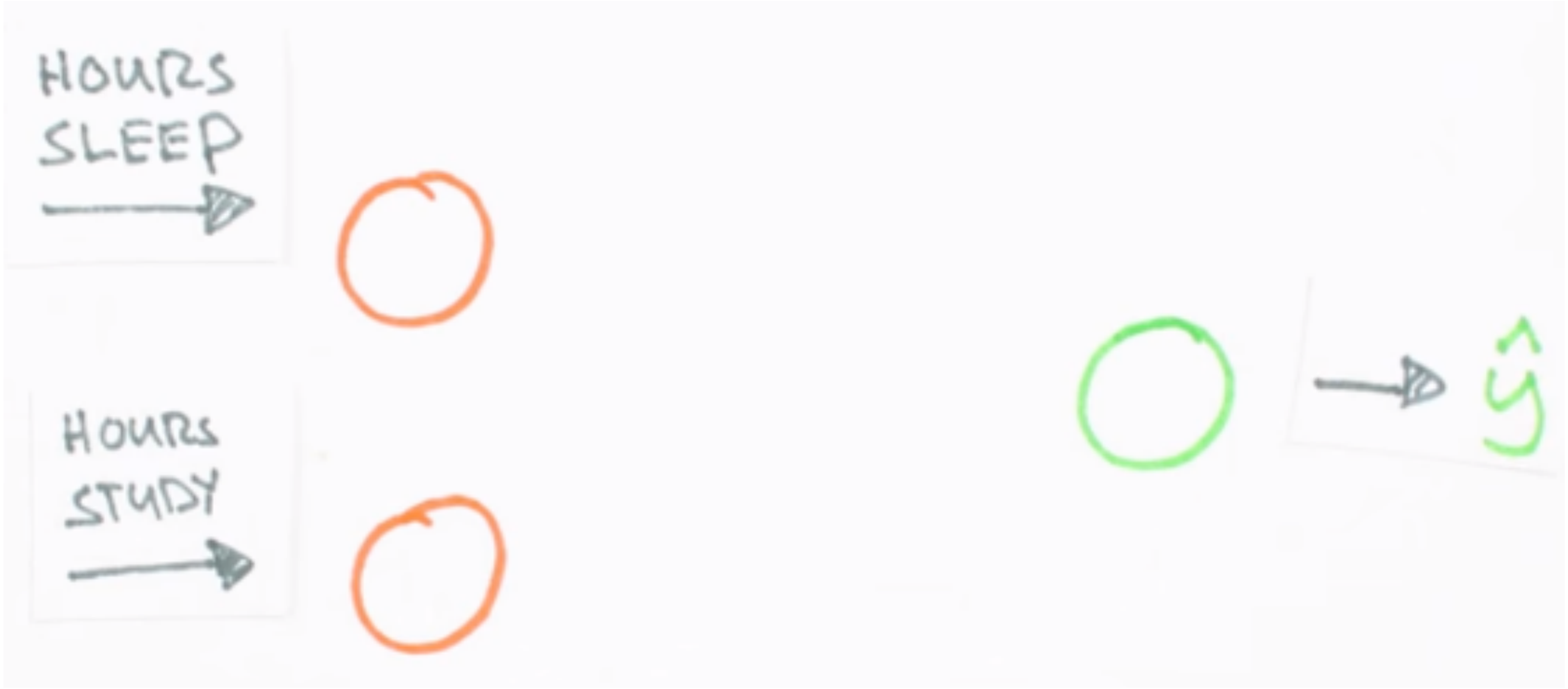
$$J_i(W) = \frac{1}{2}\sum_{j=1}^{n_L}(y_j - d_j)^2$$

Gradient Descent

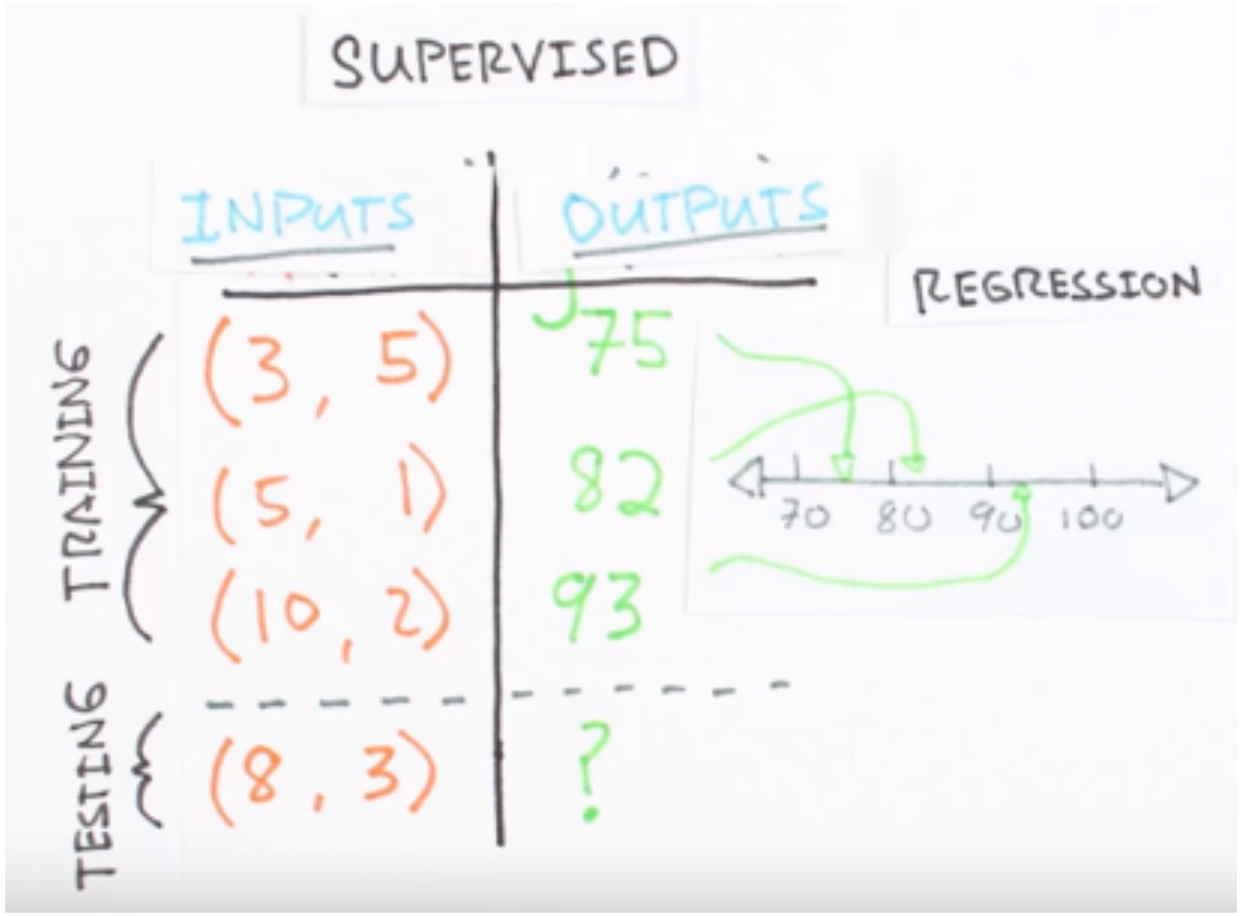
$$J_i(W) = \frac{1}{2}\sum_{j=1}^{n_L}(y_j(W, X^i) - d_j^i)^2$$

