Artificial Neural Networks (#ANN)

Elias Mohajeri (MohajeriE@yahoo.com)

Aerospace academy 2023

هوش مصنوعی (AI)

↓

شبکه‌های عصبی مصنوعی

ANN

→
- unsupervised
- supervised
- Deep learning

ANN

↓

یادگیری نظارت شده

↓

تخمین تابع

supervised →
- Control
- Identification
- function approximation

Regression ← function approximation

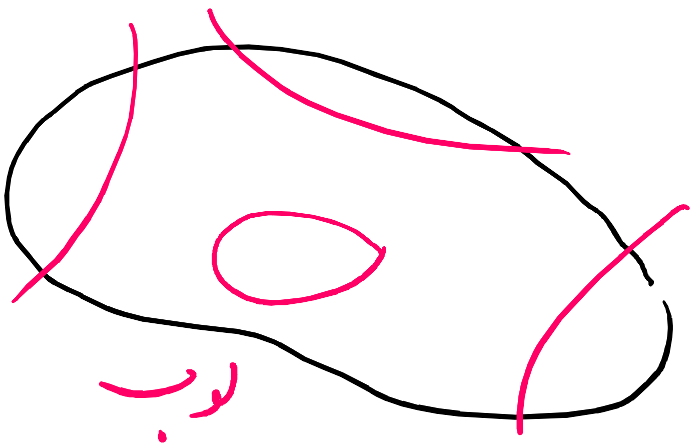
Supervised Learning → { feed-forward → SLP, MLP ✓ ; Radial → RBF ✓ }

(Regression)

