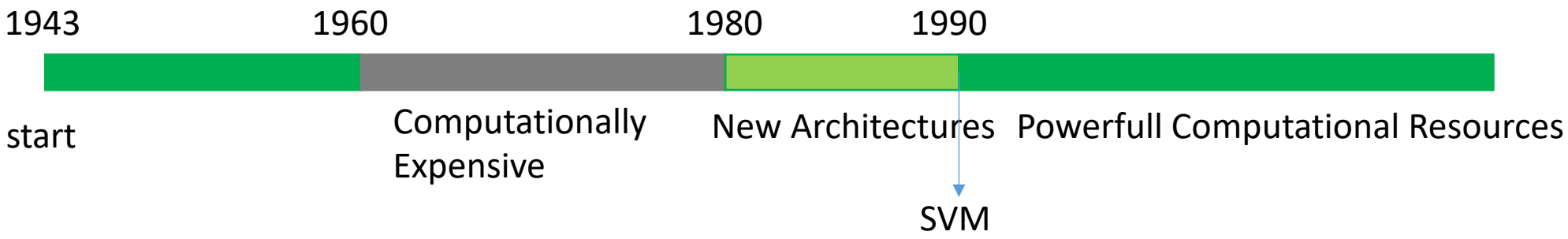
یادگیری عمیق با تنسورفلو و کراس در پایتون

فصل دوم: معرفی شبکه های عصبی مصنوعی با Keras

پژمان اقبالی

PhD Student in Biomechanics

EPFL



1) **Huge Data**

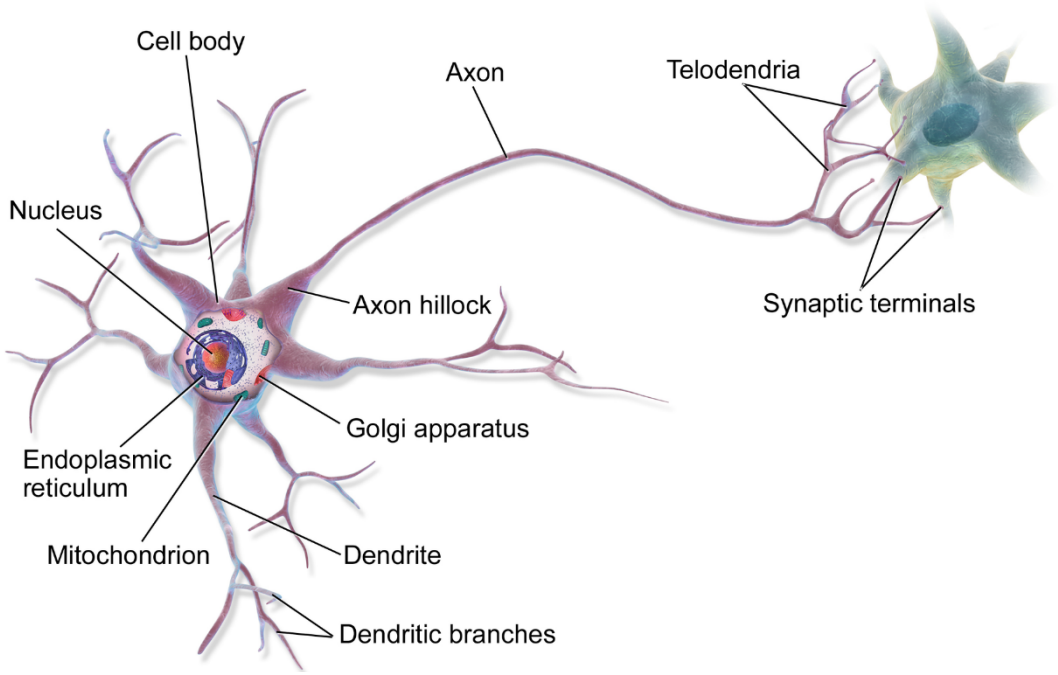
2) **Computing Power**

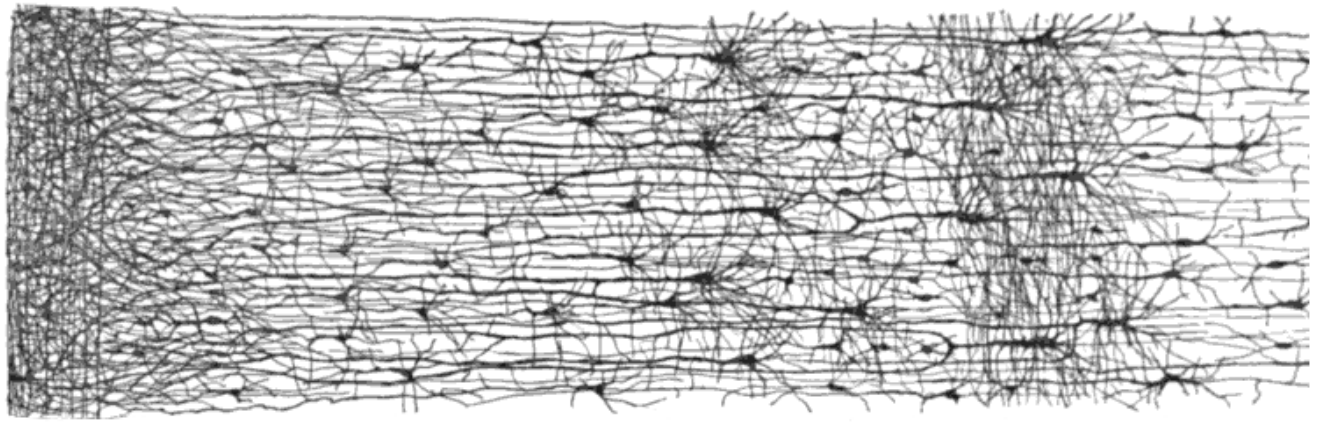
3) **Training Algorithms**

4) **Resolved limitations**

5) **Money**

(Neuron) نورون





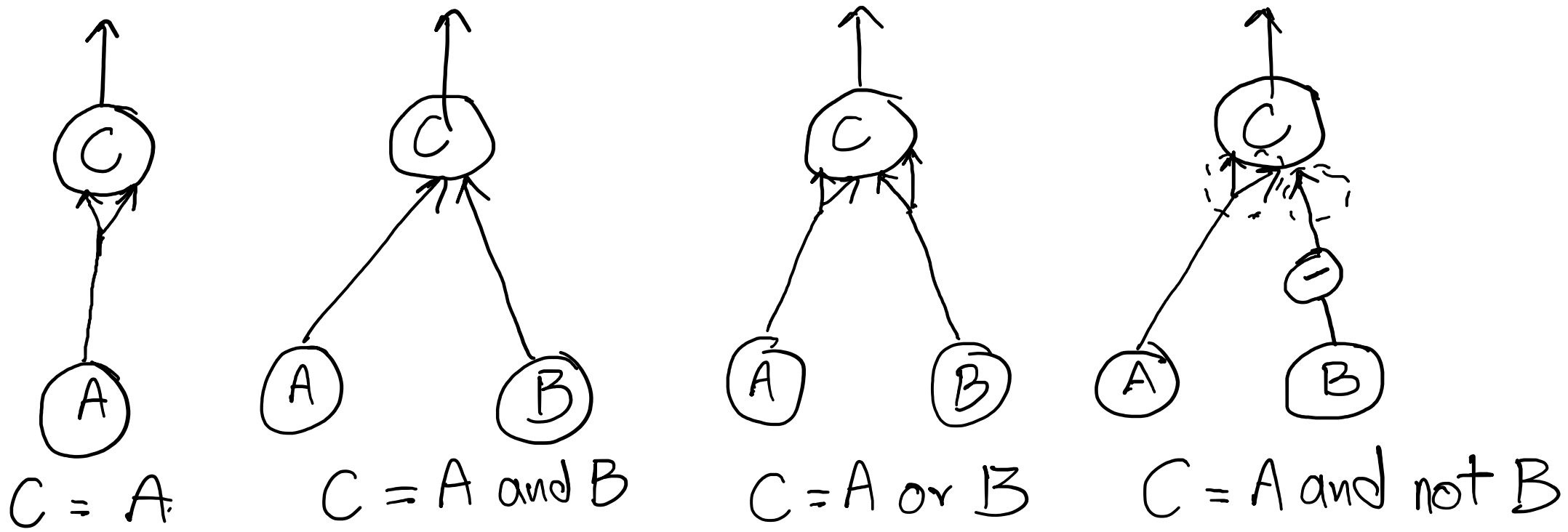
https://en.wikipedia.org/wiki/Neuron

https://en.wikipedia.org/wiki/Cerebral_cortex

(Neuron) نورون

"A logical calculus of the ideas immanent in nervous activity"