$$w_{ij}(t+1) = w_{ij}(t) - \lambda\frac{\partial J}{\partial w_{ij}}$$

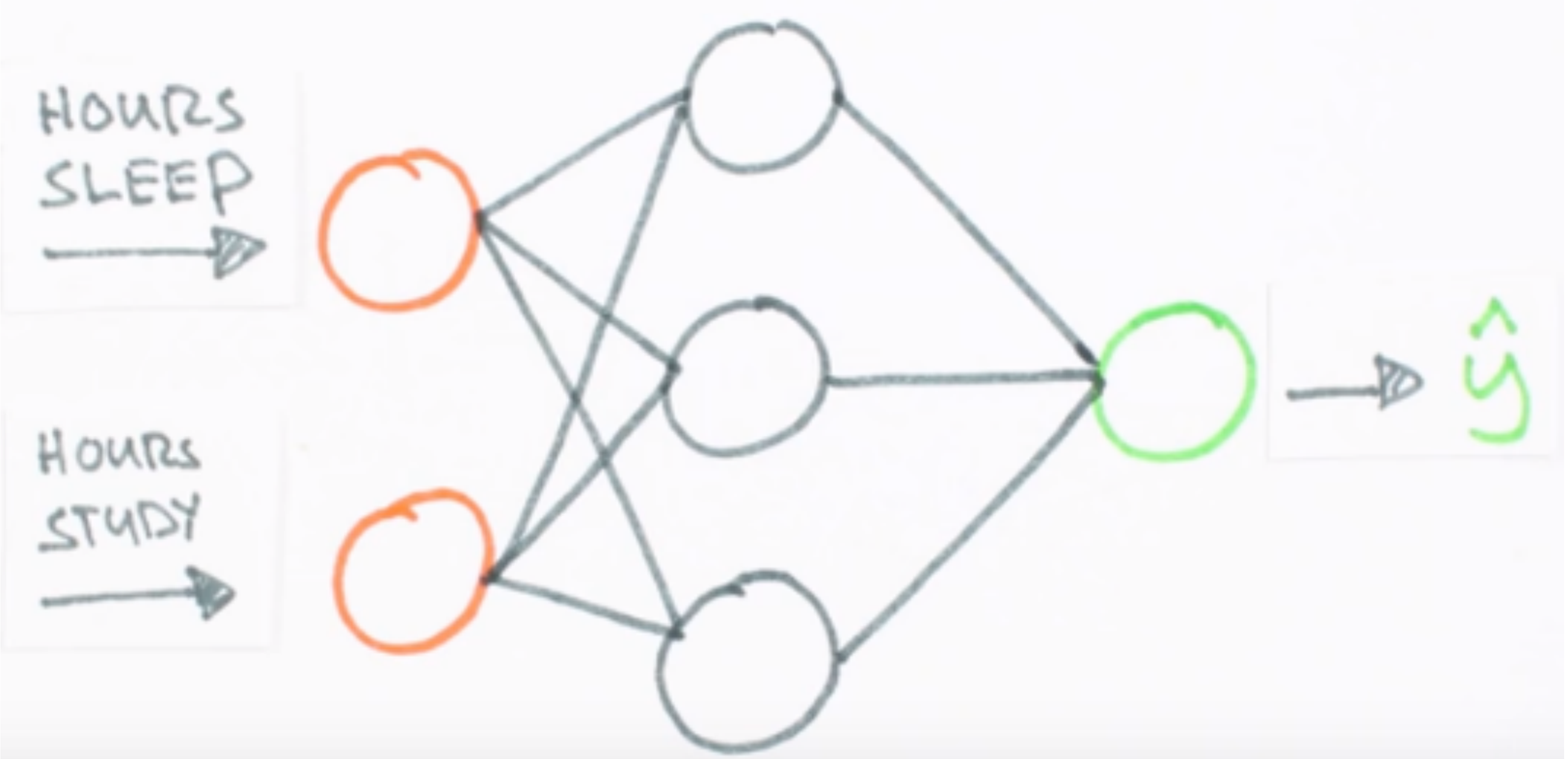
Example



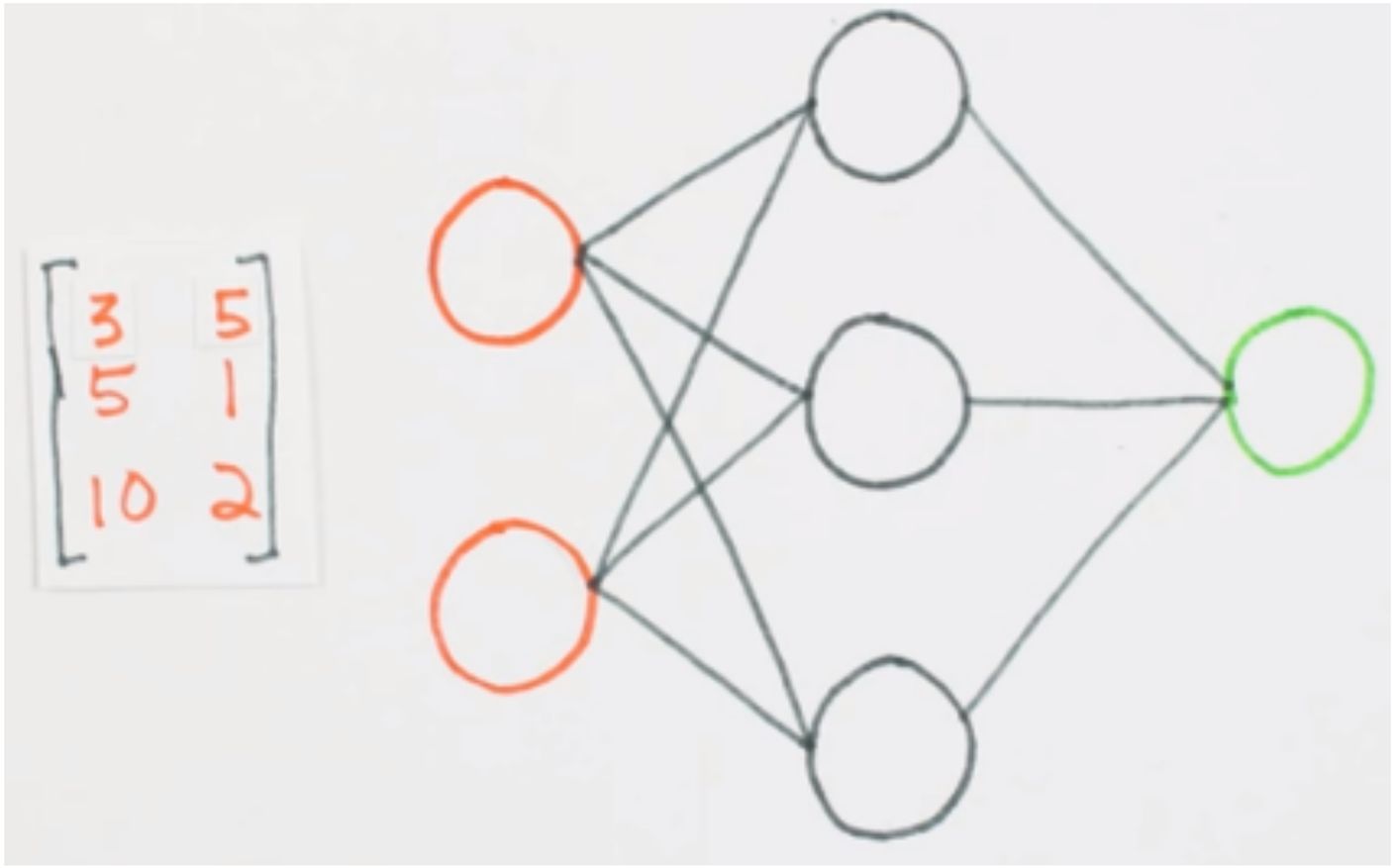
Example


SUPERVISED
INPUTS
OUTPUTS
REGRESSION
TRAINING
TESTING
(3, 5)
75
(5, 1)
82
(10, 2)
93
(8, 3)
?
70
80
90
100

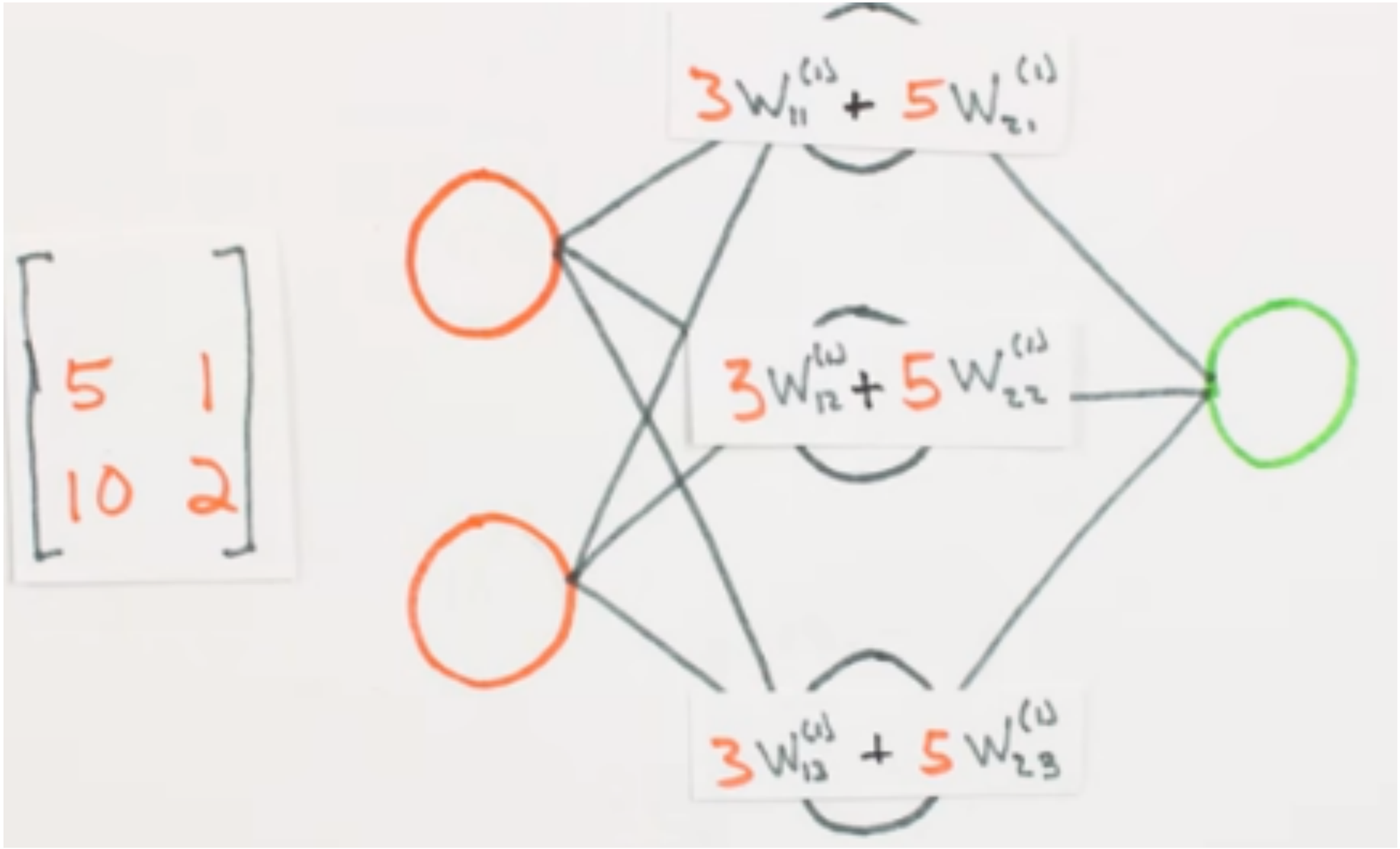
Example