شبکه عصبی مصنوعی

Brain → electronic model

training



لوب

مدل‌سازی ⇓

ANN

تجربه

experience

Som

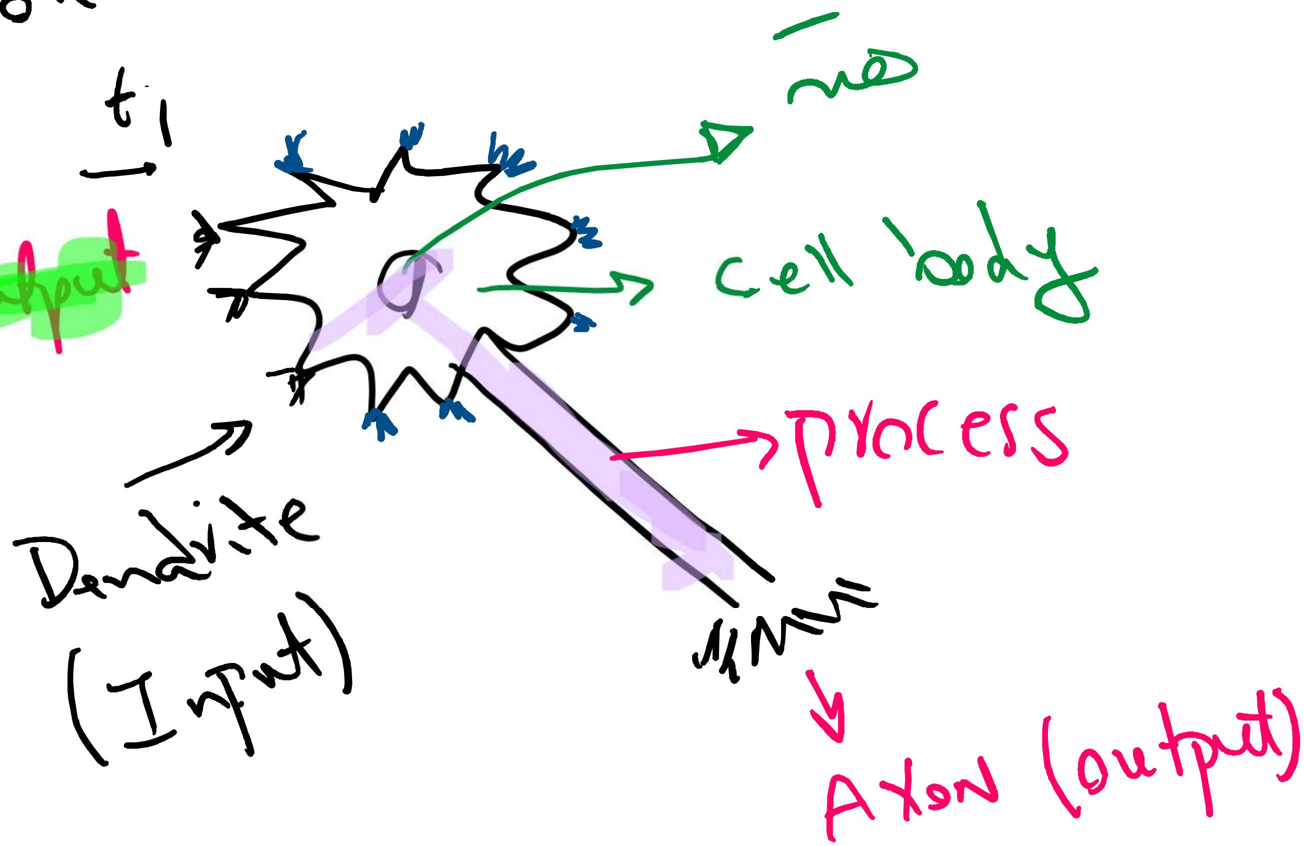
Neuron

t_0 → t_1 →

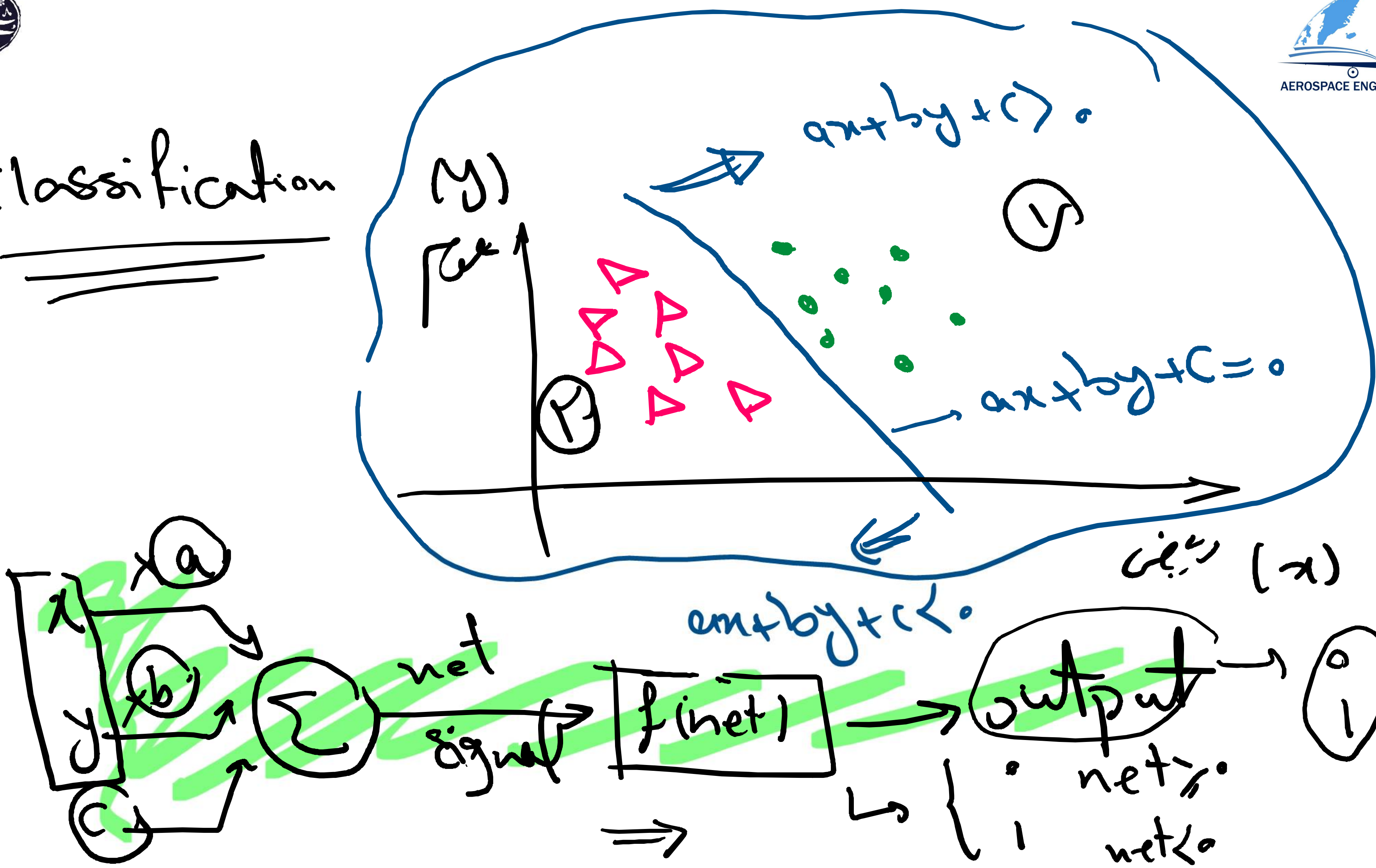
input → process → output

↓

Nonlinear operation



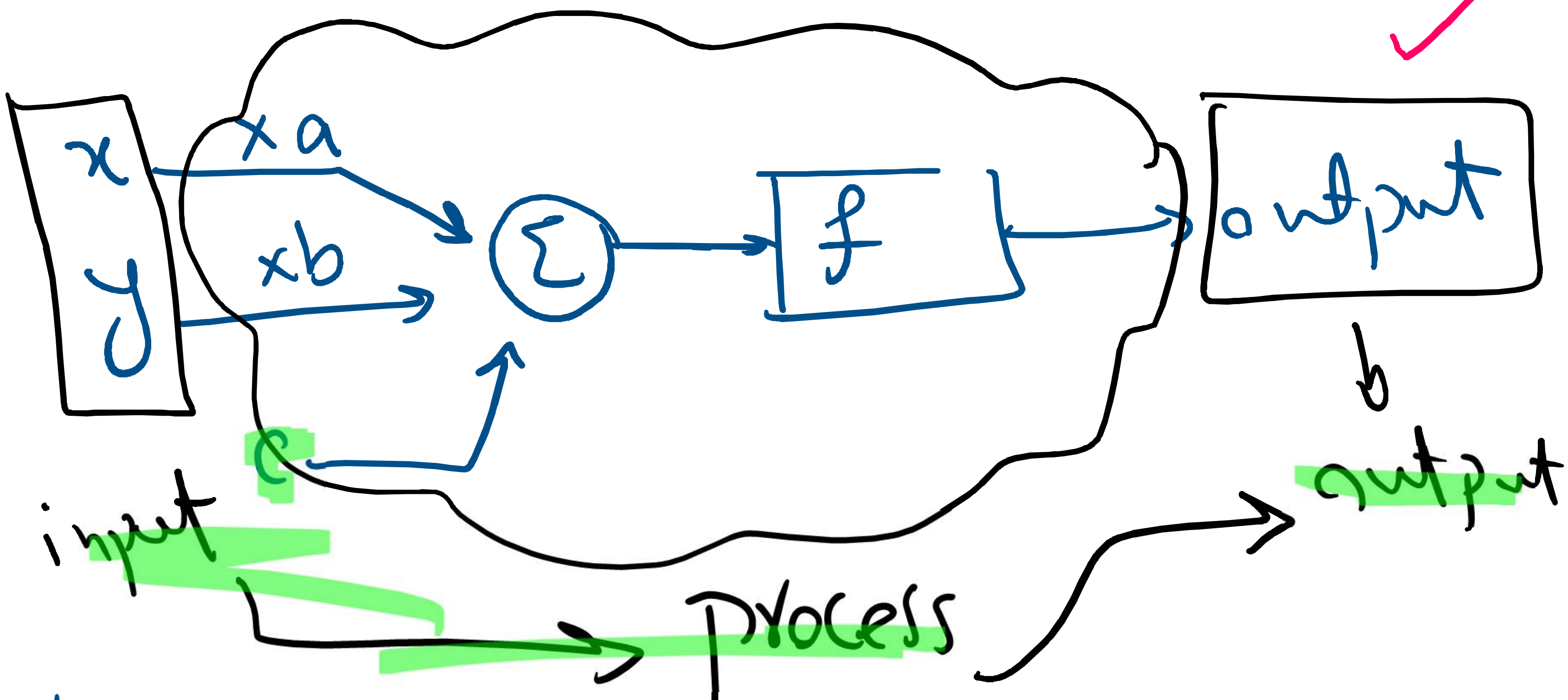
Classification



input → process → output

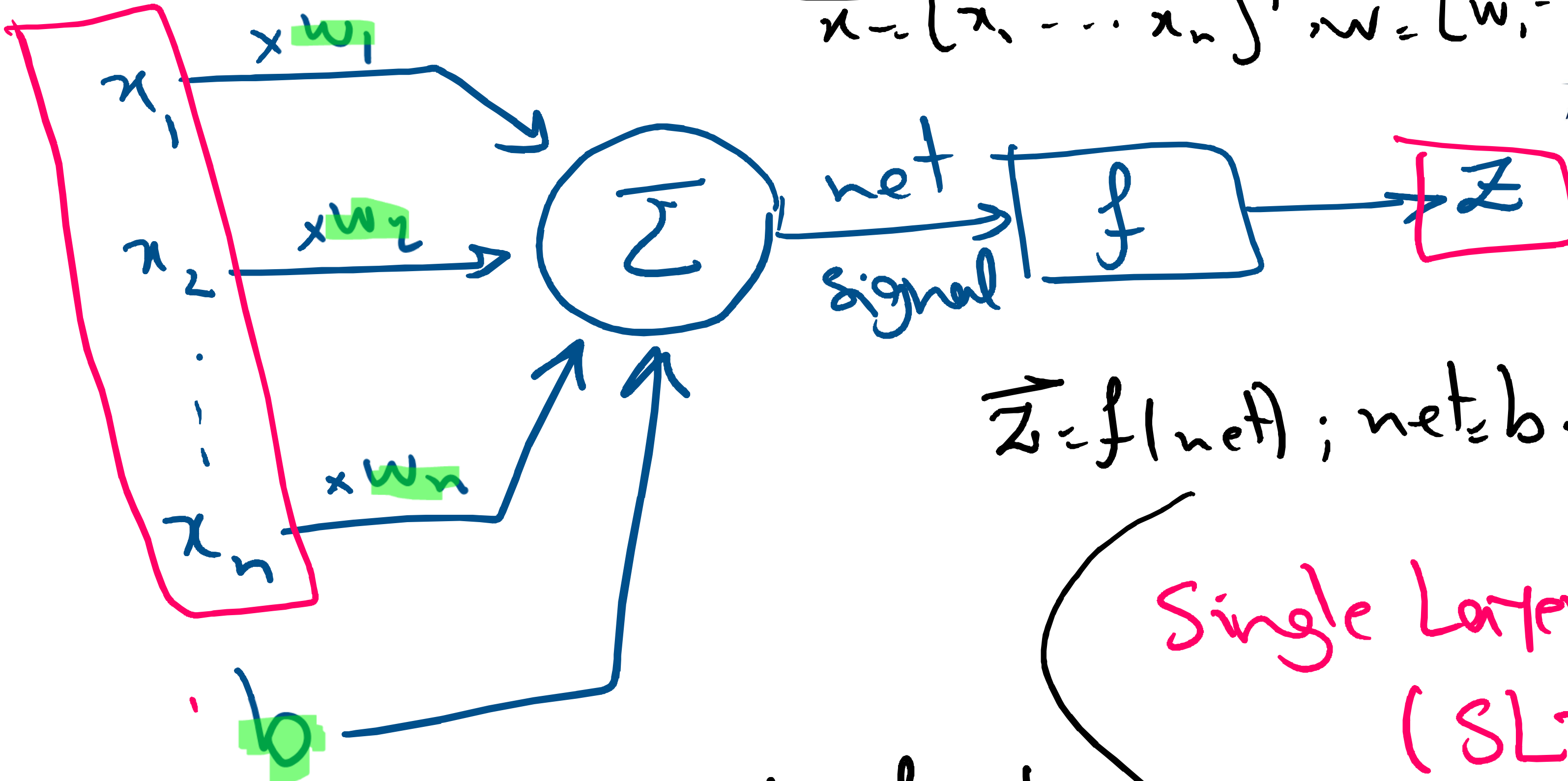
مدل ریاضی

نورون ساده



A simple Neuron

$\vec{x} = [x_1 \cdots x_n]^T$, $\vec{w} = [w_1 \cdots w_n]^T$



$$\vec{z} = f(net) \; ; \; net = b + \sum_{i=1}^{n} x_i w_i$$

Single Layer Perceptron (SLP)

SLP $$\vec{z} = f(\vec{w}^T \vec{x} + b)$$

$x_i \triangleq$ input

$w_i \triangleq$ weights

$b \triangleq$ bias

f : Activation function

z : output

input ⟶ process ⟶ output

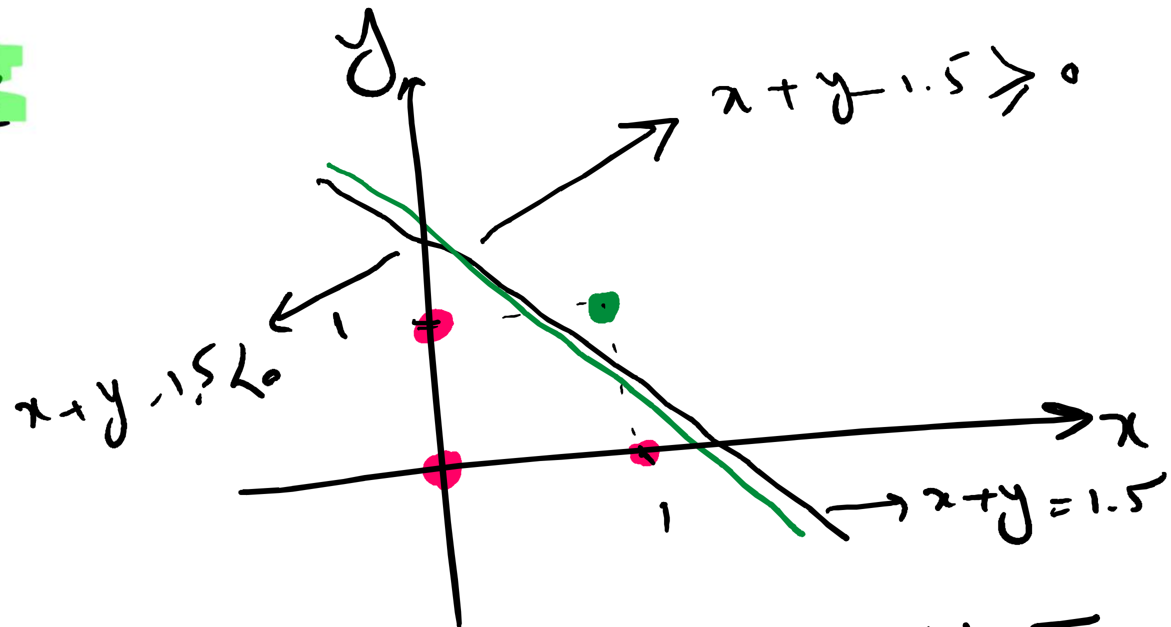
⇓

$$\vec{z} = f(\vec{w}^T \vec{x} + b)$$

$\vec{z}$, output ⟶ error

SLP example:

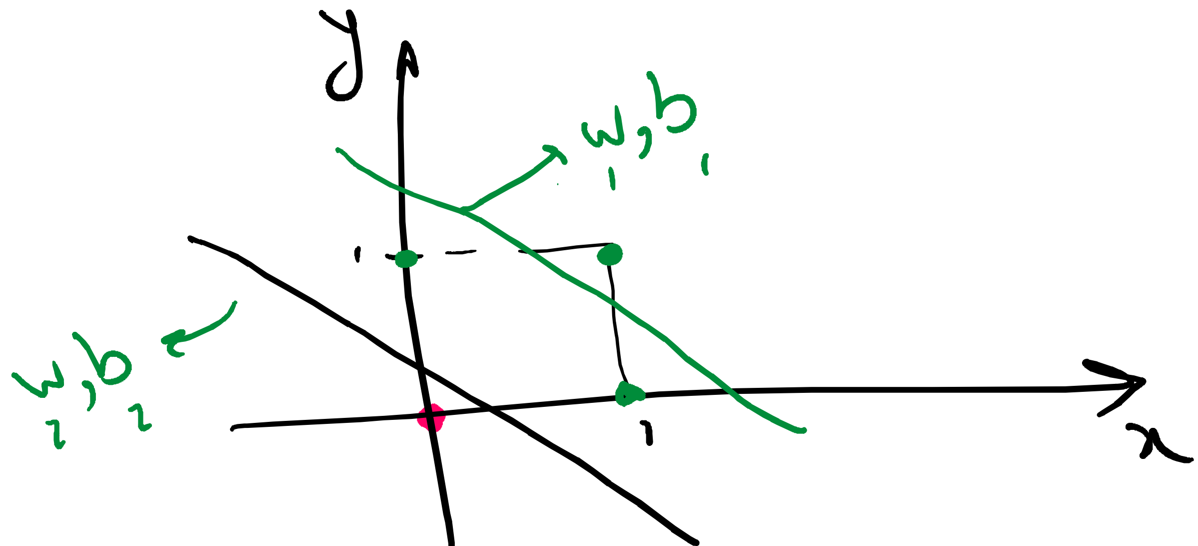
x	y	x & y
0	0	0
0	1	0
1	0	0
1	1	1



SLP → line equation

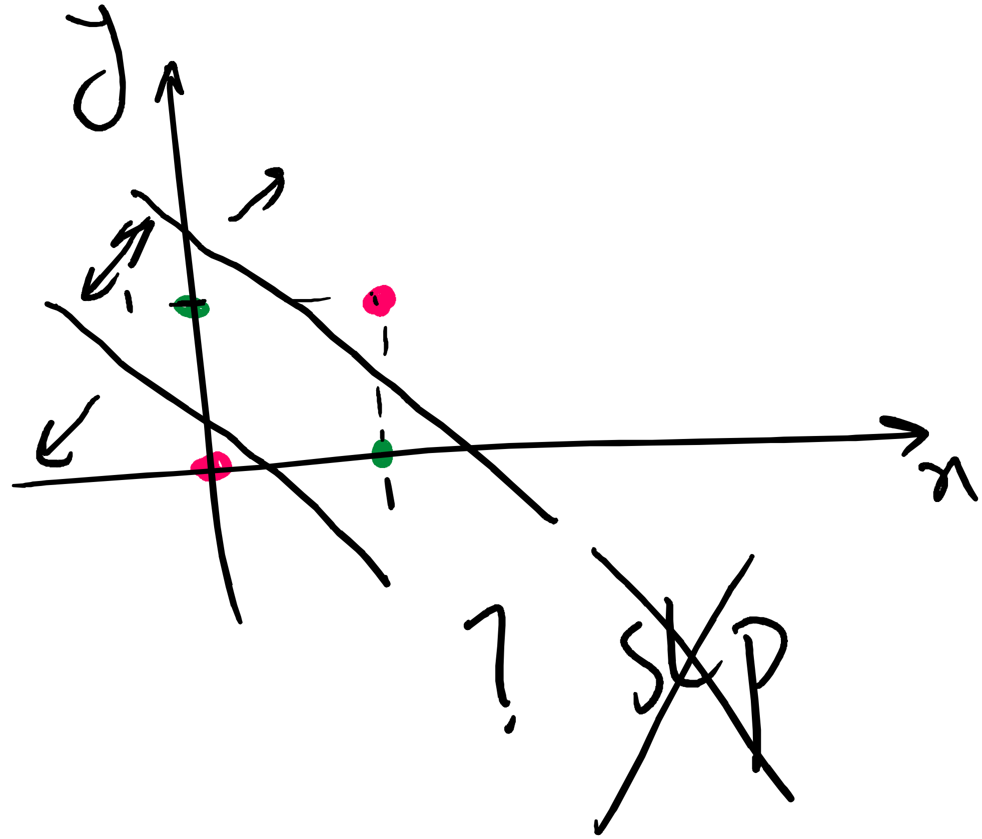
$$x+y=1.5 \rightarrow z = x \& y = \begin{cases} 0 ; & x+y \leq 1.5 \\ 1 ; & x+y > 1.5 \end{cases}$$