Warren S. McCulloch and Walter Pits



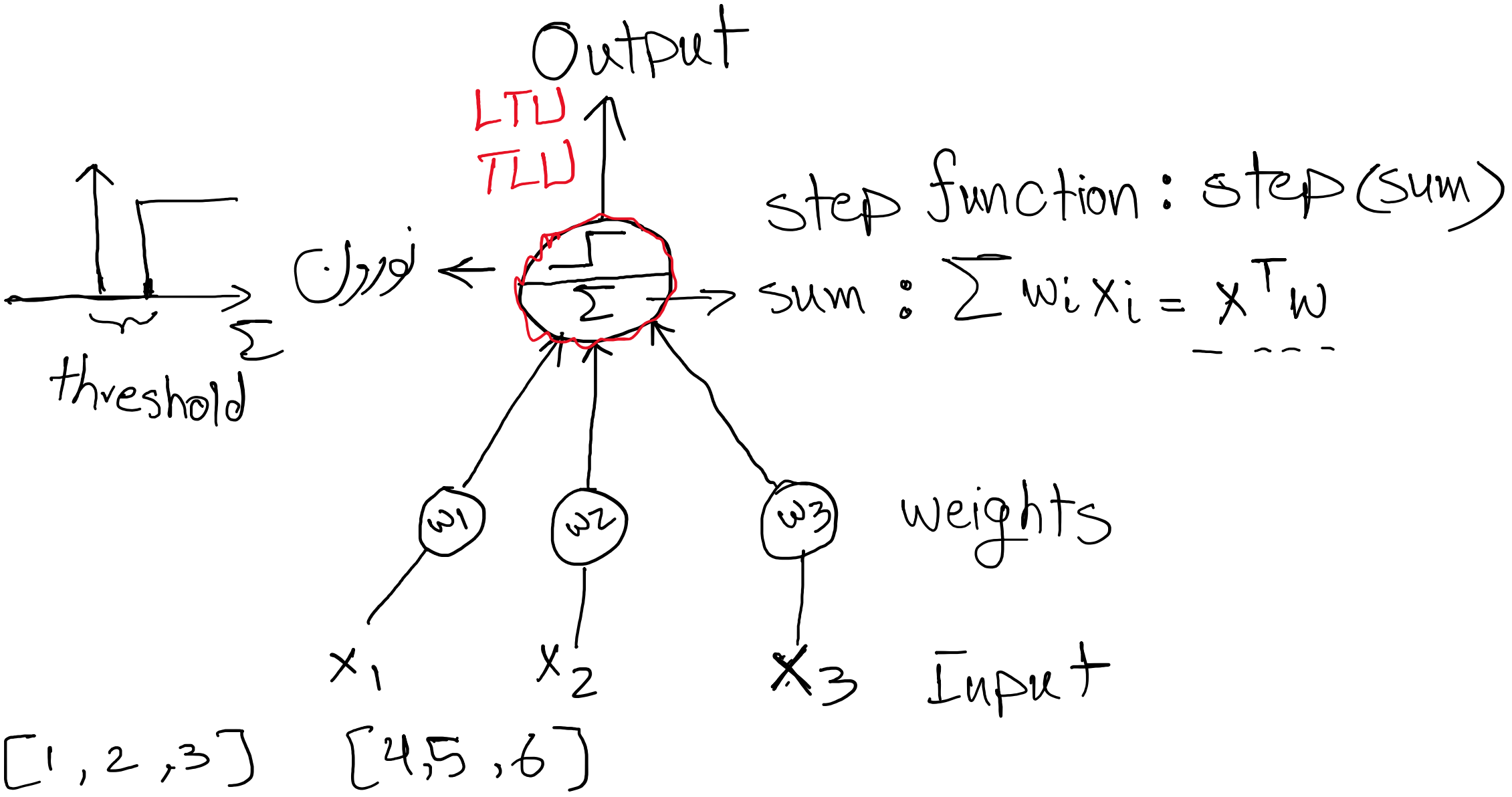
(Perceptron) پرسپترون

"The perceptron: A probabilistic model for information storage and organization in the brain"
Frank Rosenblatt 1957

"The Organization of Behavior" (Wiley)
Donald Hebb

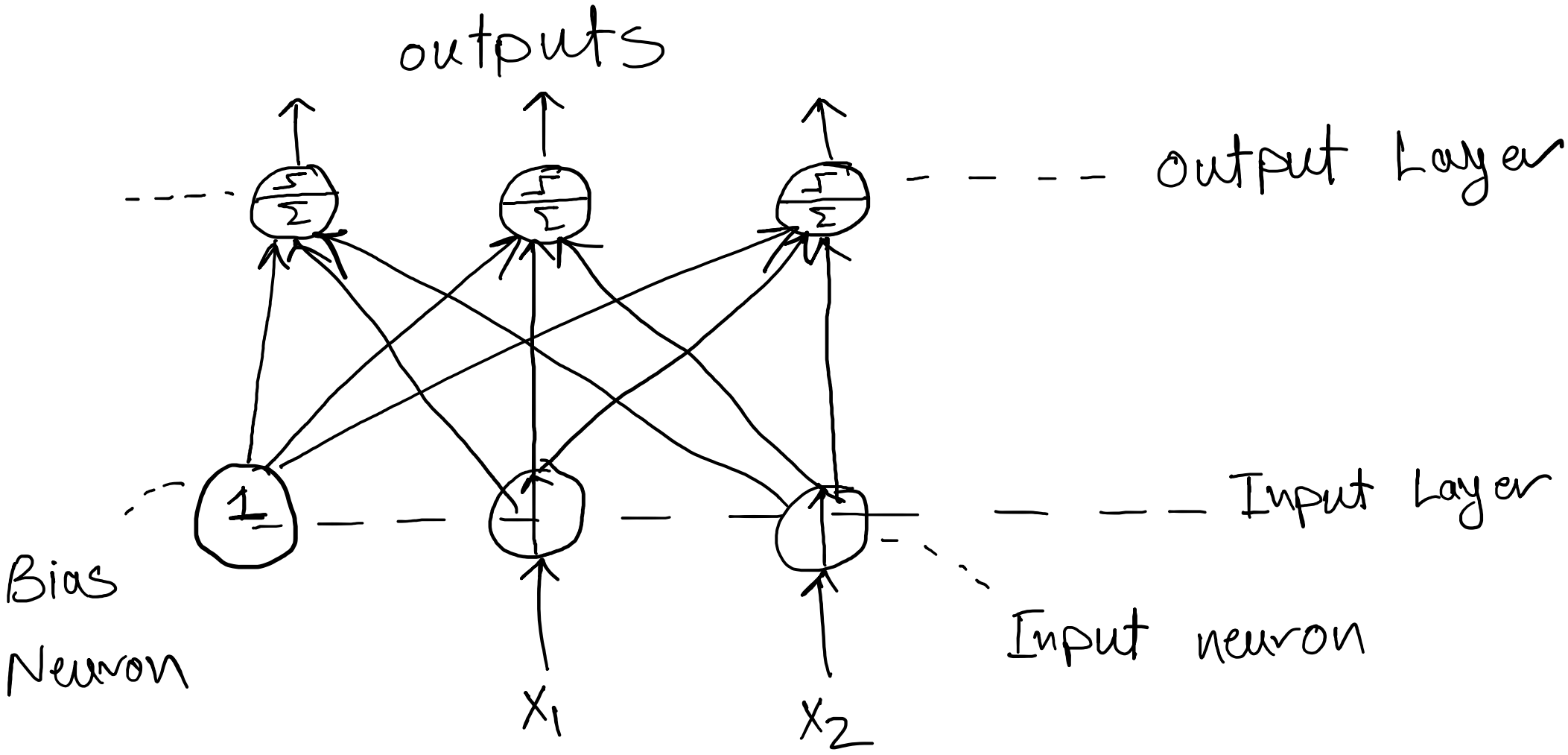
Hebb's rule
"cells that five together, wire together"

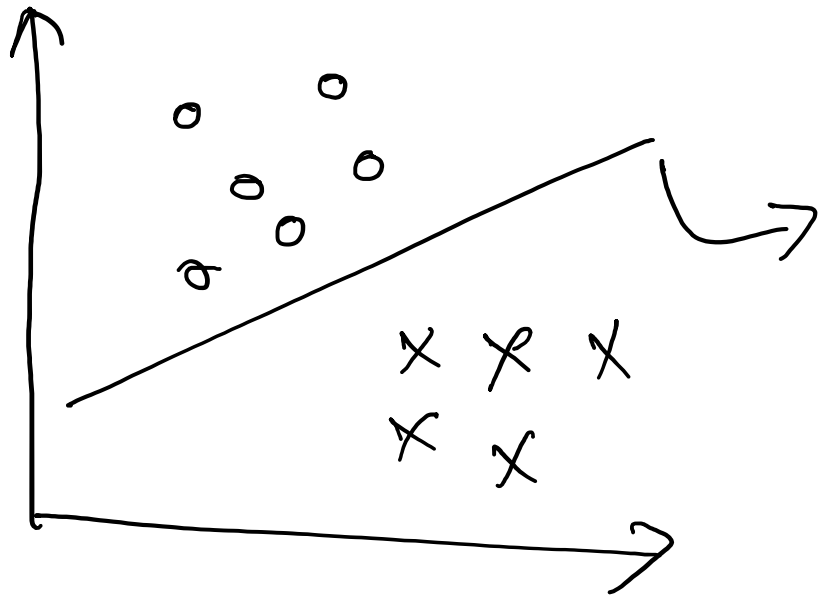
"Perceptrons"
Marvin Minsky and Seymour Papert 1969