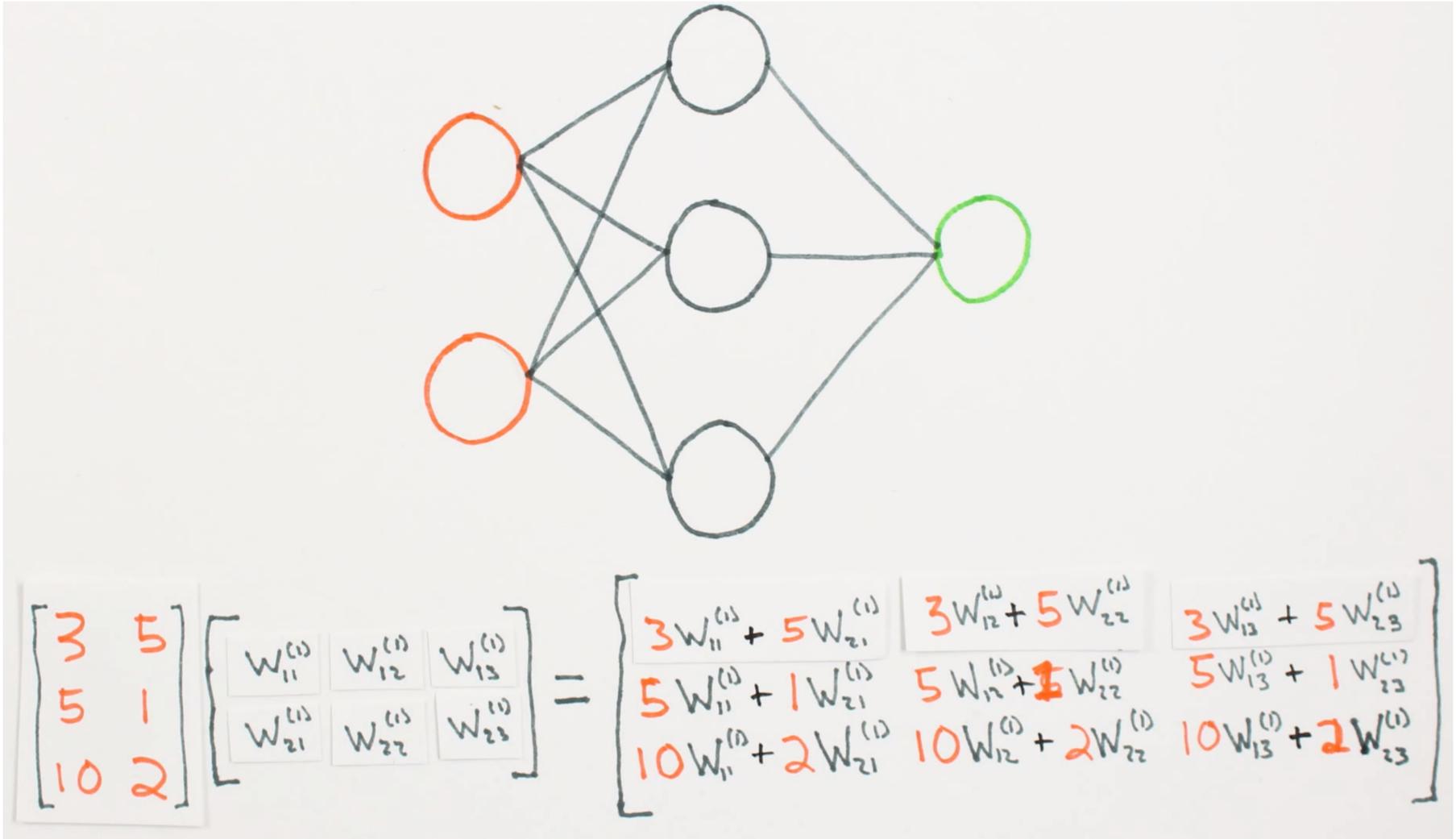
Example



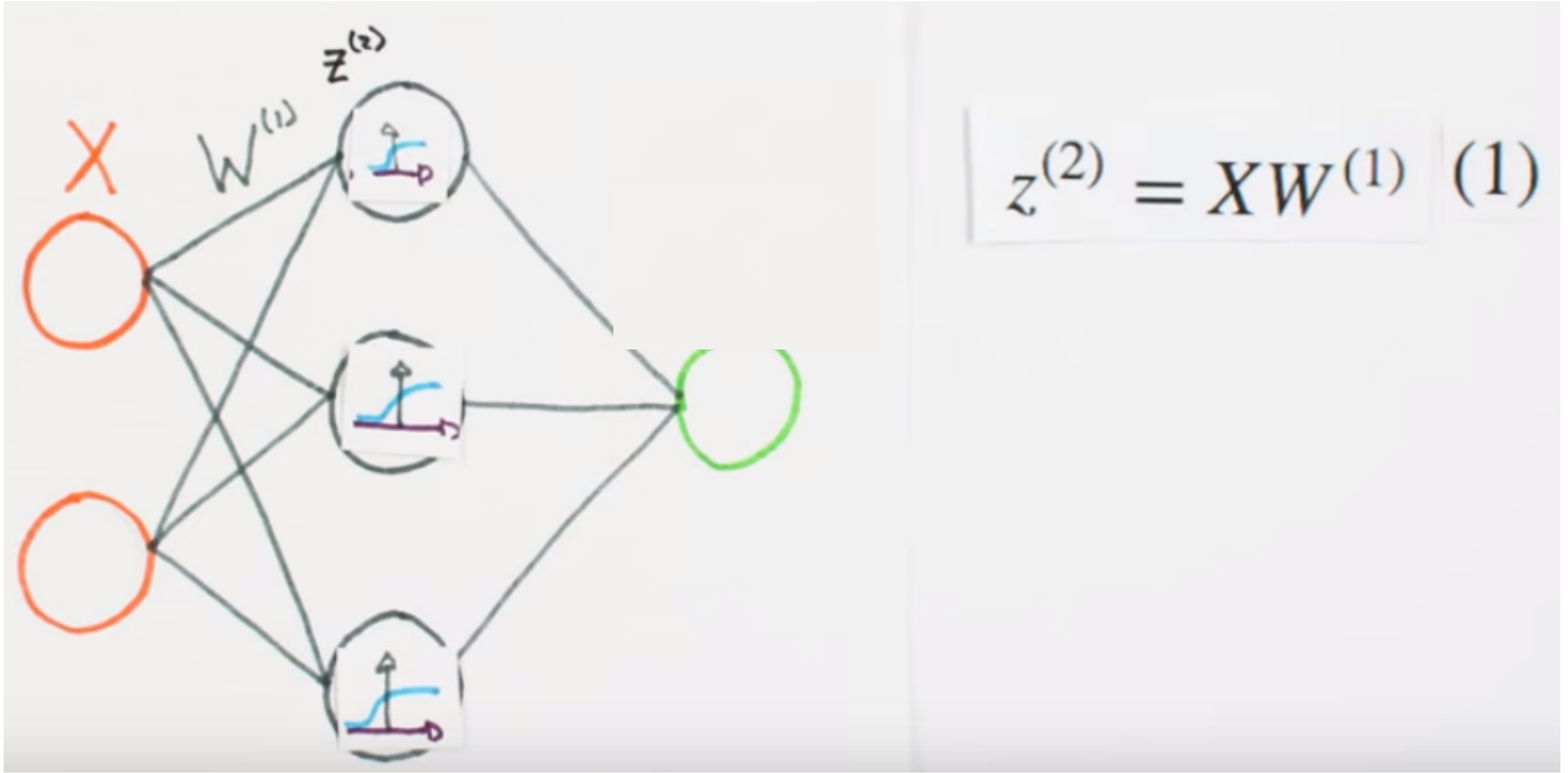
Example



Example

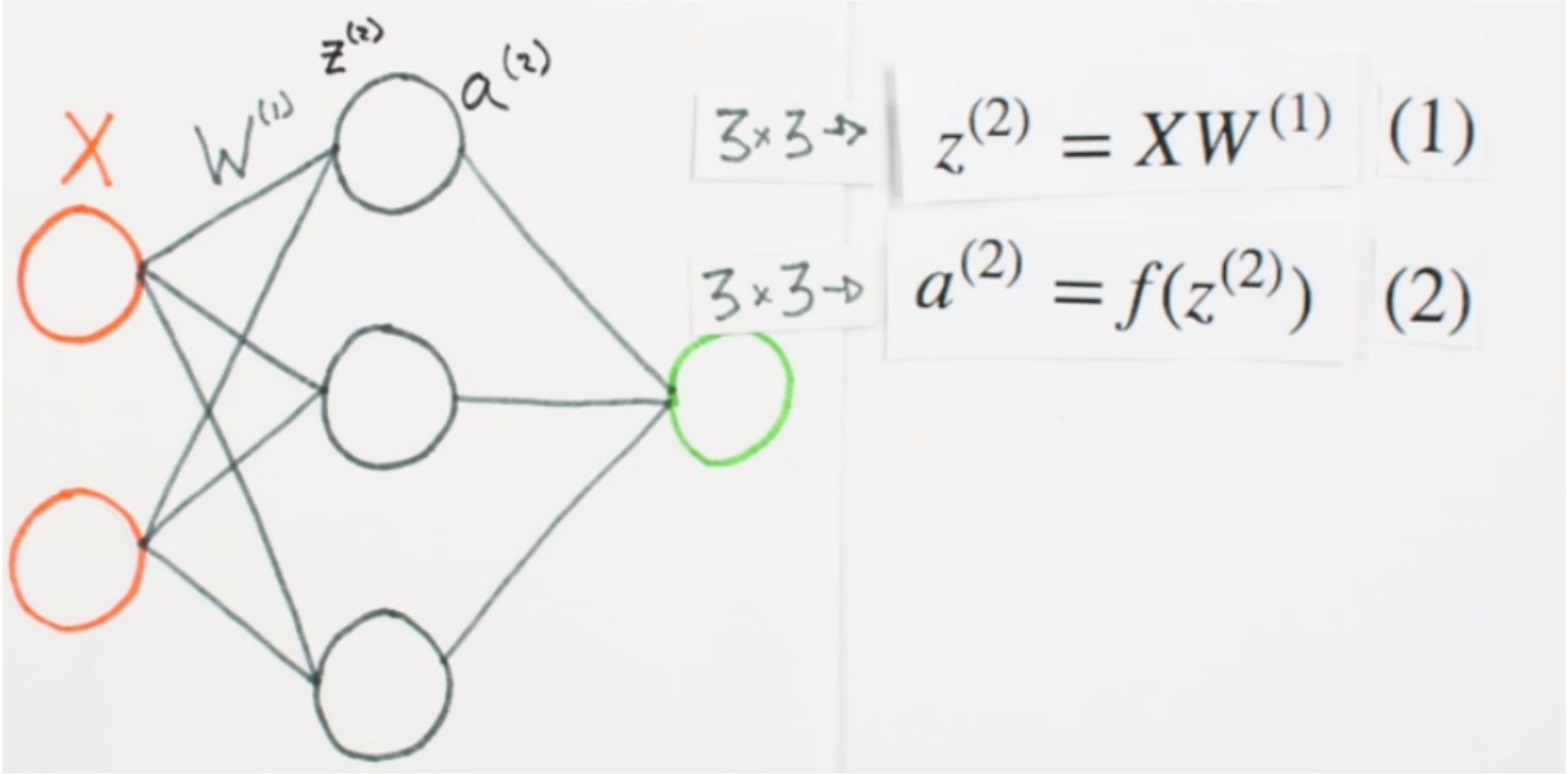

$\begin{bmatrix} 3 & 5 \\ 5 & 1 \\ 10 & 2 \end{bmatrix} \begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} & w_{13}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} & w_{23}^{(1)} \end{bmatrix} = \begin{bmatrix} 3w_{11}^{(1)} + 5w_{21}^{(1)} & 3w_{12}^{(1)} + 5w_{22}^{(1)} & 3w_{13}^{(1)} + 5w_{23}^{(1)} \\ 5w_{11}^{(1)} + 1w_{21}^{(1)} & 5w_{12}^{(1)} + 1w_{22}^{(1)} & 5w_{13}^{(1)} + 1w_{23}^{(1)} \\ 10w_{11}^{(1)} + 2w_{21}^{(1)} & 10w_{12}^{(1)} + 2w_{22}^{(1)} & 10w_{13}^{(1)} + 2w_{23}^{(1)} \end{bmatrix}$