x	y	x‖y
0	0	0
0	1	1
1	0	1
1	1	1



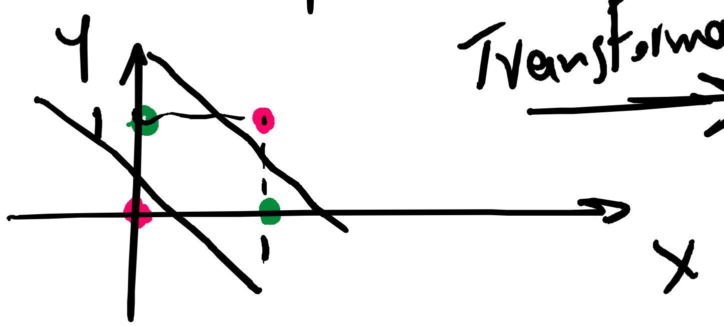
$x+y=0.5$ → $z = x \| y \begin{cases} 0 & ; \; x+y<0.5 \\ 1 & \; x+y>0.5 \end{cases}$

x	y	x XoR y
0	0	0
1	0	1
0	1	1
1	1	0

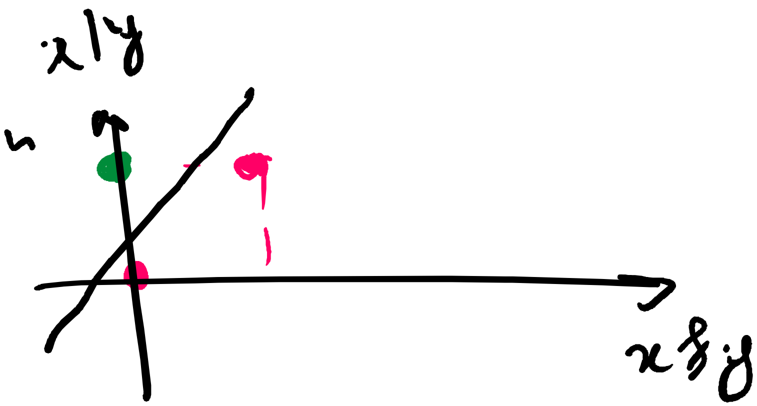


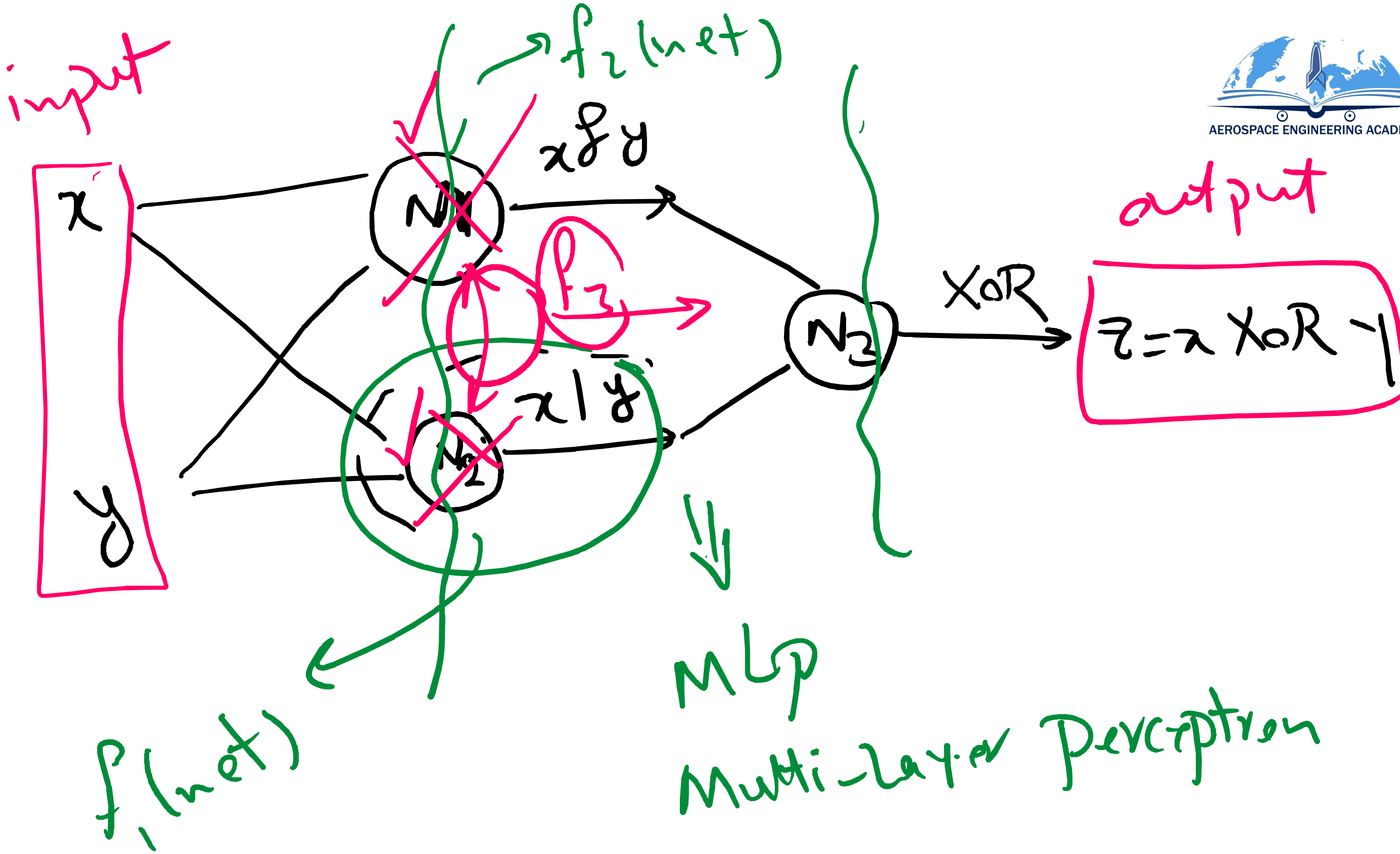
x	y	x XOR y	x & y	x | y
0	0	0	0	0
1	0	1	0	1
0	1	1	0	1
1	1	0	1	1

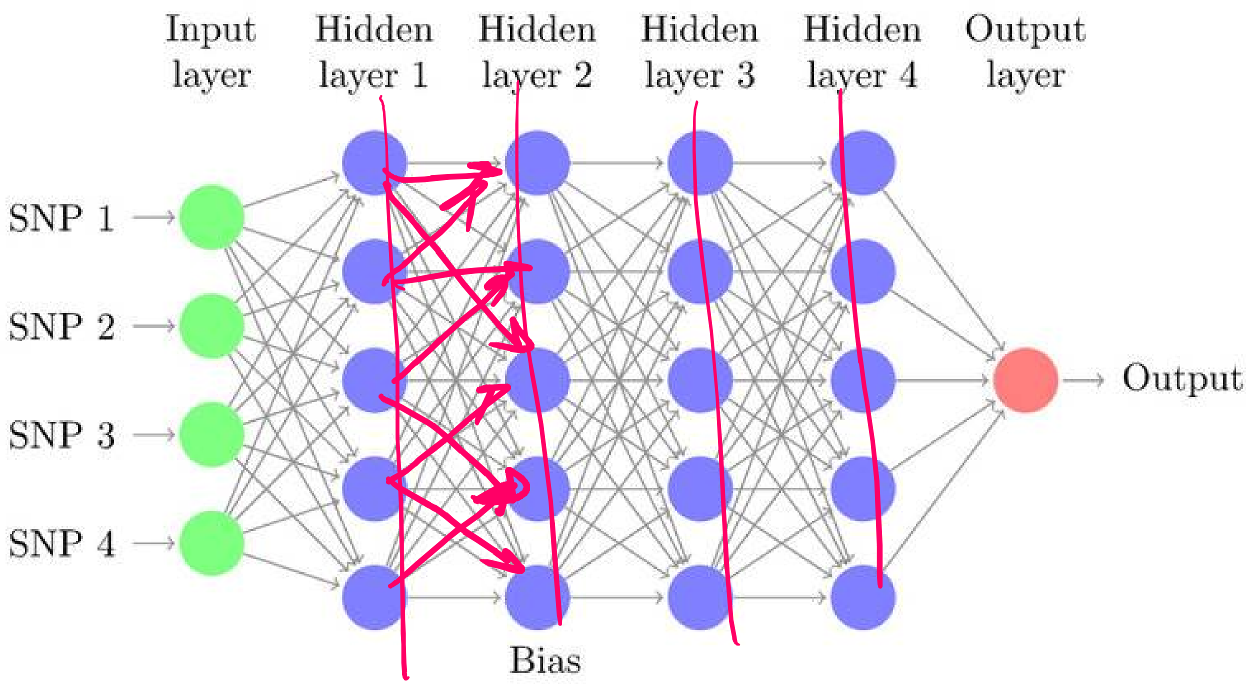
SLP ~~XOR~~

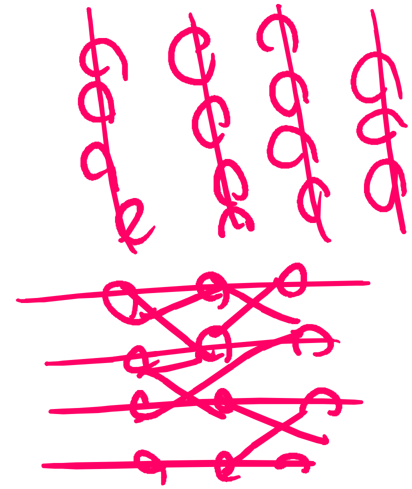


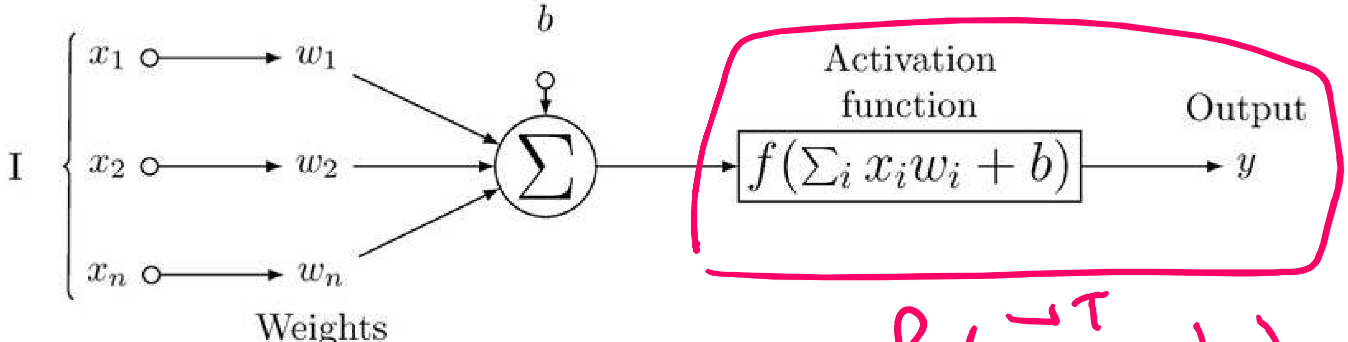
Transformation →




input
x
y
N1
N2
N3
x&y
x|y
f2(net)
f3
f1(net)
XoR
output
z = x XoR y
MLp
Multi-Layer Perceptron


Input layer
Hidden layer 1
Hidden layer 2
Hidden layer 3
Hidden layer 4
Output layer
SNP 1
SNP 2
SNP 3
SNP 4
Output