Output
LTU
TLU
threshold
Σ
نورون
step function : step(sum)
sum : $\sum w_i x_i = x^T w$
w1
w2
w3
weights
x_1
x_2
x_3
Input
[1, 2, 3]
[4, 5, 6]

$$\text{heaviside}(x): \begin{cases} 0 & \text{if } x<0 \\ 1 & \text{if } x>0 \end{cases}$$

$$\text{sgn}(x): \begin{cases} -1 & \text{if } x<0 \\ 0 & \text{if } x=0 \\ +1 & \text{if } x>0 \end{cases}$$


outputs
output Layer
Input Layer
1
Bias
Neuron
Input neuron
X_1
X_2



$$y = wx + b \times 1$$

$$= w \cdot X$$

w $\quad$ b

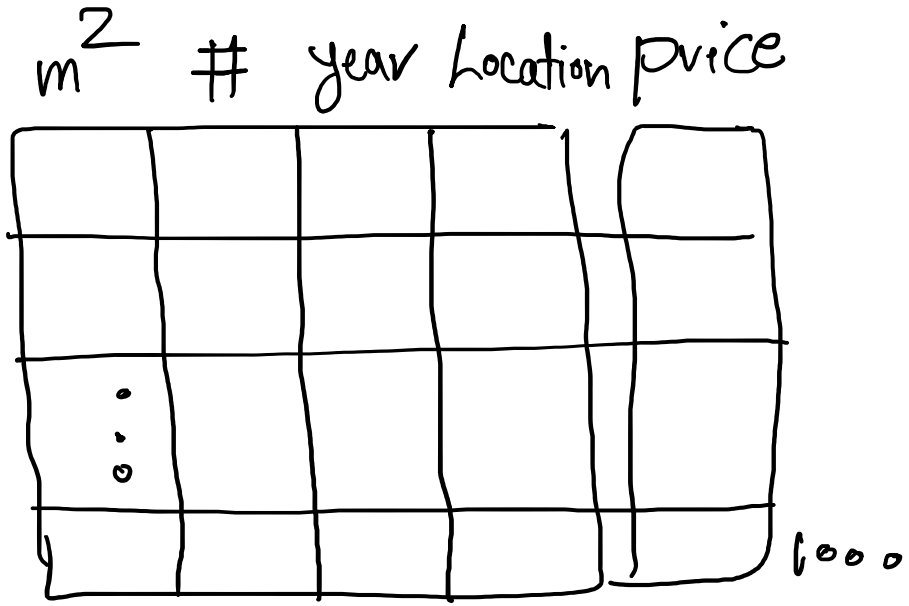
feature $\quad$ 1

$$\begin{bmatrix} \vdots & 1 \\ \vdots & 1 \\ \vdots & 1 \end{bmatrix}$$

Output : $h_\theta(x) = \text{activation\_func}(xW + b)$

X : matrix of features

Ex : X -> shape -> (1000, 4)



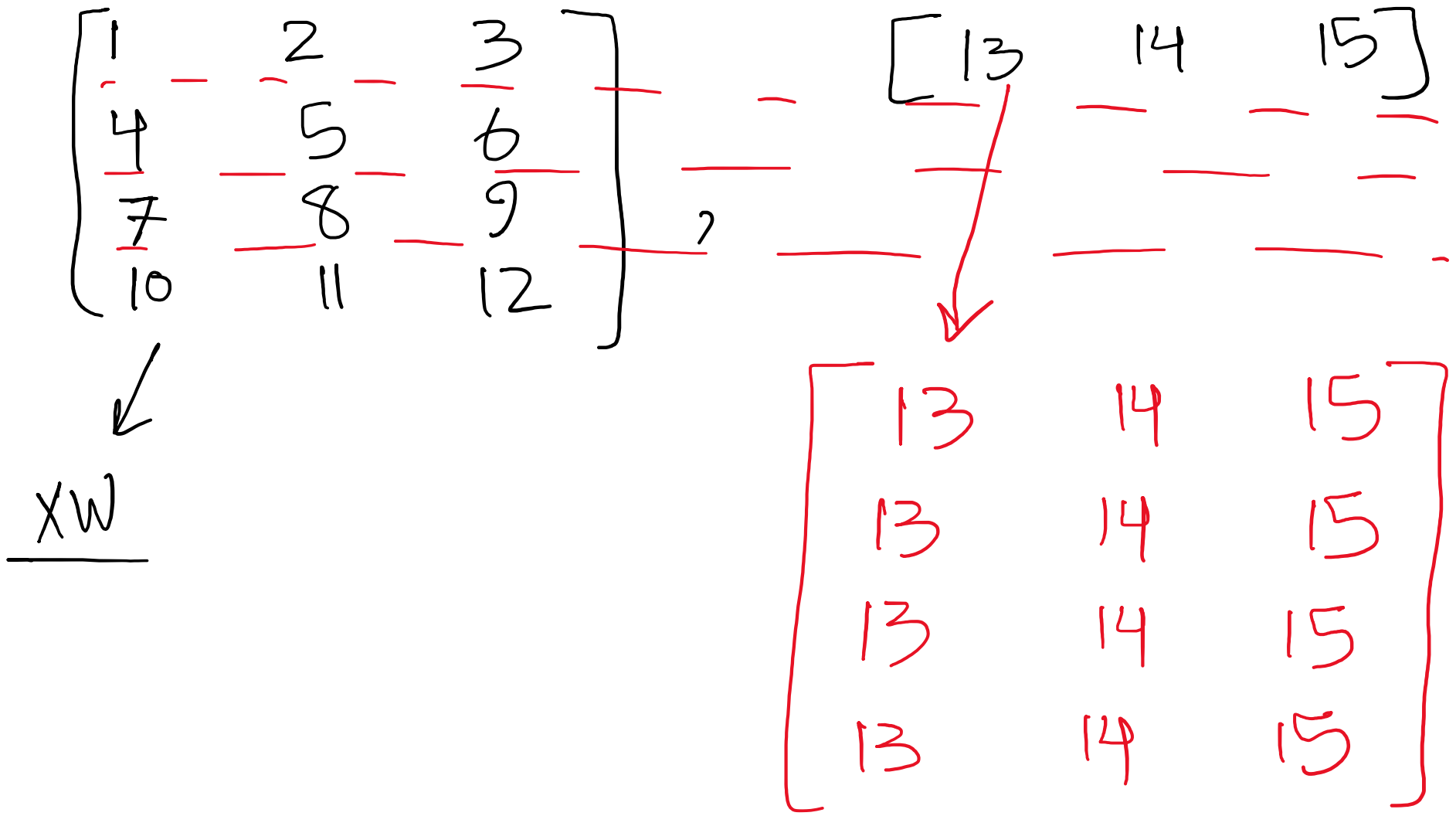
W : matrix of weights

Ex : w.shape -> (4, 1)

(1000, 4) * (4, 1) -> (1000, 1)

b : bias vector

Ex : b.shape -> (1, 1)


1 2 3
4 5 6
7 8 9
10 11 12
13 14 15
?
XW
13 14 15
13 14 15
13 14 15
13 14 15

(next step) (step)