Example

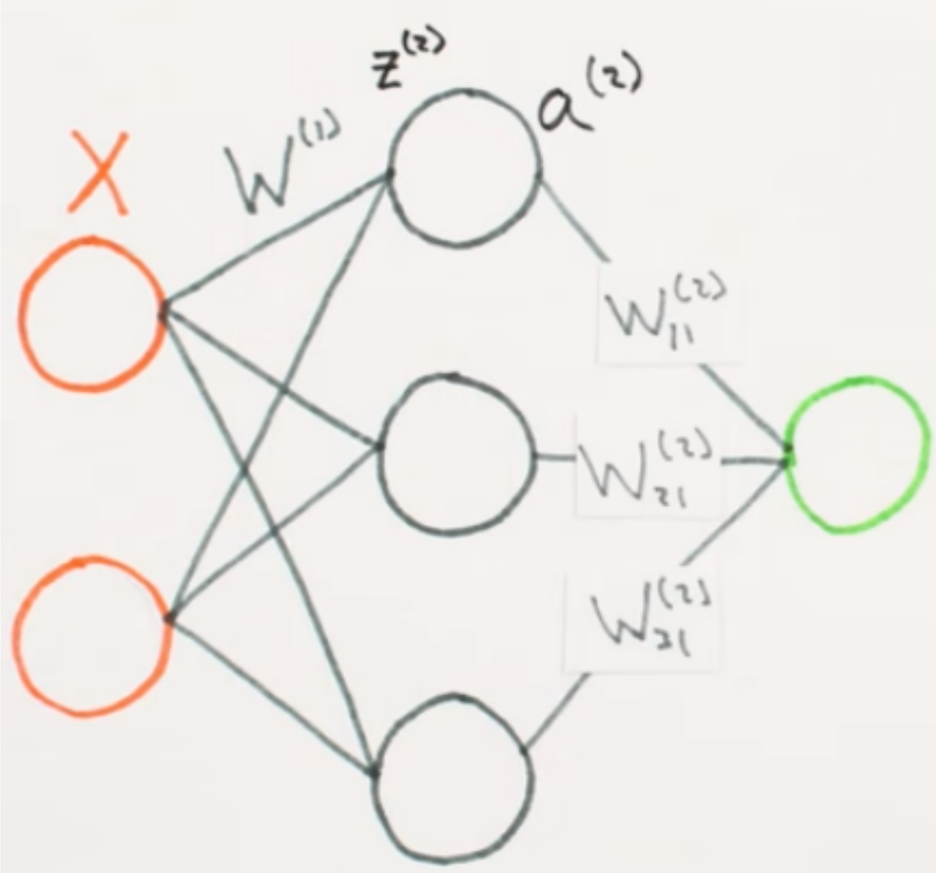


$$z^{(2)} = XW^{(1)} \quad (1)$$

Example



Example

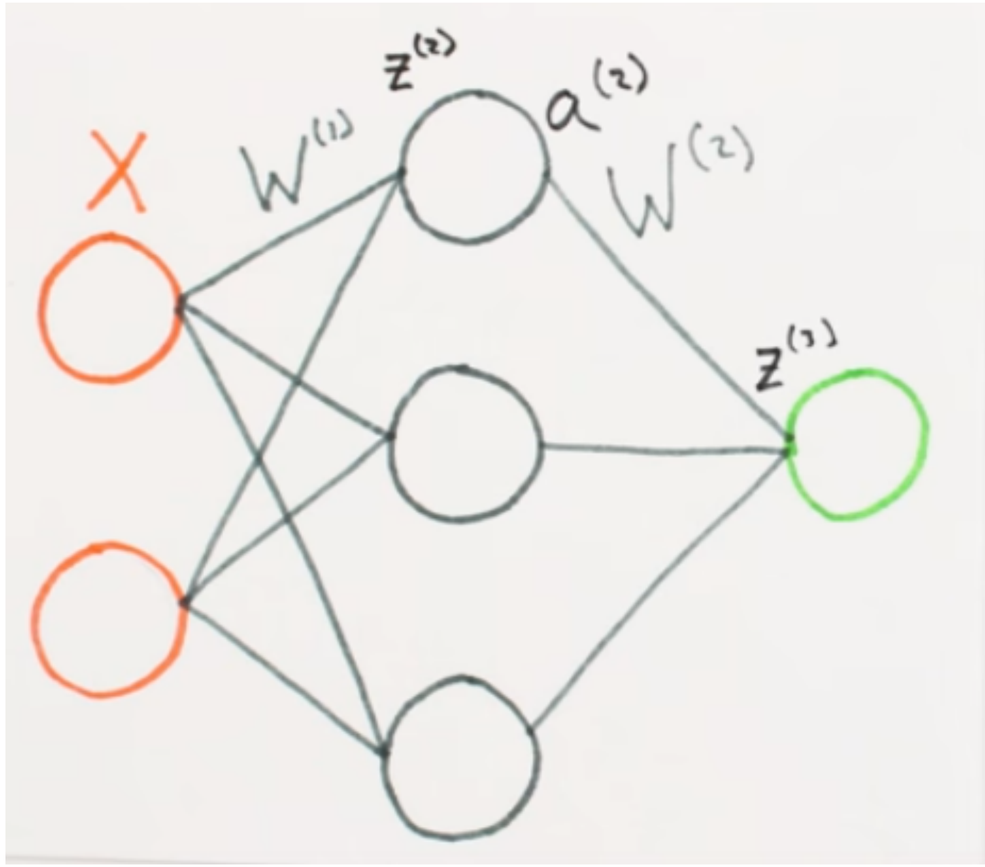


$$z^{(2)} = XW^{(1)} \quad (1)$$

$$a^{(2)} = f(z^{(2)}) \quad (2)$$

$$z^{(3)} = a^{(2)}W^{(2)} \quad (3)$$

Example



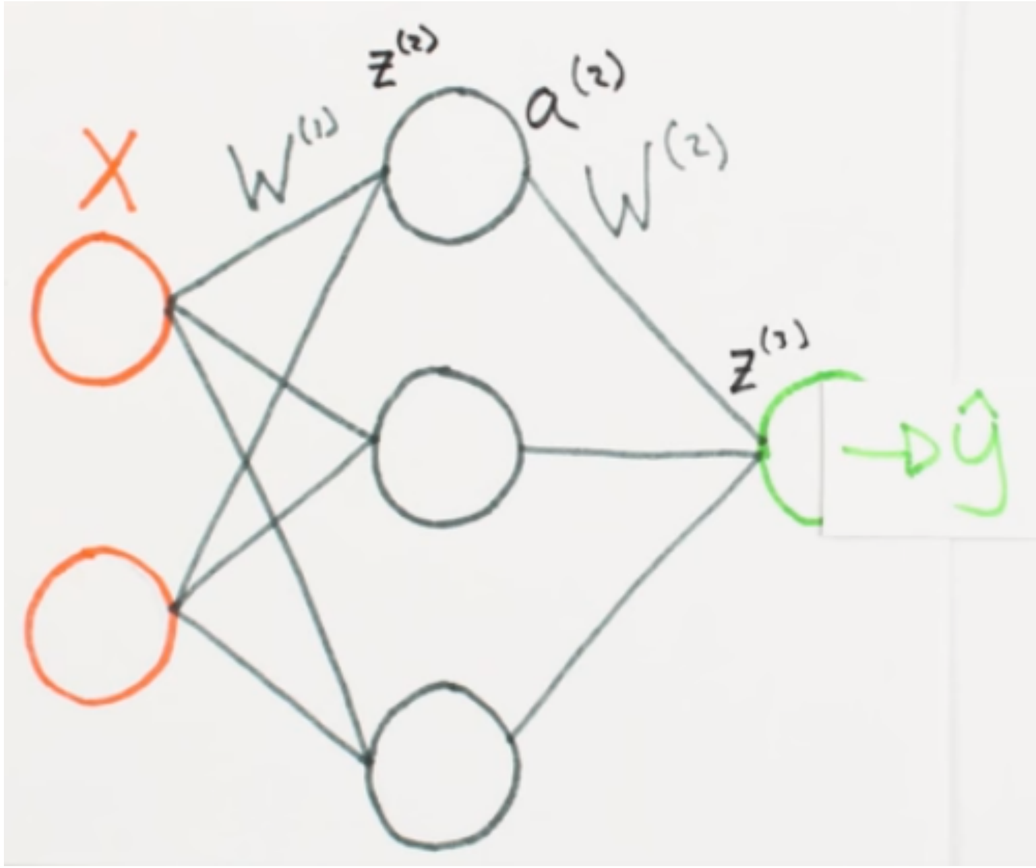
$$z^{(2)} = XW^{(1)} \quad (1)$$

$$a^{(2)} = f(z^{(2)}) \quad (2)$$

(3×1) (3×3) (3×1)