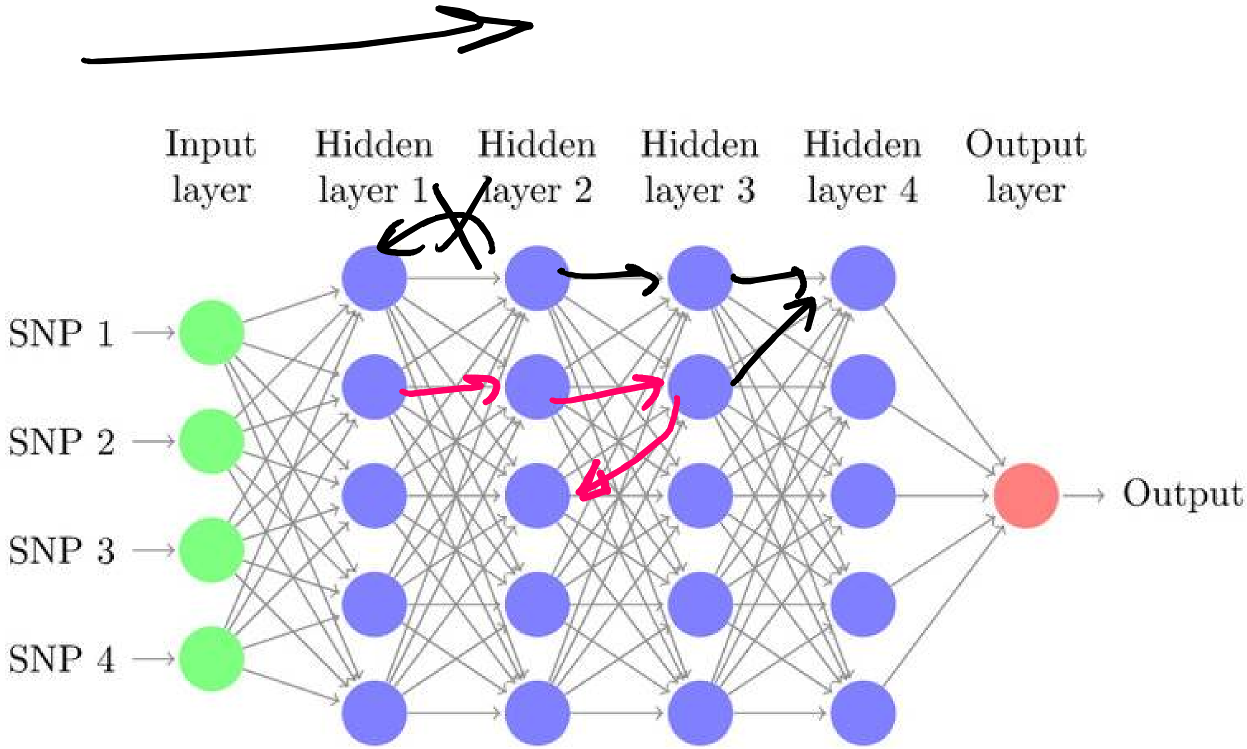

Bias
b
I
x_1
x_2
x_n
w_1
w_2
w_n
Weights
$\sum$
Activation function
$f(\sum_i x_i w_i + b)$
Output
y

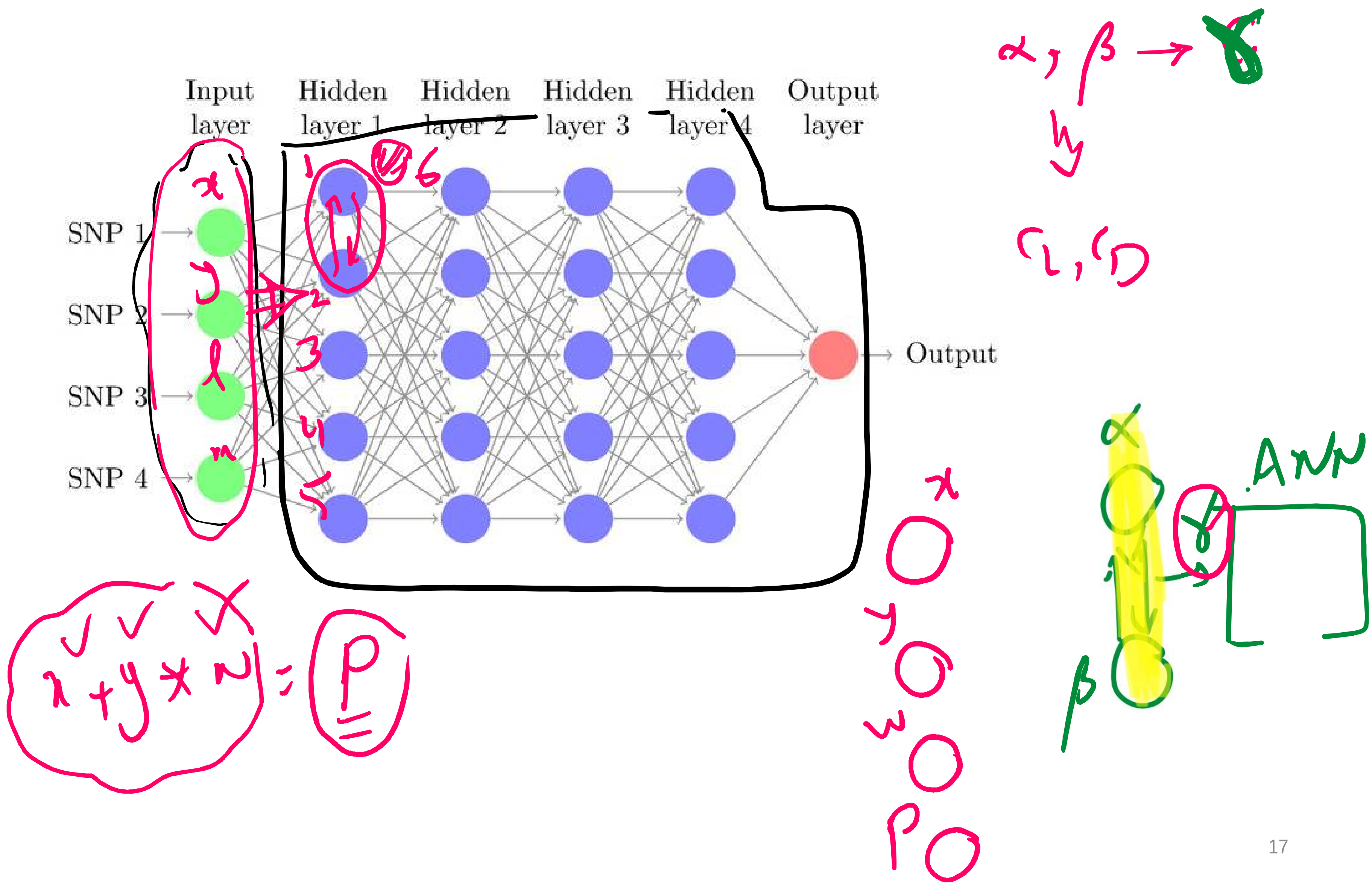
$$z = f(\vec{w}^T x + b)$$

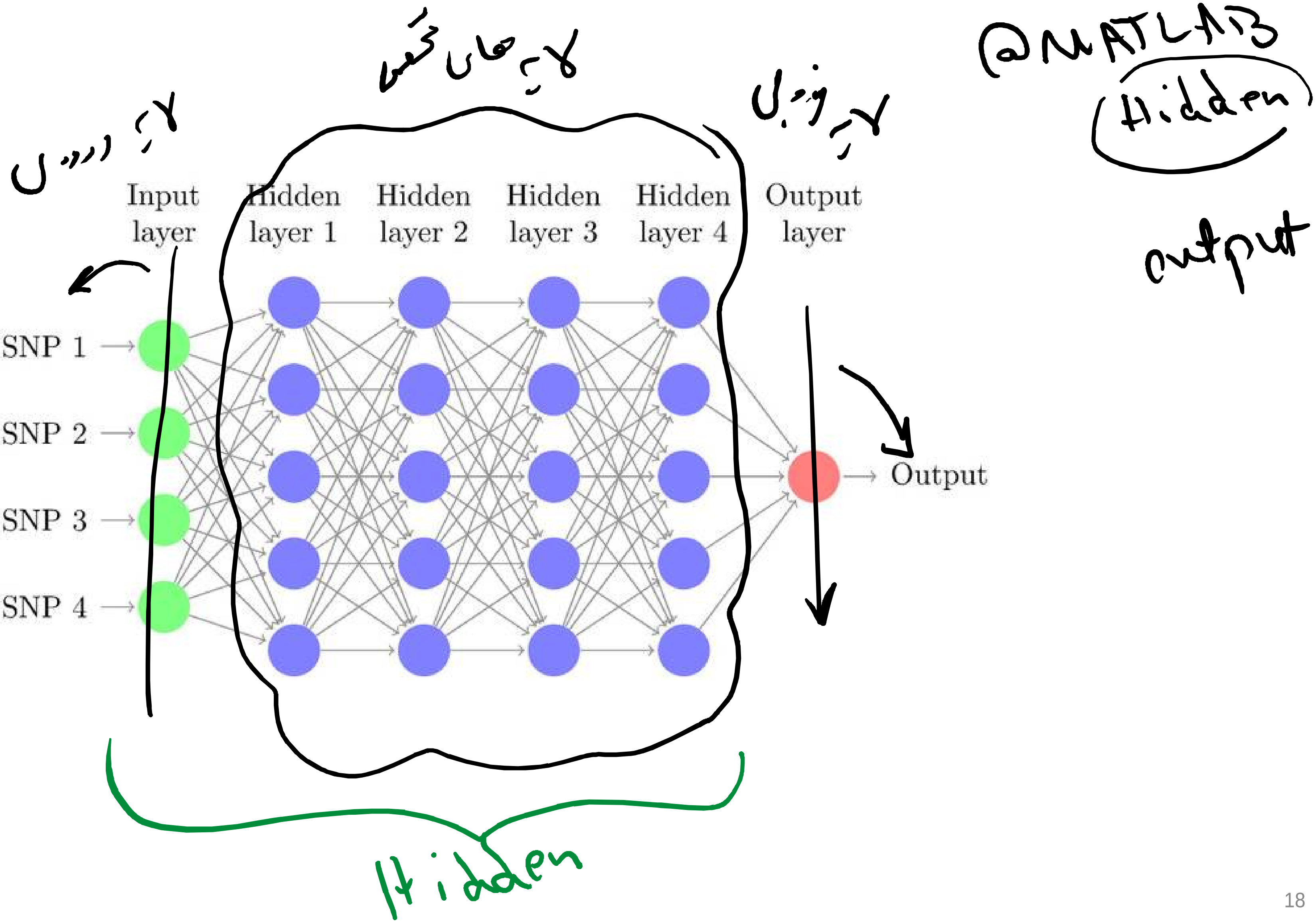
feed-forward networks → information moves one direction

⇓

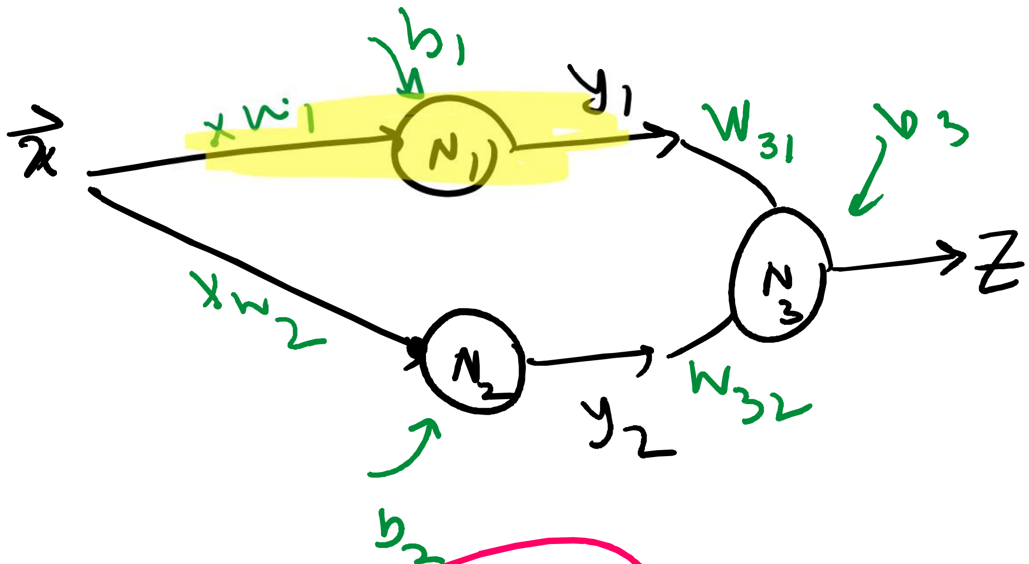
It never goes backwards




Input layer
Hidden layer 1
Hidden layer 2
Hidden layer 3
Hidden layer 4
Output layer
SNP 1
SNP 2
SNP 3
SNP 4
Output



لایه ورودی
لایه های مخفی
لایه خروجی
@MATLAB
Hidden
output
Input layer
Hidden layer 1
Hidden layer 2
Hidden layer 3
Hidden layer 4
Output layer
SNP 1
SNP 2
SNP 3
SNP 4
Output
Hidden