$$w_{j,i} = w_{j,i} + \eta (y_i - \hat{y}_i) x_j$$

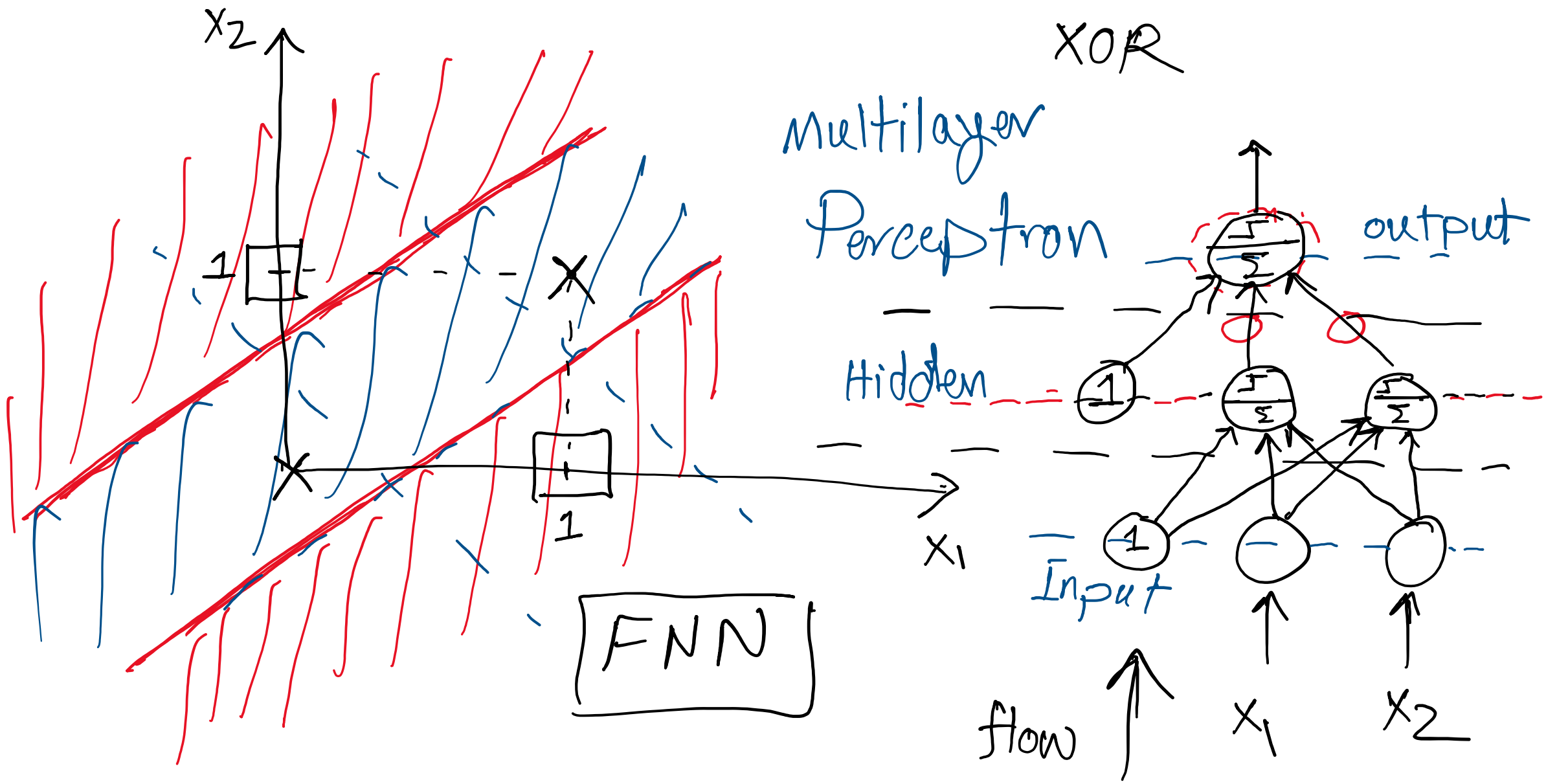
i : output

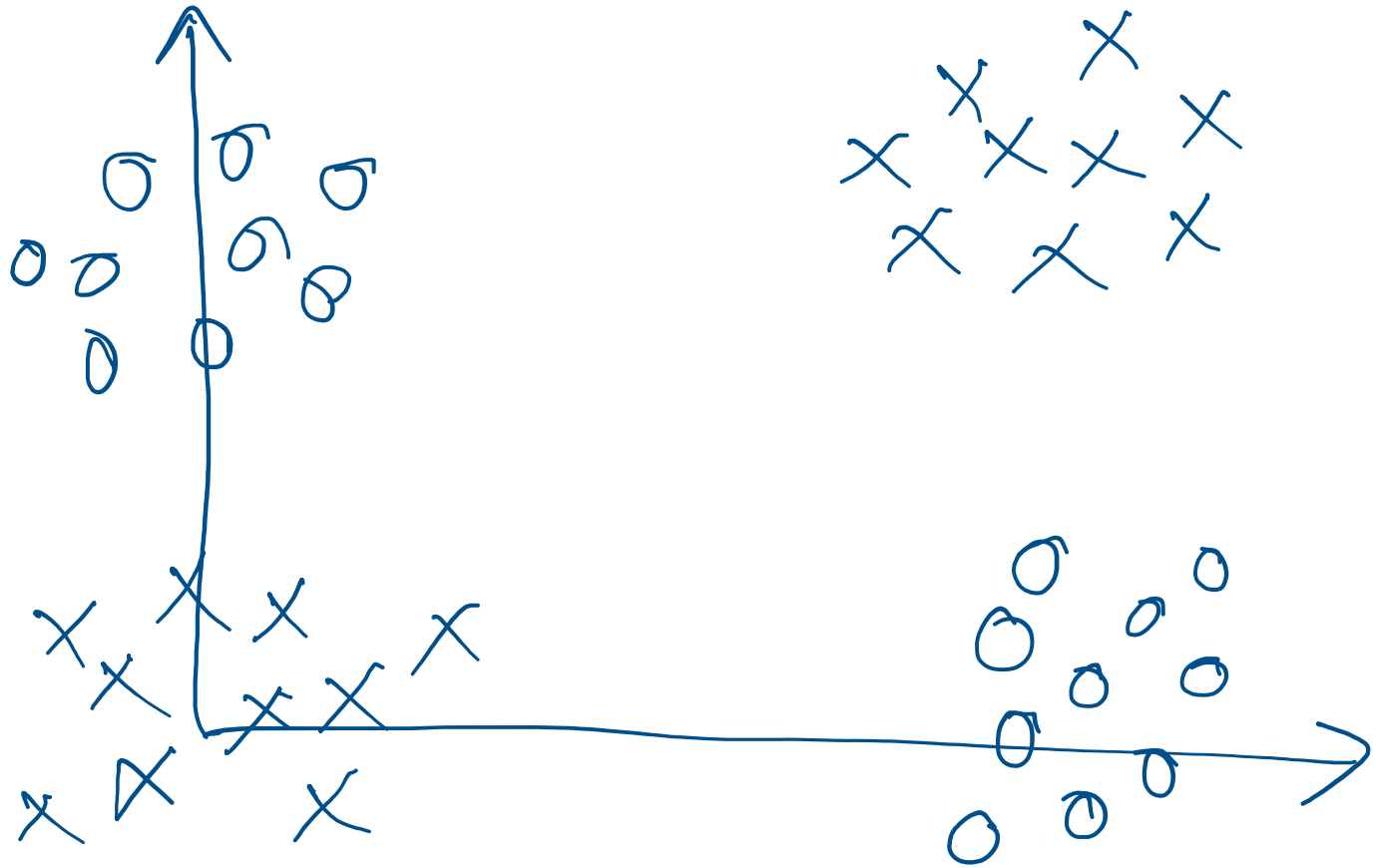
j : input

x_j : current input value

$$\Delta w = -\eta \frac{dE}{dw}$$

"Perceptron convergence Theorem"


XOR
Multilayer
Perceptron
output
Hidden
Input
flow
x_1
x_2
x_1
x_2
1
1
1
FNN



The Multilayer Perceptron and Backpropagation

"Learning Internal Representations by Error Propagation"