$$z^{(3)} = a^{(2)}W^{(2)} \quad (3)$$

Example



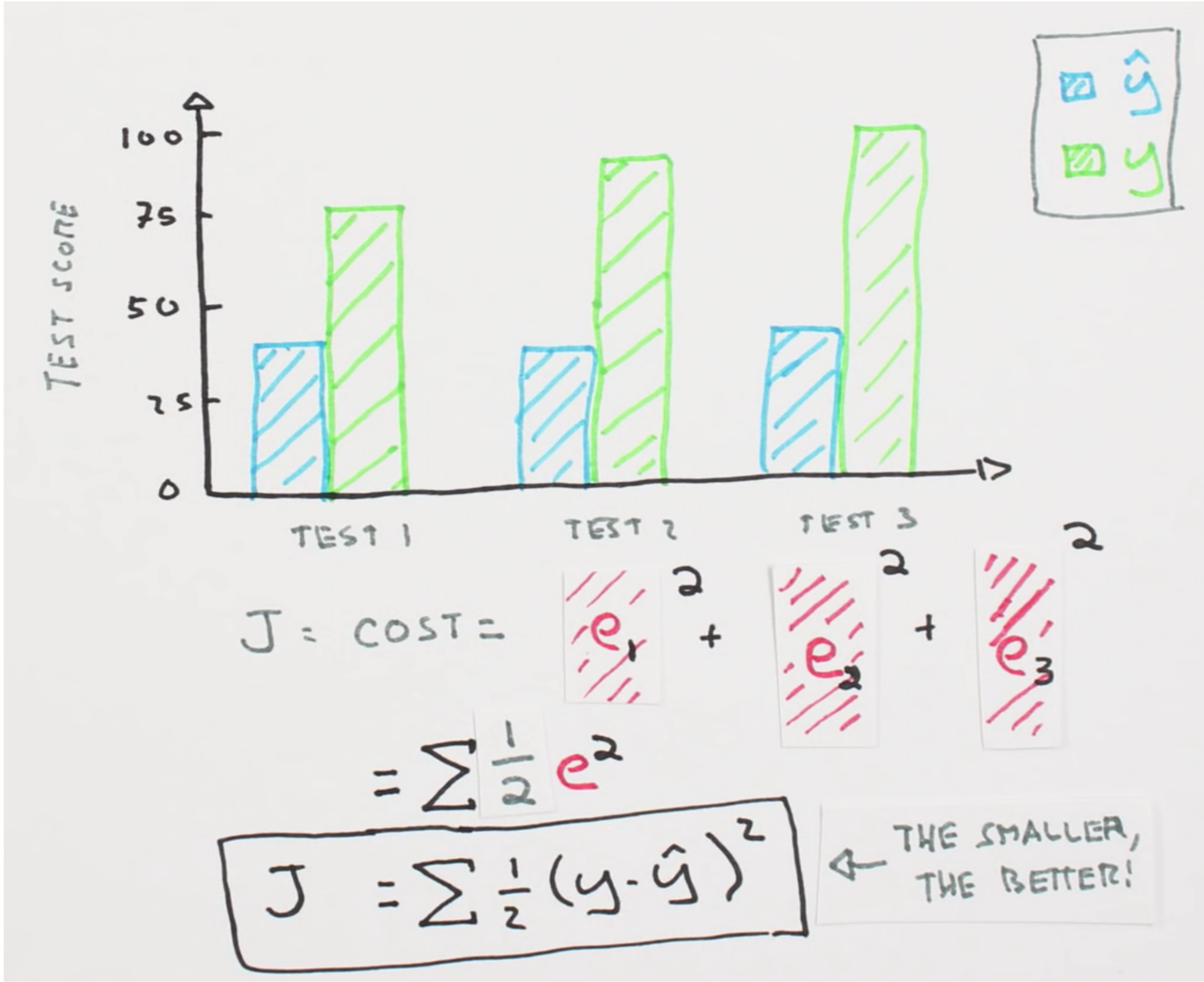
$$z^{(2)} = XW^{(1)} \quad (1)$$

$$a^{(2)} = f(z^{(2)}) \quad (2)$$

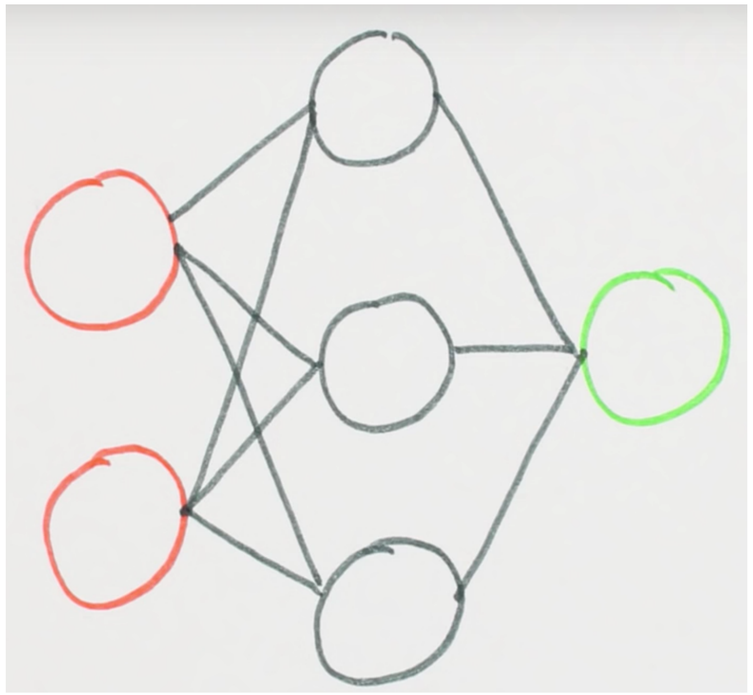
$$z^{(3)} = a^{(2)}W^{(2)} \quad (3)$$

$$\hat{y} = f(z^{(3)}) \quad (4)$$

Example


TEST SCORE
100
75
50
25
0
TEST 1
TEST 2
TEST 3
$\hat{y}$
y
$J = \text{COST} = e_1^2 + e_2^2 + e_3^2$
$= \sum \frac{1}{2} e^2$
$J = \sum \frac{1}{2}(y - \hat{y})^2$
THE SMALLER, THE BETTER!