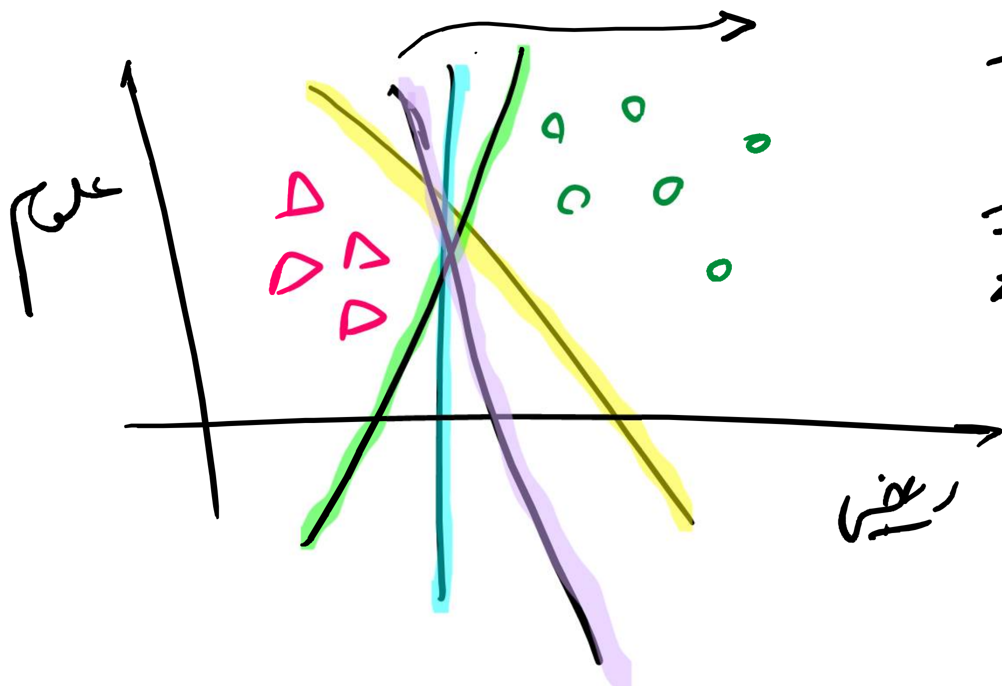
$$\vec{y} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

$$W_3 = \begin{bmatrix} W_{31} \\ W_{32} \end{bmatrix}$$

$$y_1 = f_1(\vec{w}_1^T \vec{x} + b_1) \quad ; \quad y_2 = f_2(\vec{w}_2^T \vec{x} + b_2)$$

$$\vec{z} = f_3(w_3^T \vec{y} + b_3) = f_3(w_{31} y_1 + w_{32} y_2 + b_3)$$

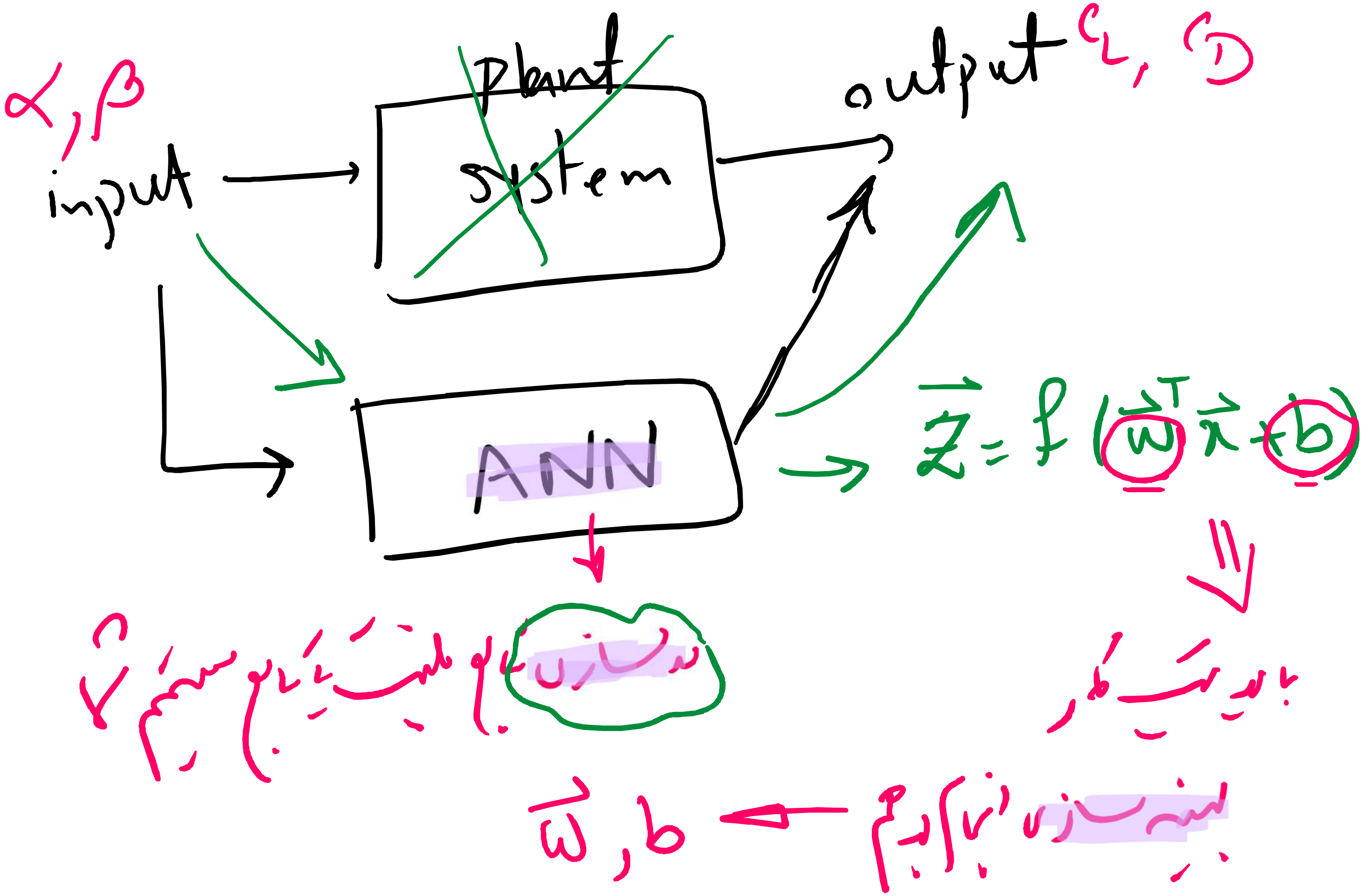
$\to z_3 = f_3(*) \Rightarrow$ MLP is a function



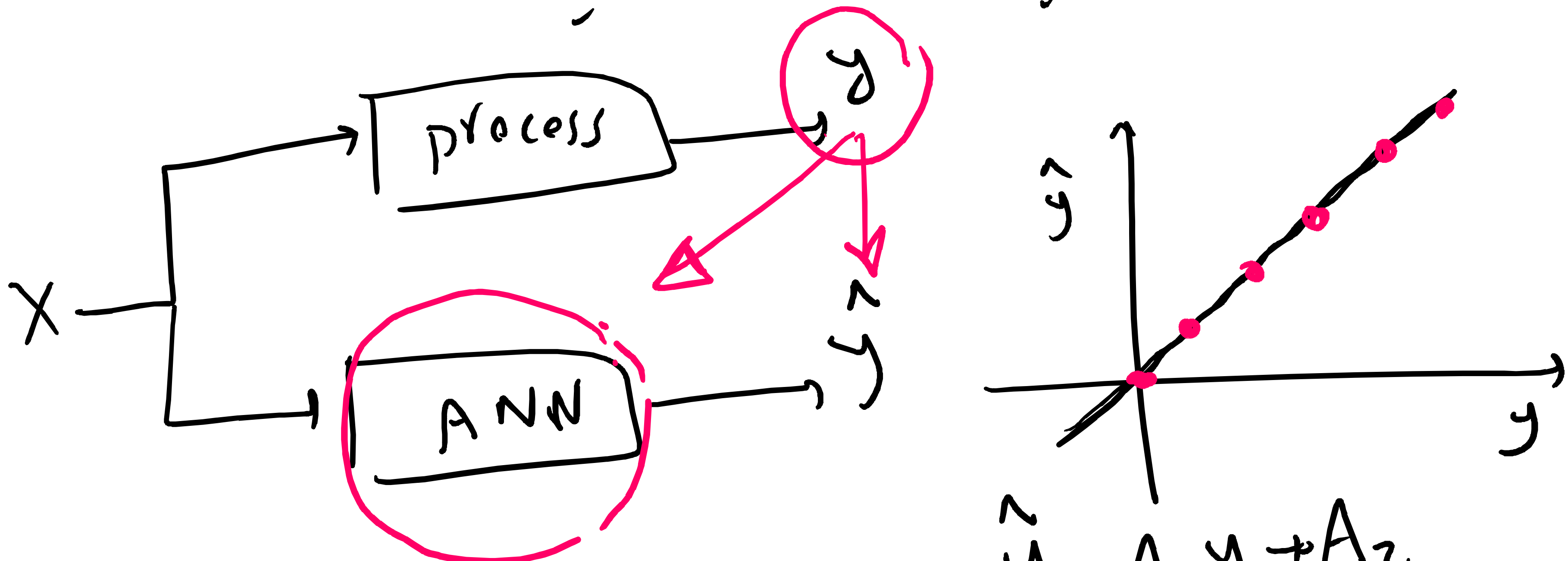
$$\vec{z} = f(\vec{w}^T x + b)$$
$$\vec{z} = f(\vec{w}^T x + b)$$
$$\vec{z} = f(\vec{w}^T x + b)$$
$$\Rightarrow \vec{z} = f(\vec{w}^T x + b)$$

Training ⇐ b, $\vec{w}$ پروسه انتخاب درستِ ؟

Decision variables
$\vec{w}$, b پارامترهای تصمیم ⇐ optimization بهینه سازی

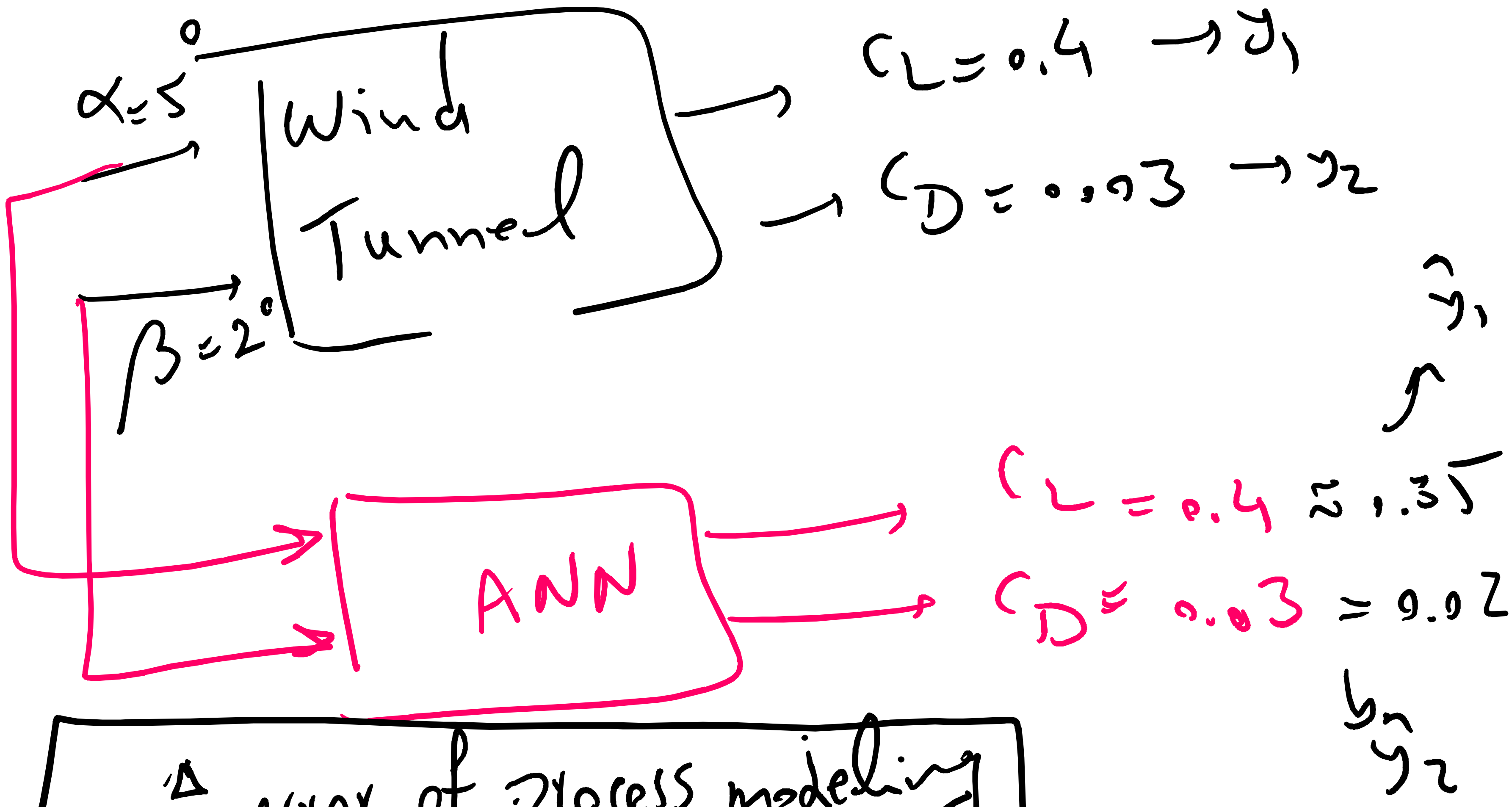

α, β
input
plant
system
output
C_L, C_D
ANN
$\vec{z} = f(\vec{w}^T \vec{x} + b)$
مدلسازی
باید یک کار
بهینه سازی پارامترها
$\vec{w}$, b