David Rumelhart, Geoffrey Hinton, Ronald Williams

1986

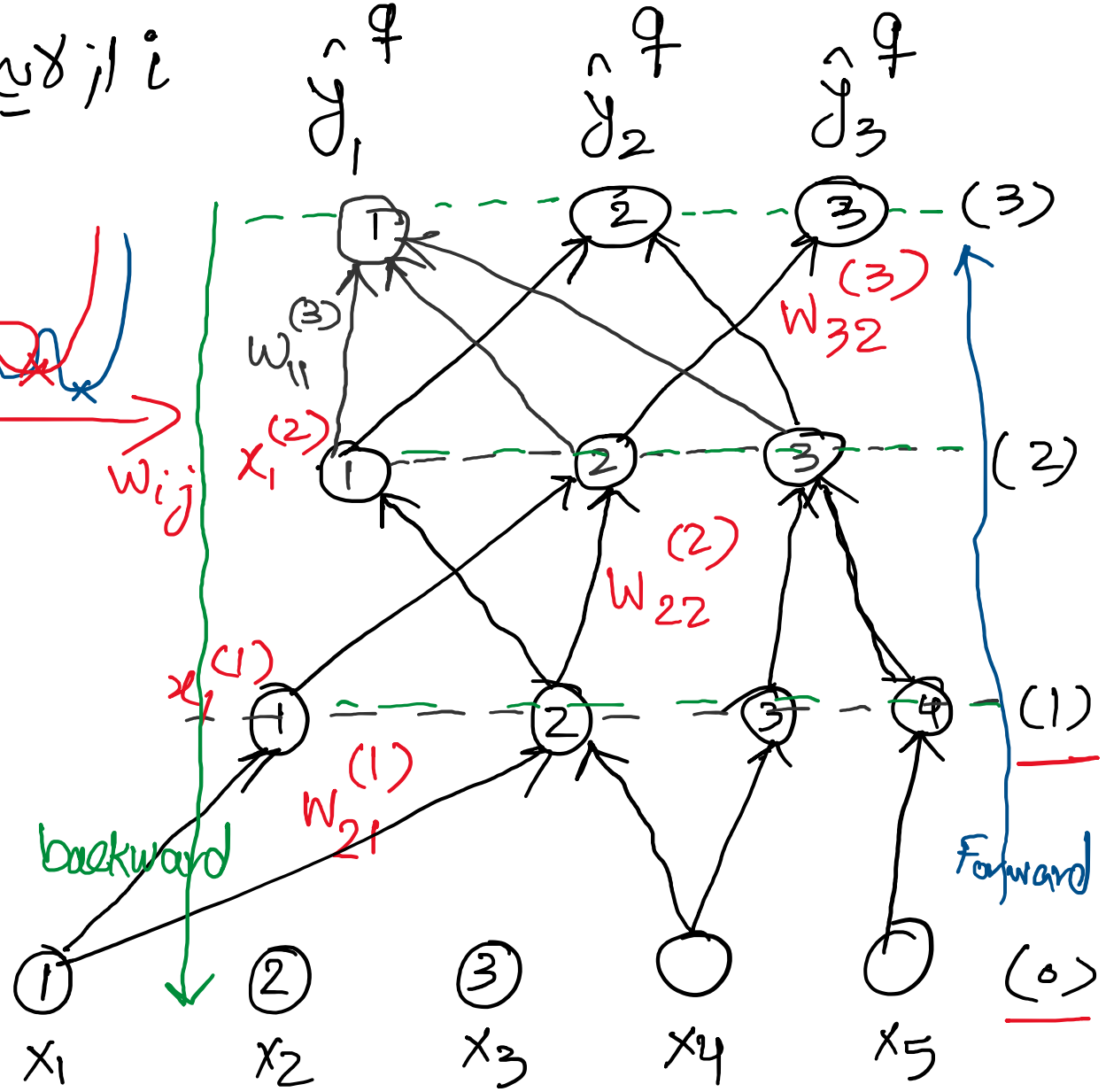
"autodiff"

$$E(w) = \frac{1}{2} \sum_{k=1}^{M} \sum_{i} \left[t_i^k - \hat{y}_i^k \right]^2$$

$w_{ij}^{(n)}$: وزن‌های نورون i از لایه‌ی $(n-1)$ را به نورون j از لایه‌ی n متصل می‌کند

$$\Delta w_{ij}^{(n)} = -\gamma \frac{dE}{dw_{ij}^{(n)}}$$

$$\begin{cases} \hat{y}_i^q = af^{(3)}\left(\sum_j w_{ij}^{(3)} x_j^{(2)}\right) \\ x_j^{(2)} = af^{(2)}\left(\sum_k w_{jk}^{(2)} x_k^{(1)}\right) \\ x_k^{(1)} = af^{(1)}\left(\sum_l w_{kl}^{(1)} x_l^{(0)}\right) \end{cases}$$



1) تعریف متغیرهای مبانی

$x_i^{(n)}$: neuron output

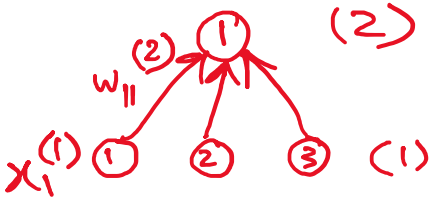
$\beta_i^{(n)}$: drive of neuron

$$\beta_i^{(n)} = \sum_l w_{il}^{(n)} x_l^{(n-1)} = \sum_l w_{il}^{(n)} af\left(\beta_l^{(n-1)}\right)$$

$$\delta_K^{(n)} = \frac{dE}{d\beta_K^{(n)}}$$

Ex: $\beta_1^{(2)} = w_{11}^{(2)} x_1^{(1)} + w_{12}^{(2)} x_2^{(1)} + w_{13}^{(2)} x_3^{(1)}$

$$\frac{d\beta_1^{(2)}}{dw_{11}^{(2)}} = x_1^{(1)}$$



2 ج به روز رسانی فرایند (وزن ها)

$$\Delta w_{ij}^{(n)} = -\gamma \frac{dE}{dw_{ij}^{(n)}} = -\gamma \frac{dE}{dB_i^{(n)}} \frac{dB_i^{(n)}}{dw_{ij}^{(n)}}$$

$$\Delta w_{ij}^{(n)} = -\gamma \delta_i^{(n)} x_j^{(n-1)}$$

$$\delta_i^{(n)} \propto \delta_i^{(n-1)}$$

$$\delta_3^{(2)} = \frac{dE}{d\beta_3^{(2)}} = \sum_j \frac{dE}{d\beta_j^{(3)}} \frac{d\beta_j^{(3)}}{d\beta_3^{(2)}}$$

// تکرار وابستگی (3

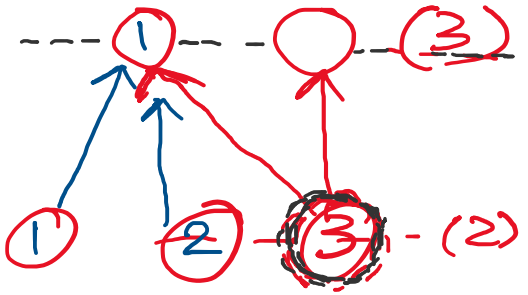
$$= \sum_j \delta_j^{(3)} w_{j3}^{(3)} (af)'$$

Ex: $\beta_1^{(3)} = w_{11}^{(3)} af^{(2)}(\beta_1^{(2)}) + w_{12}^{(3)} af^{(2)}(\beta_2^{(2)}) + \cdots$

$$\frac{d\beta_1^{(3)}}{d\beta_2^{(2)}} = w_{12}^{(3)} (af^{(2)})'$$

$5\sin(x)$

$5\cos(x)$



گراف وابستگی

$$\delta_k^{(n-1)} = \sum_j \delta_j^{(n)} w_{jk}^{(n)} (af)'$$

الگوریتم back propagation

0- مقداردهی اولیه‌ی وزن‌ها

1- انتخاب یک داده x^q

2- انتشار اطلاعات به سمت جلو‌ی شبکه

$$x_j^{(n)} = af^{(n)}(\beta_j^{(n)})$$

$$= af^{(n)}\left(\sum_k w_{jk}^{(n)} x_k^{(n-1)}\right) \Rightarrow \text{output: } \hat{y}_i^q = x_i^{(\text{last layer})}$$

3- محاسبه‌ی خطا در خروجی

$$\delta_i^{(\text{last layer})} = \left(af\left(\beta_i^{(\text{last layer})}\right)\right)' \left[\hat{y}_i^q - t_i^q\right]$$

4- انتشار خطا در مسیر برگشت

$$\delta_i^{(n)} \longrightarrow \delta_j^{(n-1)}$$

$$\delta_j^{(n-1)} = \left(af^{(n-1)}\left(\beta_j^{(n-1)}\right)\right)' \sum_i w_{ij}^{(n)} \delta_i^{(n)}$$

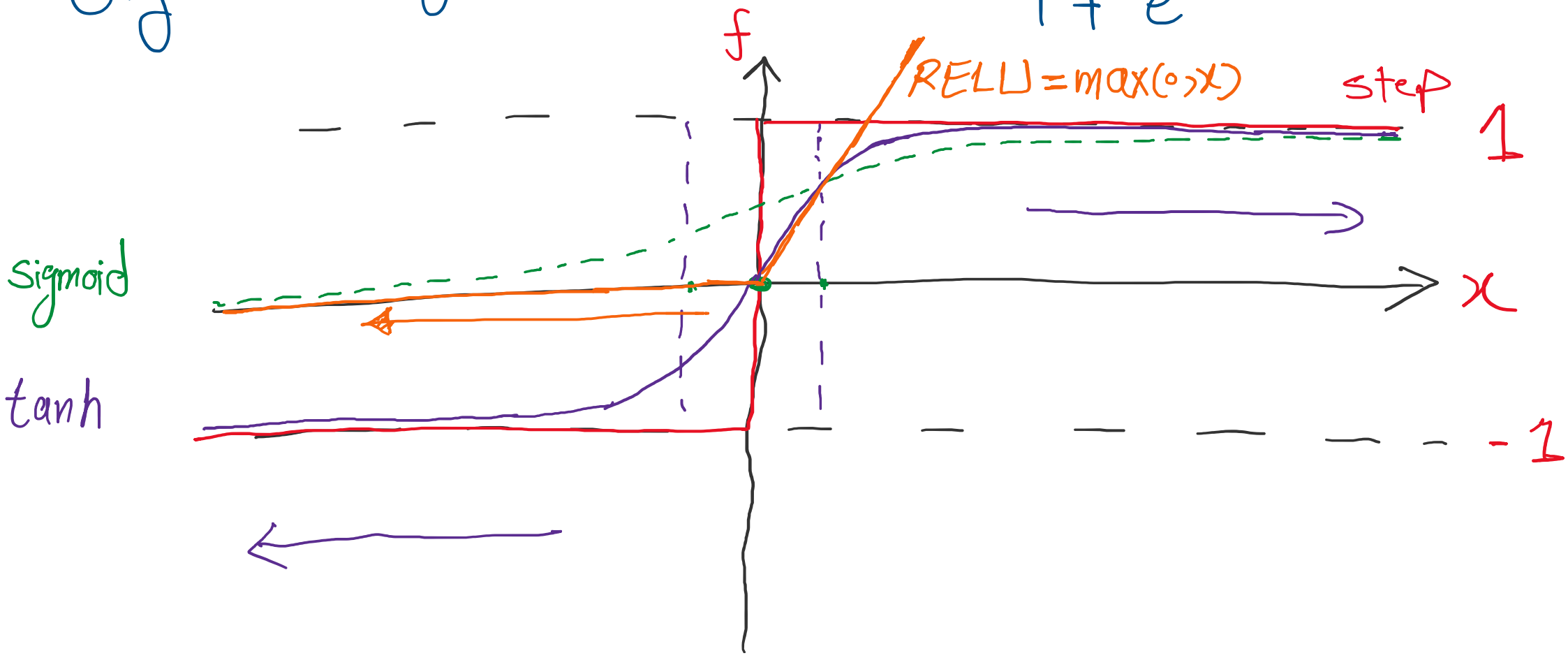
5- به روزرسانی ضرایب (وزن‌ها)