Example

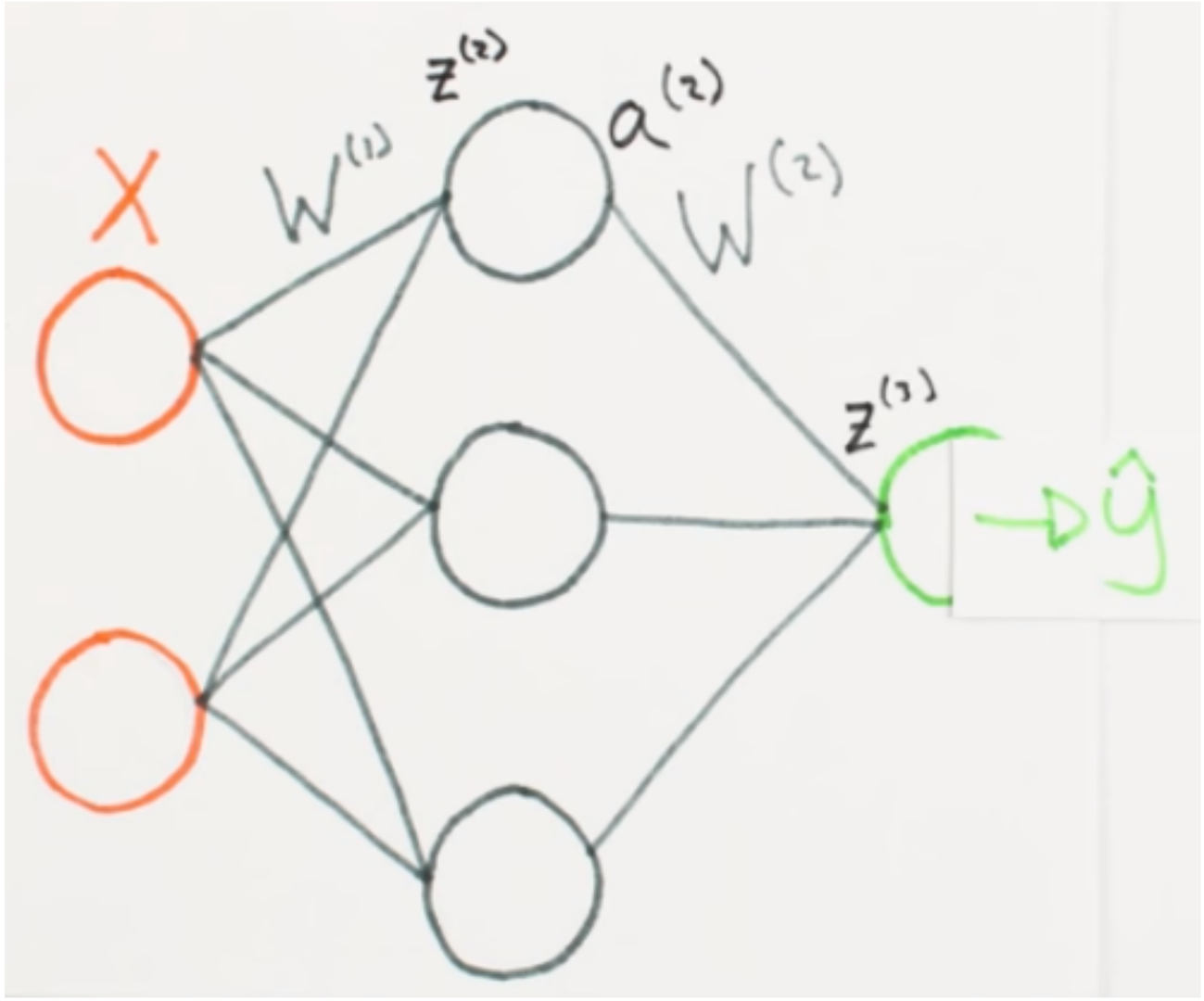


Training a Network

=

Minimizing a Cost Function

Example



$$z^{(2)} = XW^{(1)} \quad (1)$$

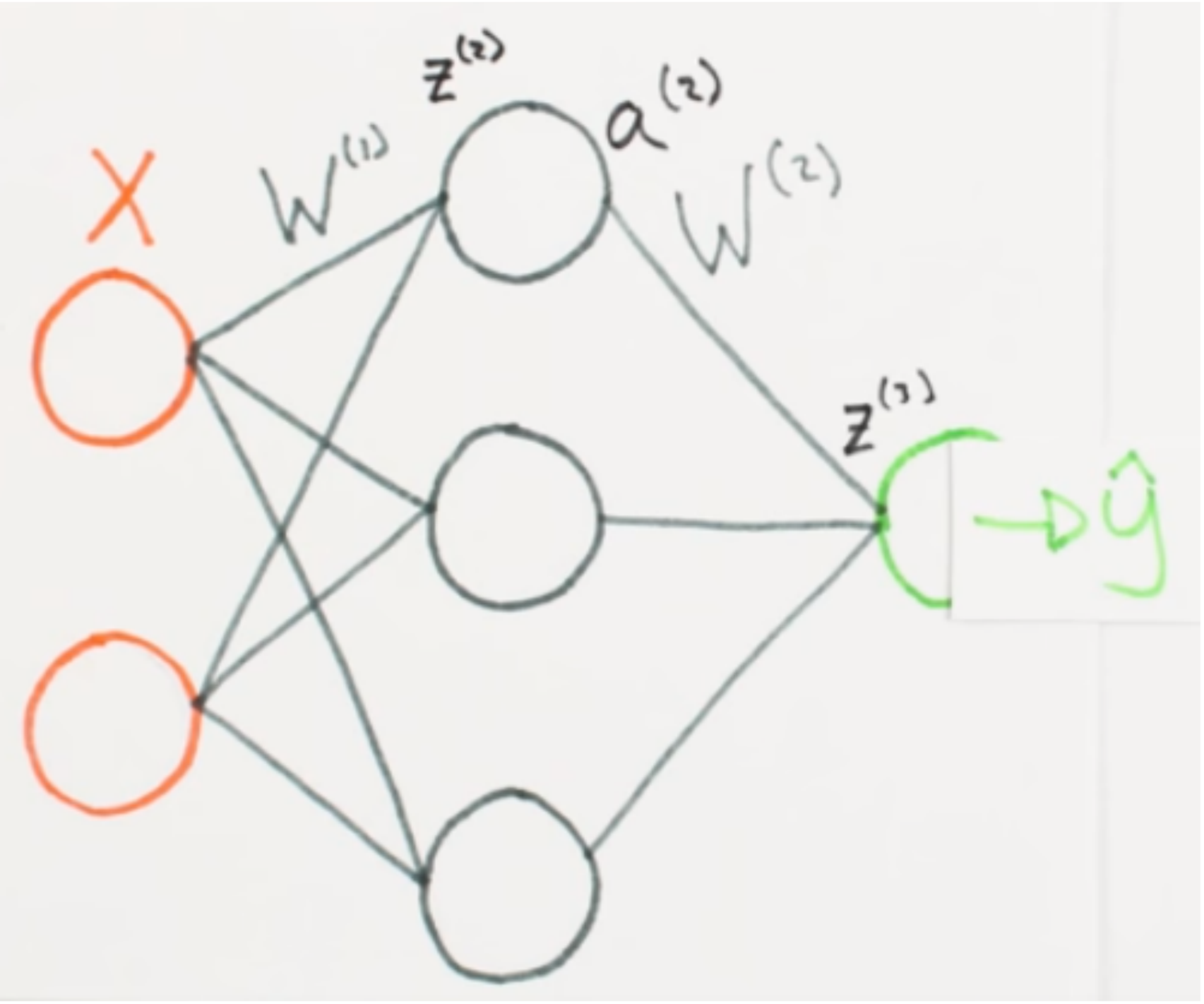
$$a^{(2)} = f(z^{(2)}) \quad (2)$$

$$z^{(3)} = a^{(2)}W^{(2)} \quad (3)$$

$$\hat{y} = f(z^{(3)}) \quad (4)$$

$$J = \sum \frac{1}{2}(y - \hat{y})^2 \quad (5)$$

Example



$$z^{(2)} = XW^{(1)} \quad (1)$$

$$a^{(2)} = f(z^{(2)}) \quad (2)$$

$$z^{(3)} = a^{(2)}W^{(2)} \quad (3)$$