$\vec{w}, b$ ← متغیرهای بهینه‌سازی ← برای برازش/آموزش مدل شبکه عصبی



$$\hat{y} = A_1 y + A_2$$

$$\boxed{\hat{y} = y} \leftarrow \text{if } A_1 = 1,\ A_2 = 0$$



$e \triangleq$ error of process modeling

$$\text{ANN} \rightarrow \vec{z} = f(\vec{w}^T x + b) \rightarrow \hat{y}$$

از طریق بهینه‌سازی پیداشان کنید

$$e \downarrow \text{(MSE)}$$

روش‌های بهینه‌سازی

تحلیلی → GD و M

تکاملی/هوشمند → GA, PSO, ACO, ...

$$e = y - \hat{y} = y - f(x \mid \underbrace{w, b}_{\theta})$$

$$e = y - f(x \mid \theta)$$

$$e^2 = (y - f(x \mid \theta))^2$$

$$\Downarrow$$

$$(w^*, b^*)\ \theta^* = \min_{\theta} e^2$$

$\{w, b\}$

$$e = y_i - \hat{y}_i = y_i - f(x_i | \theta)$$

$$\min \sum_{i=1}^{N} \alpha_i e_i^2 \quad ; \; \alpha_i = \frac{1}{N}$$

Mean square error (MSE)

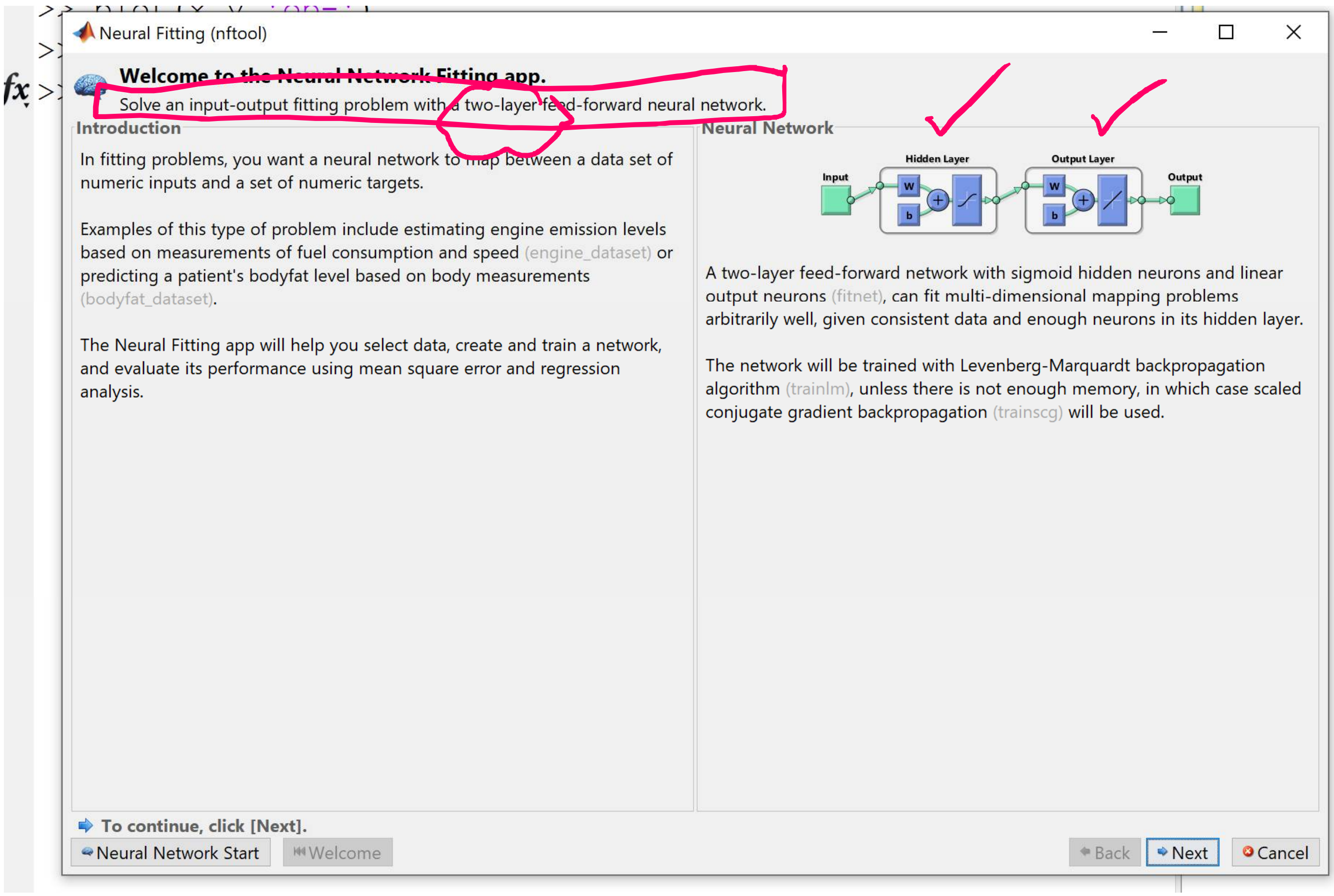
y_1 $\quad$ $\hat{y}_1$

y_2 $\quad$ $\hat{y}_2$

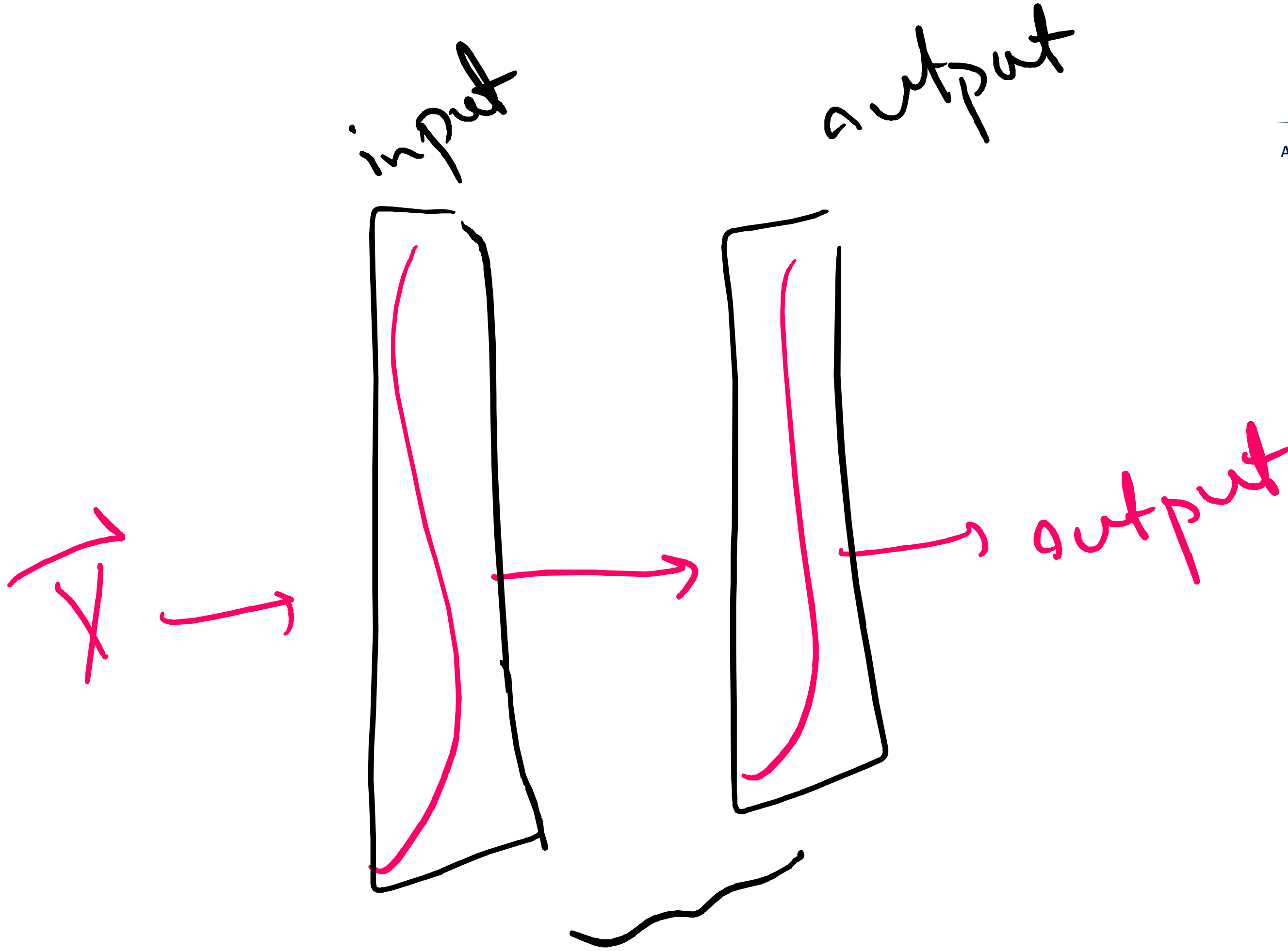
y_3 $\quad$ $\hat{y}_3$ MSE

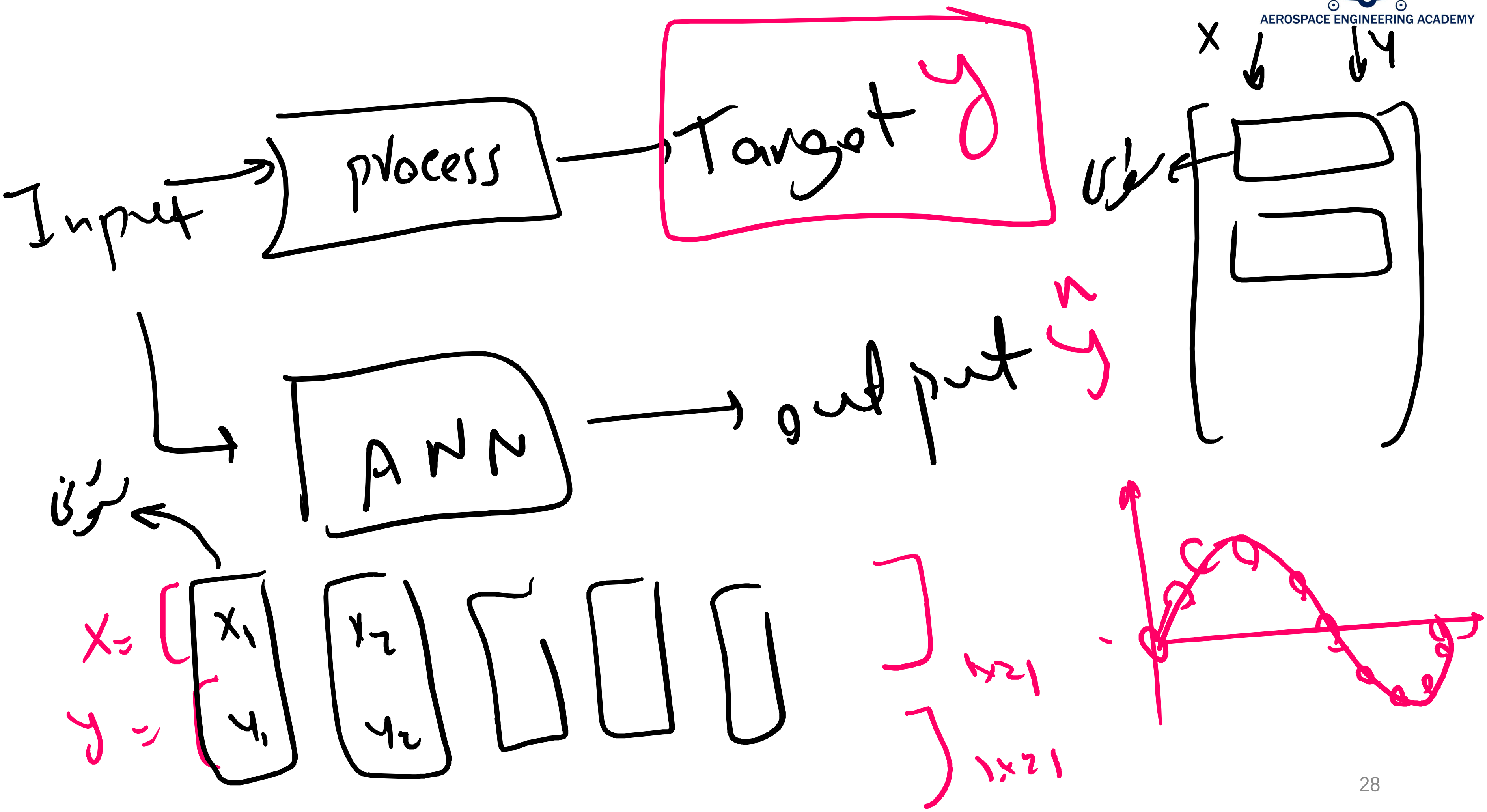
$$= \left[\frac{1}{3} \left((y_1 - \hat{y}_1)^2 + (y_2 - \hat{y}_2)^2 + (y_3 - \hat{y}_3)^2 \right) \right]$$