$$\Delta w_{ij}^{(n)} = -\gamma \delta_i^{(n)} x_j^{(n-1)}$$

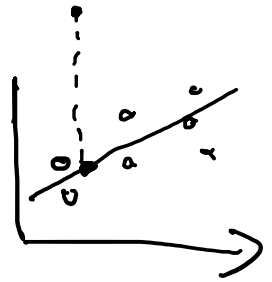
6- برگردیم سراغ 1

Activation Function

Sigmoid (logistic) $\sigma(z) = \frac{1}{1 + e^{-z}}$

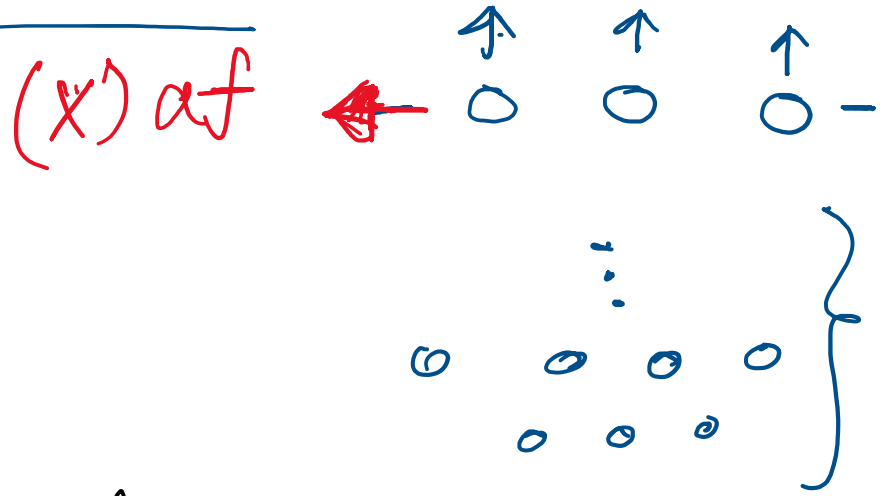


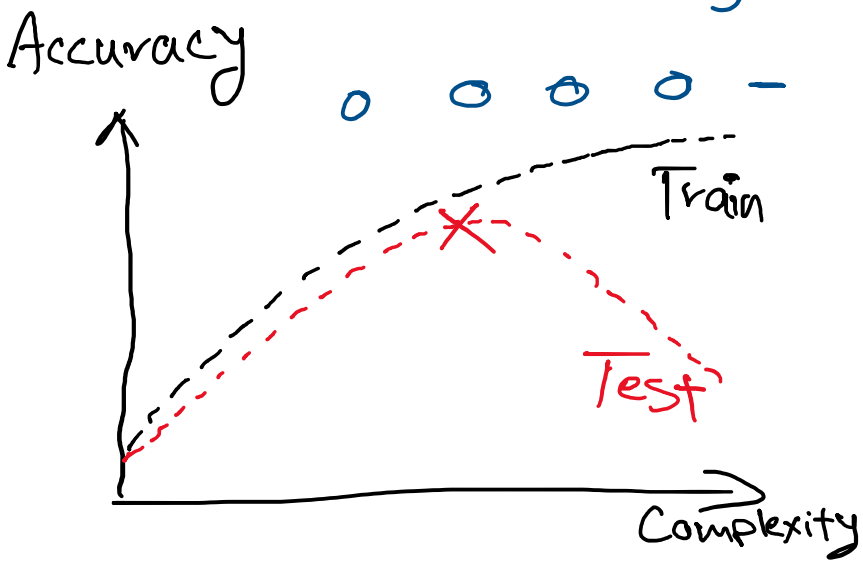
رگرسیون (Regression MLPs)



one output neuron per output dimension

Hyperparameter	مقادیر معمول
تعداد نورون ورودی	یکی به ازای هر feature
چند تا لایه ی مخفی	1 - 5
چند نورون در هر لایه ی مخفی	10 - 100
af در لایه های مخفی	RELU (SELU)
Loss Function	MSE or MAE , Huber



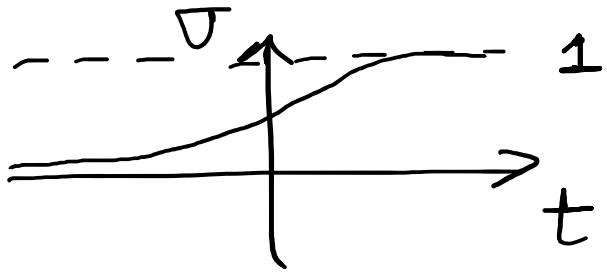


طبقه بندی (Classification MLPs)

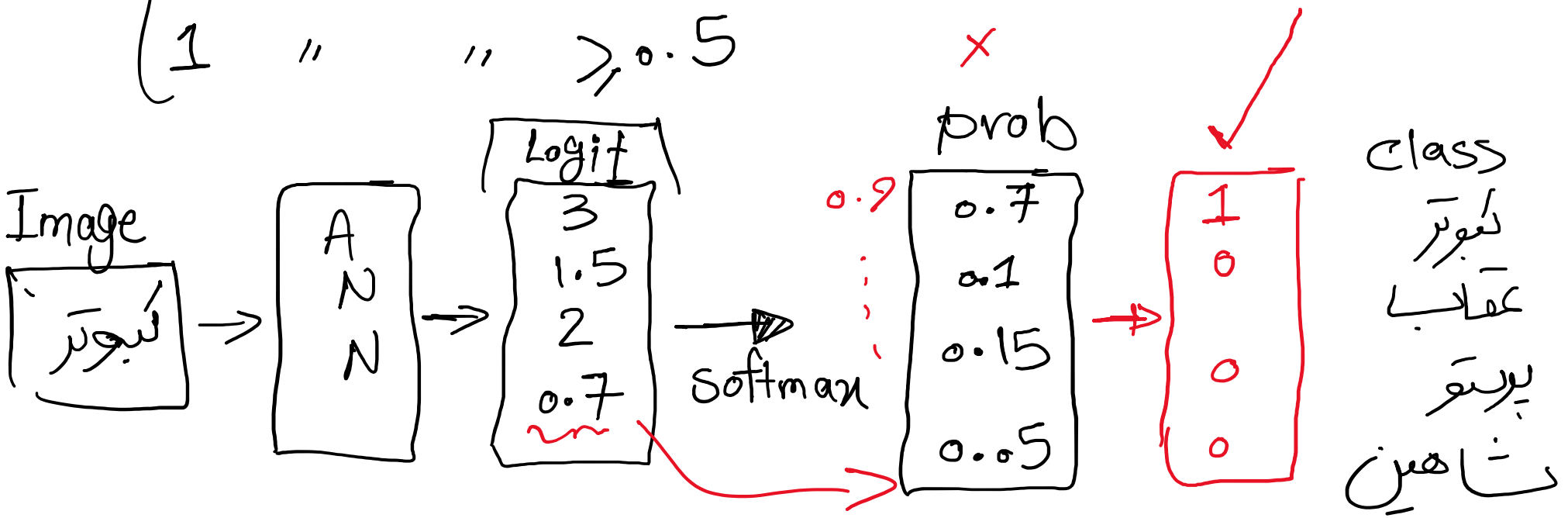
Hyperparameters	binary classification	Multilabel binary classification	Multiclass classification
نورون خروجی	1	1 برای هر label	1 برای هر کلاس
تابع فعال‌سازی خروجی af	Logistic	Logistic	Softmax
Loss function	Cross Entropy	Cross Entropy	Cross Entropy

Loss (classification)