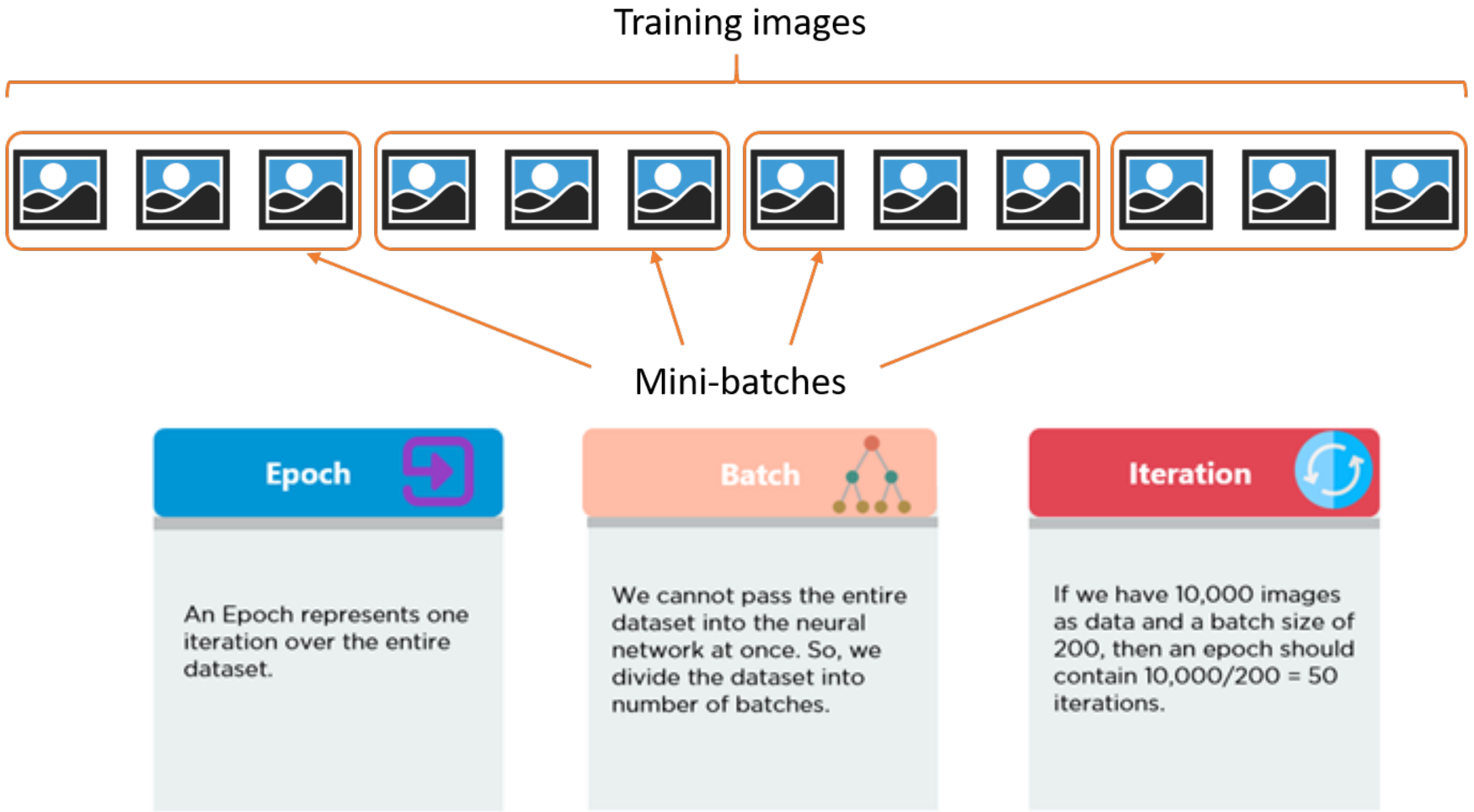
$$\hat{y} = f(z^{(3)}) \quad (4)$$

$$J = \sum \frac{1}{2}(y - \hat{y})^2 \quad (5)$$

$$J = \sum \frac{1}{2}(y - f(f(XW^{(1)})W^{(2)}))^2$$

Epochs, Batch

- Average error, gradients



Resources

- https://playground.tensorflow.org/
- https://betterexplained.com/articles/derivatives-product-power-chain/
- http://www.3blue1brown.com/videos/2017/10/9/neural-network
- http://cs231n.github.io/neural-networks-2/
- http://cs231n.github.io/neural-networks-3