>>nftool ↵


Neural Fitting (nftool)
Welcome to the Neural Network Fitting app.
Solve an input-output fitting problem with a two-layer feed-forward neural network.
Introduction
In fitting problems, you want a neural network to map between a data set of numeric inputs and a set of numeric targets.
Examples of this type of problem include estimating engine emission levels based on measurements of fuel consumption and speed (engine_dataset) or predicting a patient's bodyfat level based on body measurements (bodyfat_dataset).
The Neural Fitting app will help you select data, create and train a network, and evaluate its performance using mean square error and regression analysis.
Neural Network
Input
Hidden Layer
W
b
Output Layer
W
b
Output
A two-layer feed-forward network with sigmoid hidden neurons and linear output neurons (fitnet), can fit multi-dimensional mapping problems arbitrarily well, given consistent data and enough neurons in its hidden layer.
The network will be trained with Levenberg-Marquardt backpropagation algorithm (trainlm), unless there is not enough memory, in which case scaled conjugate gradient backpropagation (trainscg) will be used.
To continue, click [Next].
Neural Network Start
Welcome
Back
Next
Cancel

nftool




Input
process
Target y
ANN
output ŷ
X
Y
X =
y =
x1
y1
x2
y2
1x21
1x21

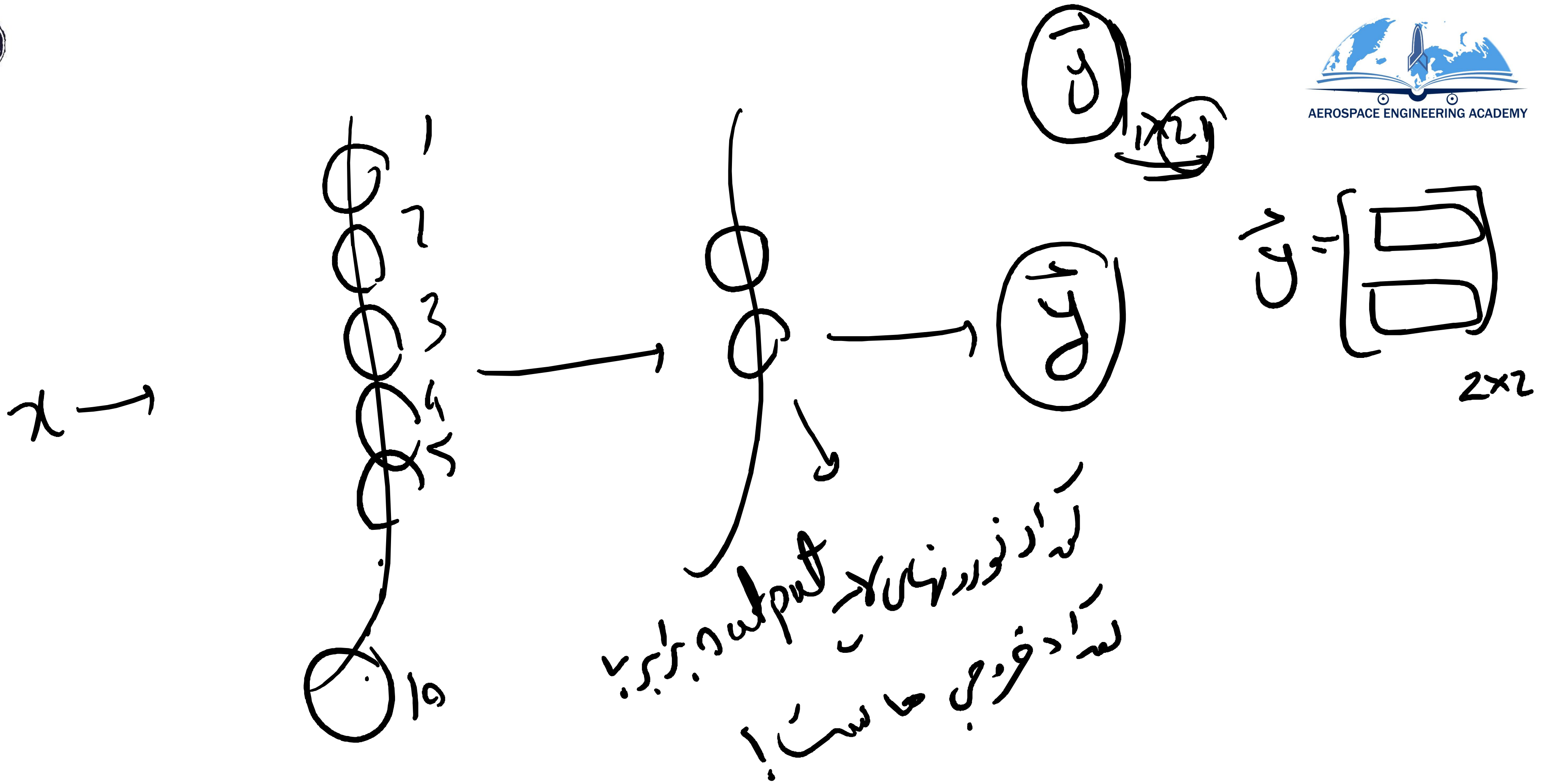
14 sample ← 21 sample

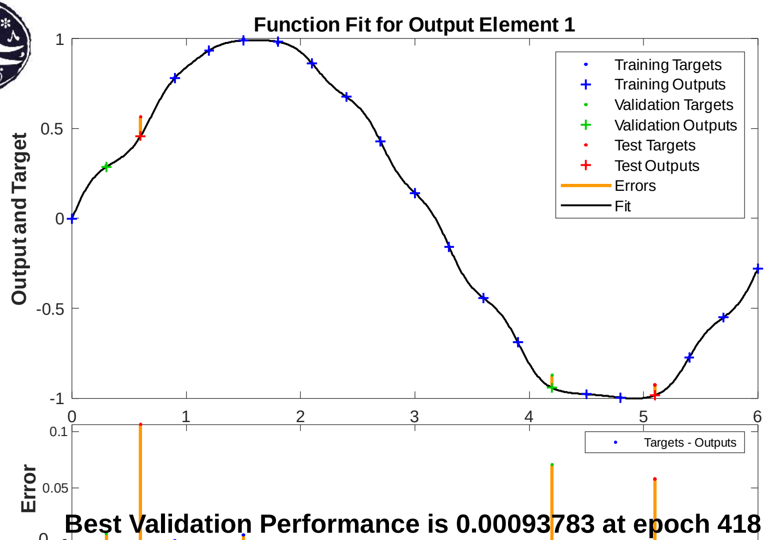
3 kinds of samples:

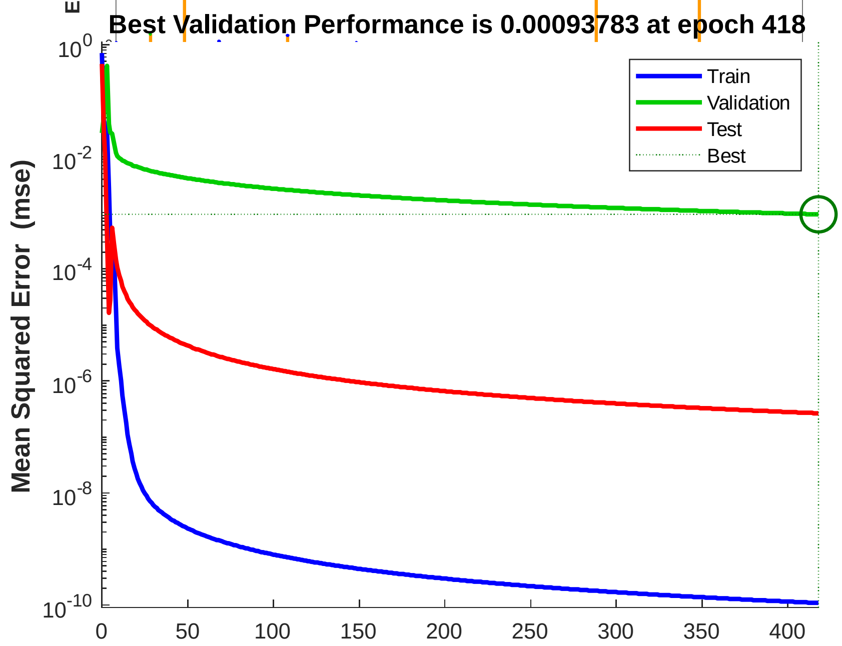
1. Training ✓ داده‌های آموزشی (14)
2. validation داده‌های اعتبارسنجی (4)
3. Testing ✓ داده‌های آزمایش (3) ✓