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



$$\hat{y} = \begin{cases} 0 & \text{if } \sigma(t) < 0.5 \\ 1 & \text{" " } \geq 0.5 \end{cases}$$



Softmax

$$S(t_i) = \frac{e^{t_i}}{\sum_{i=1}^{n} e^{t_i}}$$

Ex: $S(3) = \frac{e^{3}}{e^{3} + e^{1.5} + e^{2} + e^{0.7}} \rightarrow 0.7$

$S(1.5) \rightarrow 0.5$

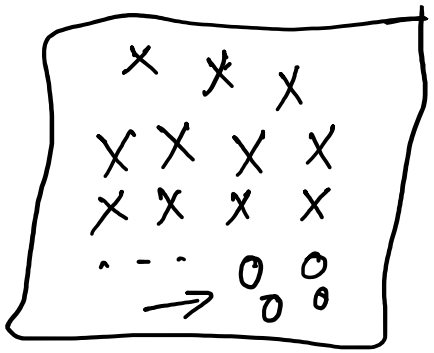
$S(2) \rightarrow 0.15$

$S(0.7) \rightarrow 0.05$

Cross - Entropy

Entropy

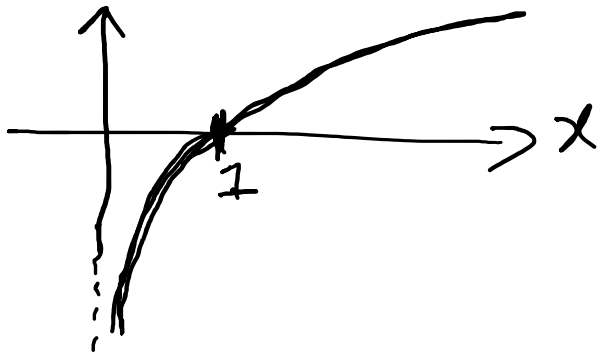
$$H(x) = -\sum_{x} p(x) \log_2 p(x)$$

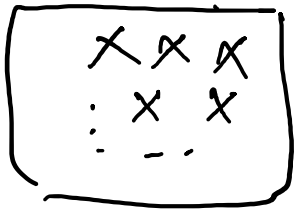


26 x
4 0

$$H(x) = -\left[\frac{26}{30}\log_2\left(\frac{26}{30}\right) + \frac{4}{30}\log_2\left(\frac{4}{30}\right)\right]$$

$$= 0.57$$





30 x
0 0

$$H(x) = -\left[\frac{30}{30}\log_2\left(\frac{30}{30}\right) + 0\right]$$

$$= 0$$

Cross-Entropy Loss

$$L_{CE} = -\sum_{i=1}^{n} t_i \log(p_i)$$

n class

t_i ← واقعی, Label

p_i ← softmax

$$L_{CE} = -\sum_{i=1}^{n} t_i \log(p_i) = -\left[t \log(p) + (1-t)\log(1-p)\right]$$

prediction

True Label

① $[0.9, 0.07, 0.015, 0.015]$, $[1, 0, 0, 0]$

② $[0.6, 0.1, 0.2, 0.1]$ > $[1, 0, 0, 0]$

---

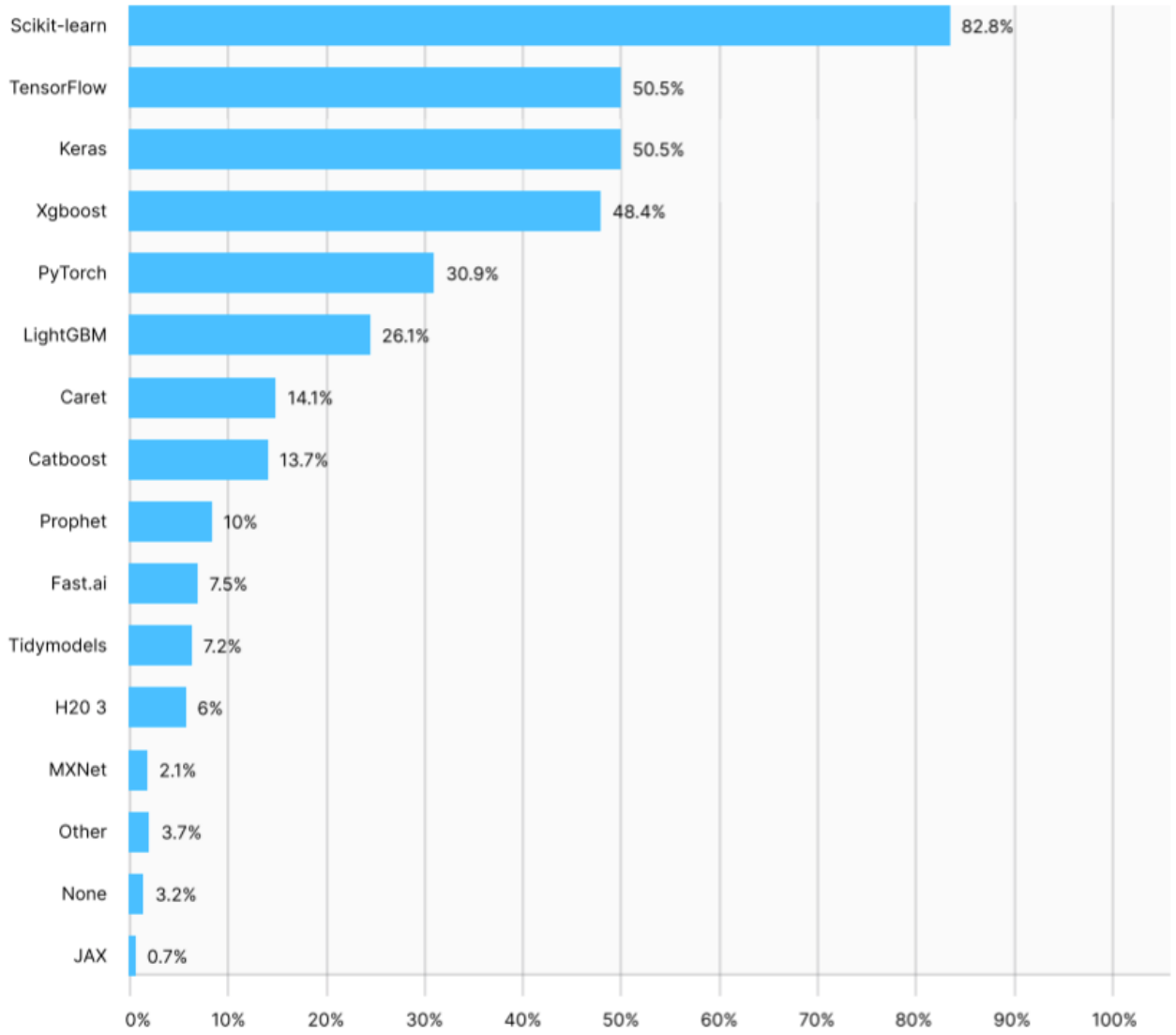
① $L_{CE} = -\sum_{i=1}^{n} t_i \log(S_i) = -[1 \times \log_2(0.9) + 0] = 0.15$

② $L_{CE} = -[1 \times \log_2(0.6) + 0 \times \log_2(0.1) + 0] = 0.7$

Categorical Cross-Entropy → Label one-hot [1, 0, 0, 0] ... [0, 0, 0, 1]

Sparse categorical Cross-Entropy → Label integer encoded [0] ... [3]

فریمورک های یادگیری ماشین



https://www.kaggle.com/kaggle-survey-2020

Keras → March 2015

https://keras.io/

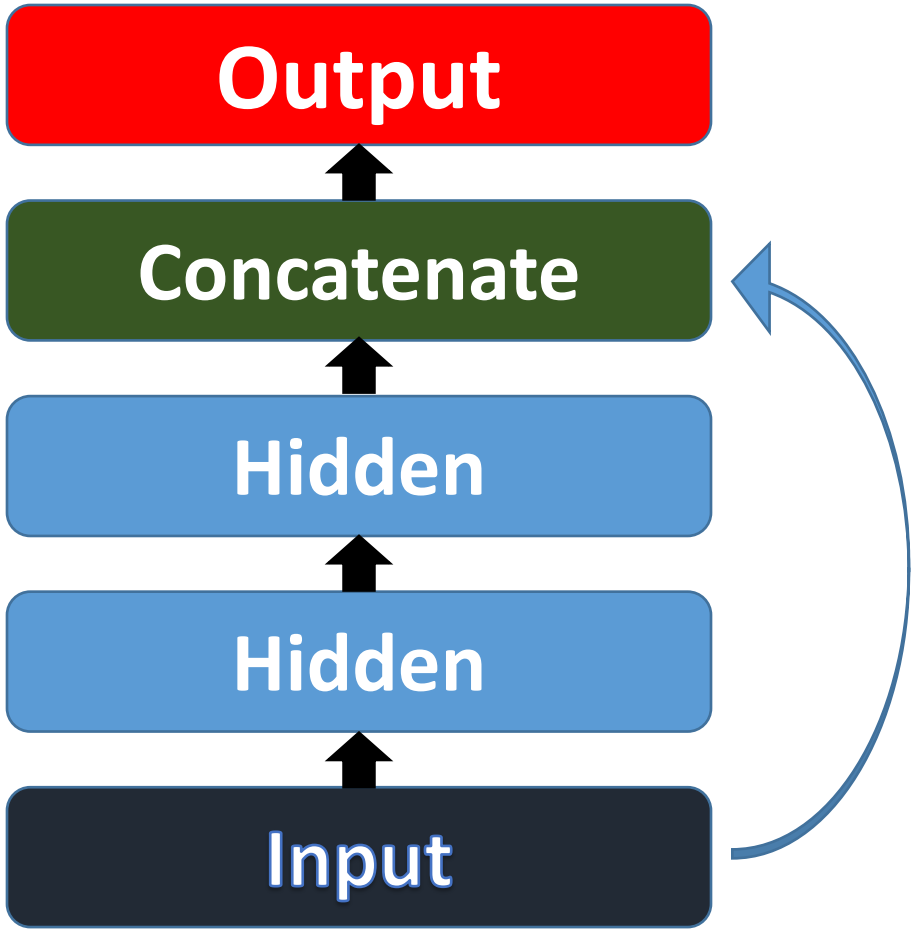
"ONEIROS"

- Tensorflow
- CNTK
- Theano

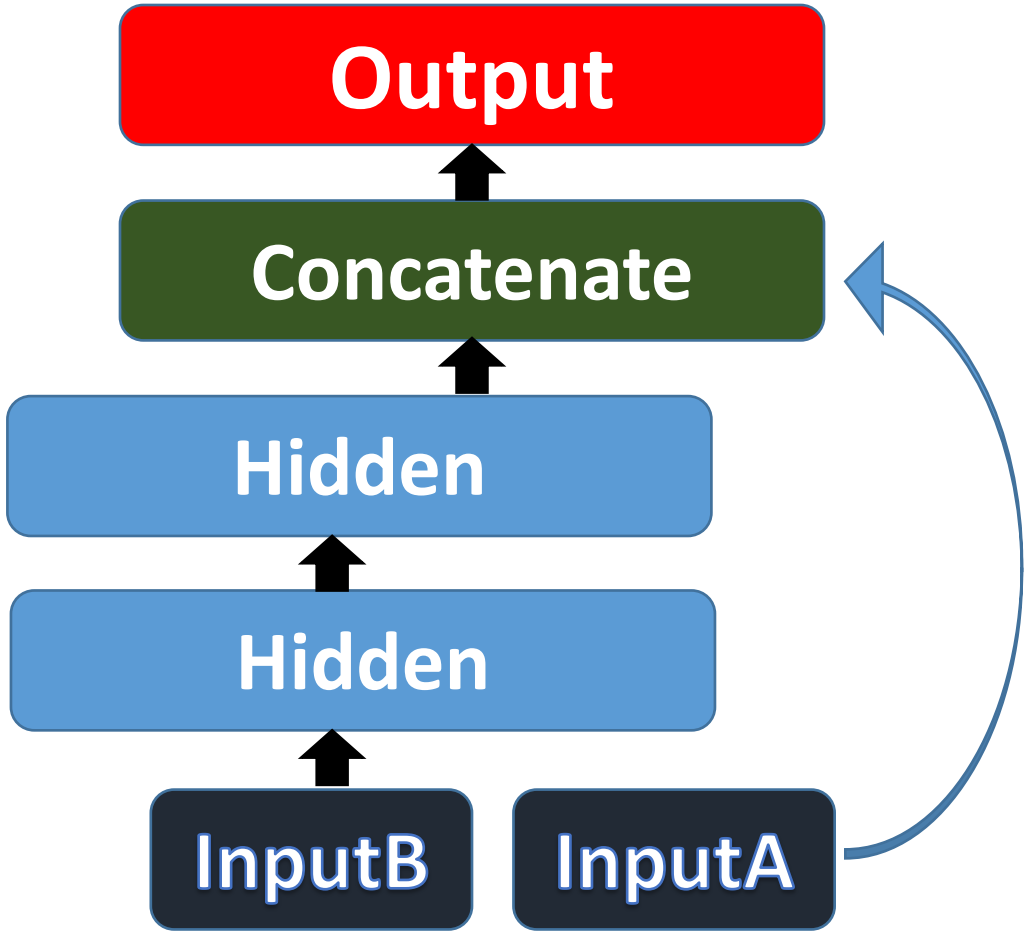
tf.keras

Wide and Deep Network

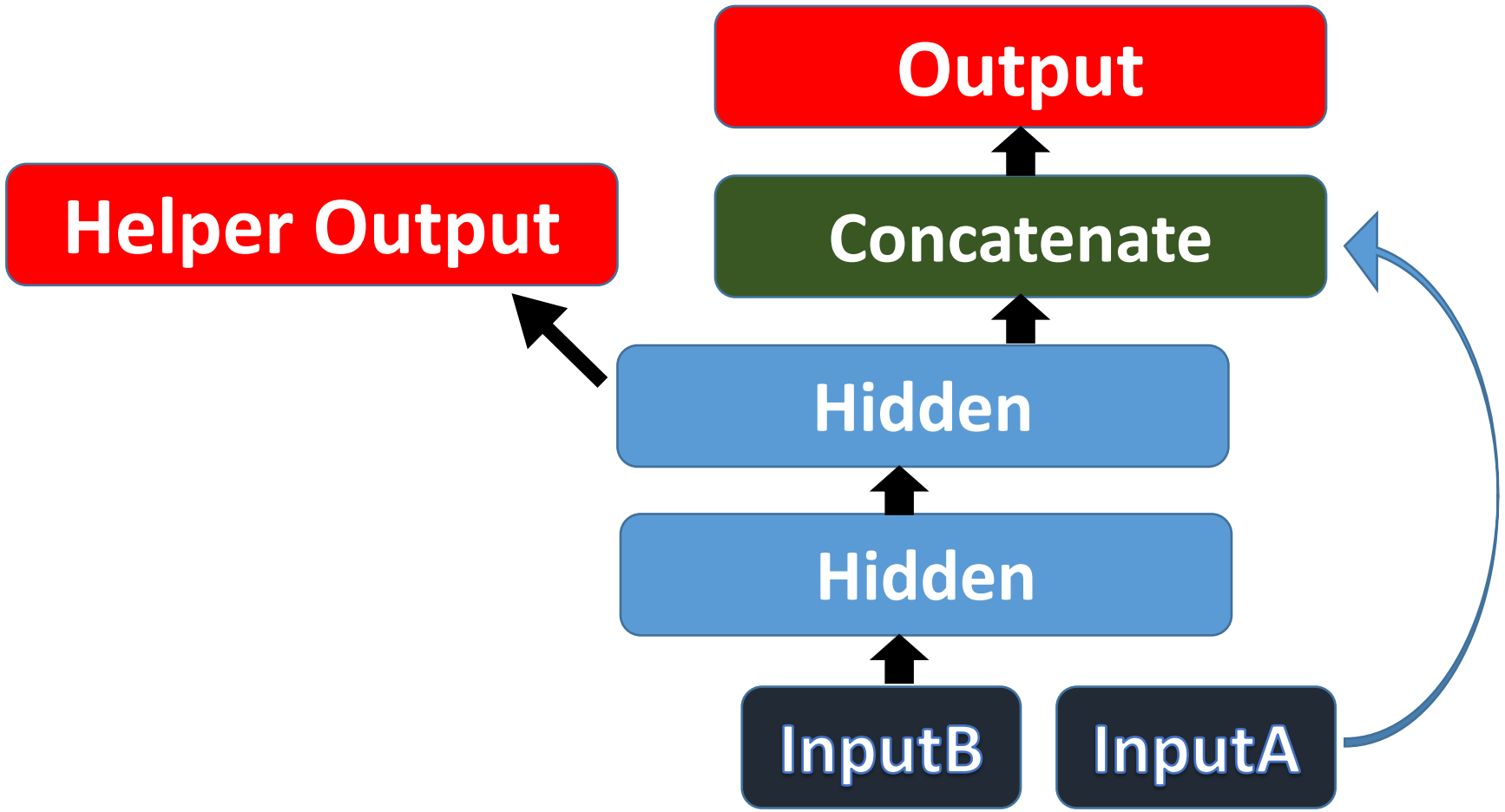
“Wide and Deep Learning for Recommender Systems” Heng-Tze Cheng et al.



Wide and Deep Network



Wide and Deep Network



- **Number of hidden layers**
- **Number of neurons per hidden layer**
- **Learning rate**
- **Optimizer**
- **Batch size**
- **Activation function**

A disciplined approach to neural network hyperparameters
L. N. Smith 2018
arXiv 1803.09820