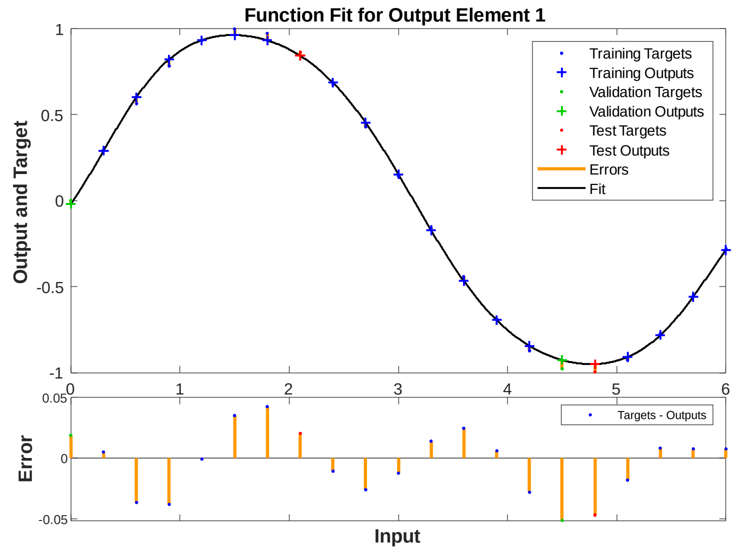
training: تمرین → testing: پایان ترم

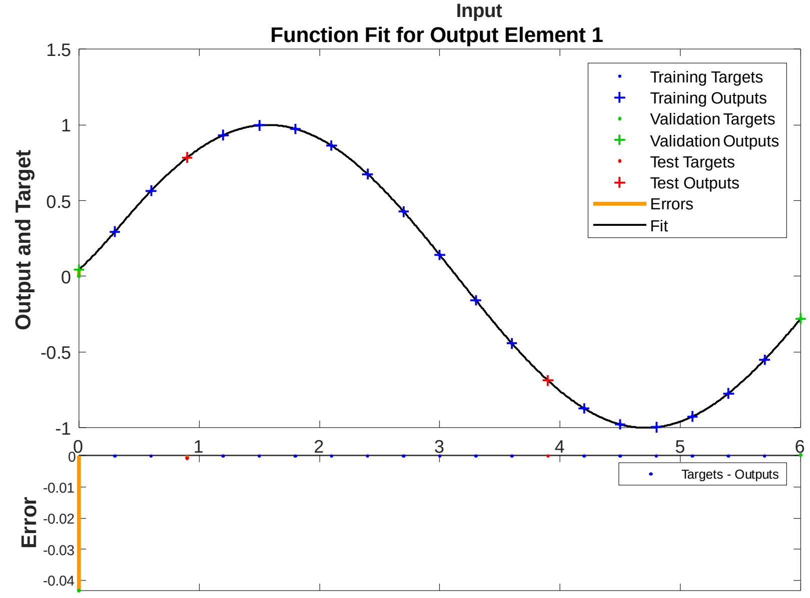
→ Quiz → Validation data

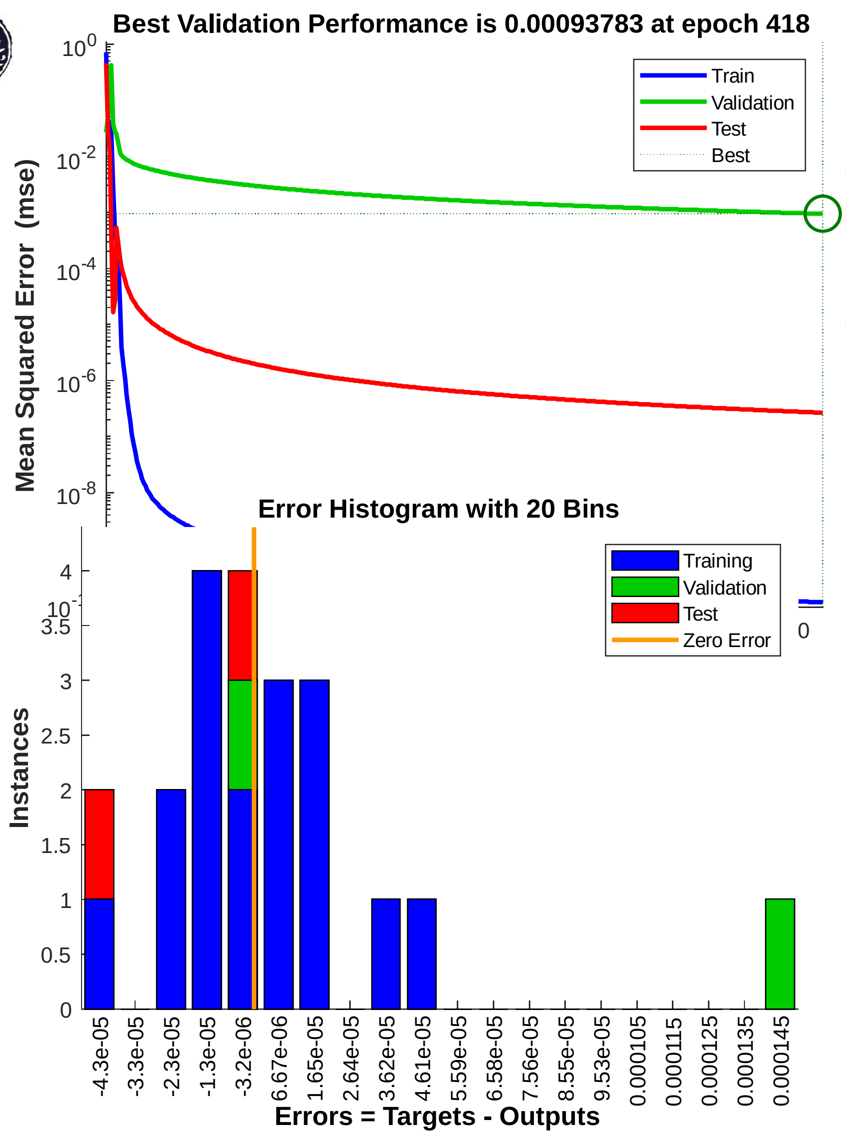

x
1
2
3
4
5
10
ŷ
1×2
ŷ =
2×2
تعداد نورون‌های لایه output برابر با
تعداد خروجی‌ها است!

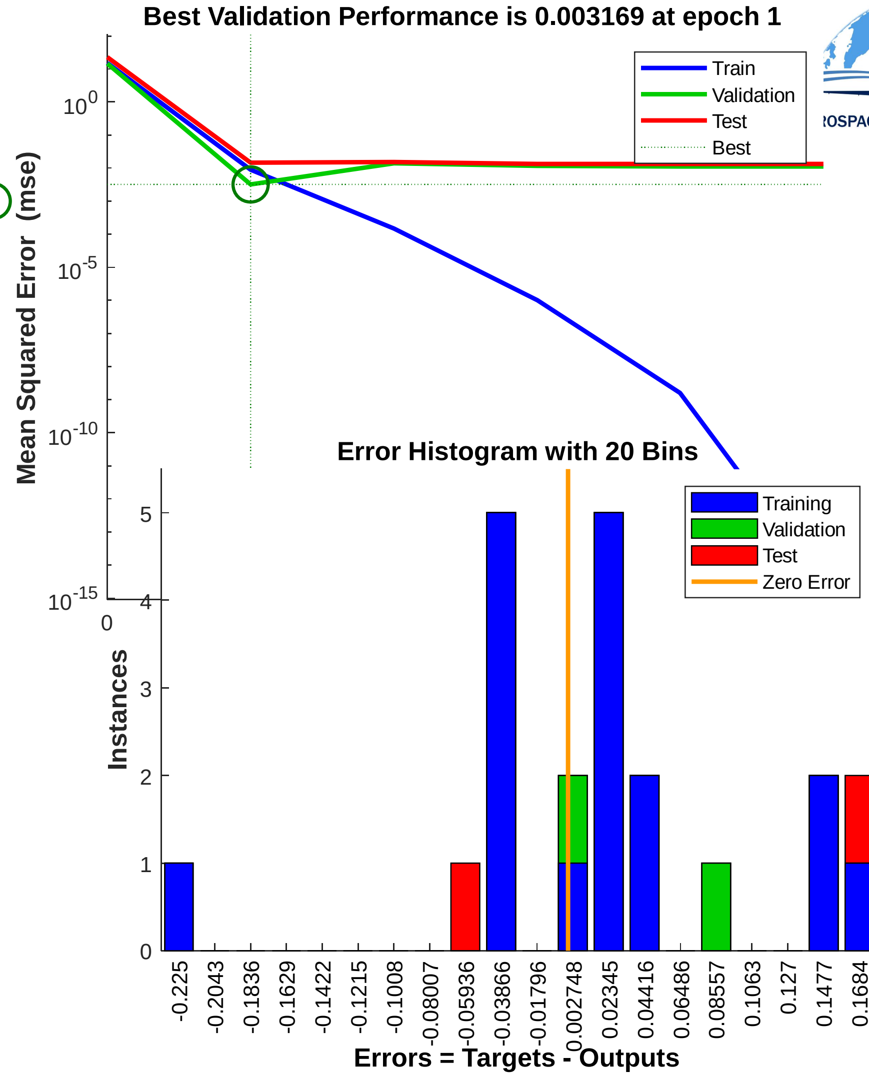

Function Fit for Output Element 1
Output and Target
Error
Training Targets
Training Outputs
Validation Targets
Validation Outputs
Test Targets
Test Outputs
Errors
Fit
Targets - Outputs

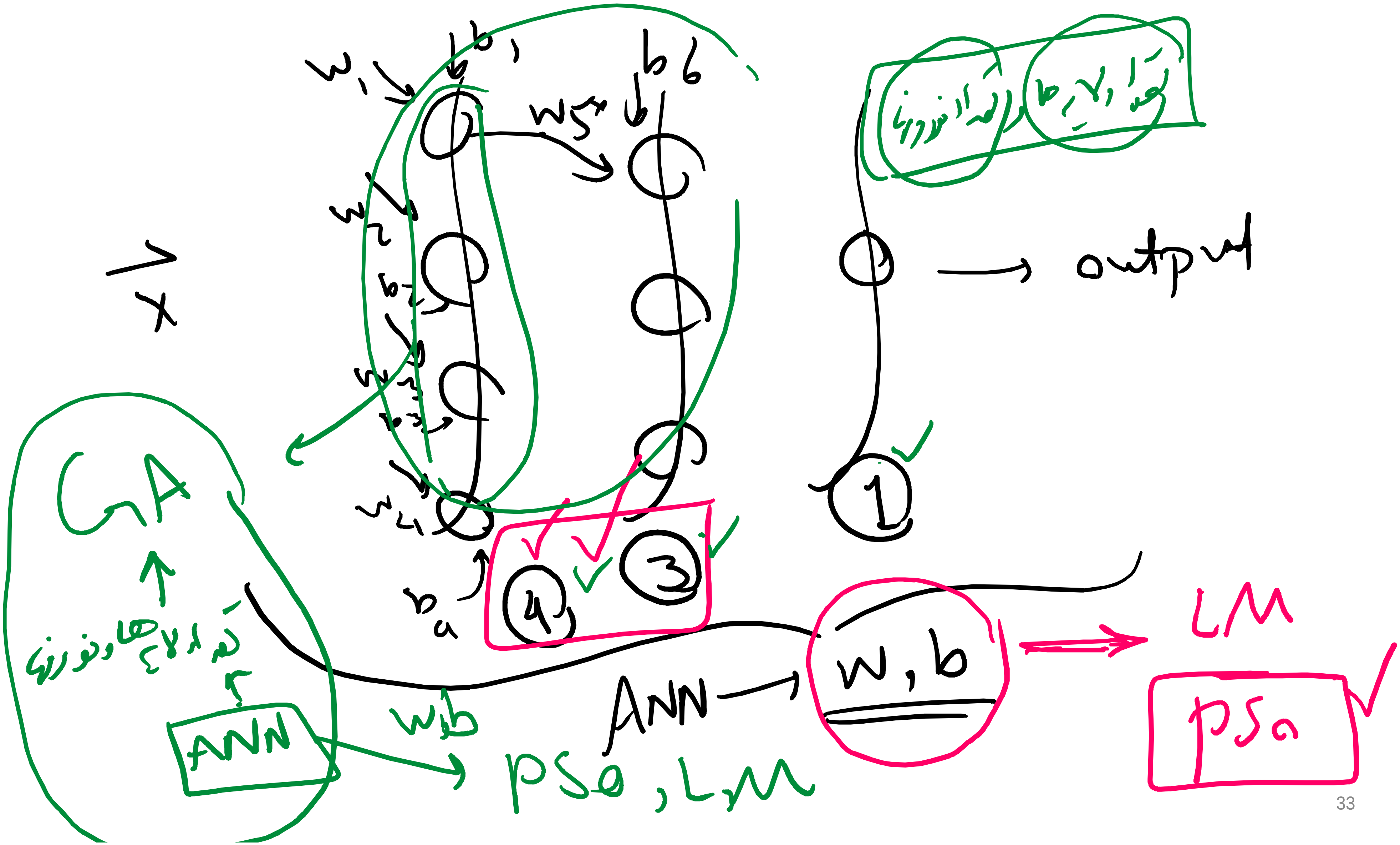

Best Validation Performance is 0.00093783 at epoch 418
Mean Squared Error (mse)
Train
Validation
Test
Best


Function Fit for Output Element 1
Output and Target
Error
Input
Training Targets
Training Outputs
Validation Targets
Validation Outputs
Test Targets
Test Outputs
Errors
Fit
Targets - Outputs


Function Fit for Output Element 1
Output and Target
Error
Training Targets
Training Outputs
Validation Targets
Validation Outputs
Test Targets
Test Outputs
Errors
Fit
Targets - Outputs


Best Validation Performance is 0.00093783 at epoch 418
Mean Squared Error (mse)
Train
Validation
Test
Best
Error Histogram with 20 Bins
Training
Validation
Test
Zero Error
Instances
-4.3e-05
-3.3e-05
-2.3e-05
-1.3e-05
-3.2e-06
6.67e-06
1.65e-05
2.64e-05
3.62e-05
4.61e-05
5.59e-05
6.58e-05
7.56e-05
8.55e-05
9.53e-05
0.000105
0.000115
0.000125
0.000135
0.000145
Errors = Targets - Outputs


Best Validation Performance is 0.003169 at epoch 1
Mean Squared Error (mse)
Train
Validation
Test
Best
Error Histogram with 20 Bins
Training
Validation
Test
Zero Error
Instances
-0.225
-0.2043
-0.1836
-0.1629
-0.1422
-0.1215
-0.1008
-0.08007
-0.05936
-0.03866
-0.01796
0.002748
0.02345
0.04416
0.06486
0.08557
0.1063
0.127
0.1477
0.1684
Errors = Targets - Outputs



X
b1
b6
w1
w2
w5
w21
b2
b3
b9
output
تعداد لایه ها
تعداد نورون ها
4
3
1
GA
تعداد لایه ها و نورون
ANN
w,b
PSO, LM
ANN
w, b
LM
PSO

$\vec{x}_{10\times 1} = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$

$$f(x) = \sum_{i=1}^{10} x \longrightarrow \vec{x} = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$$

$= 0$

$1\ 0\ 1$

$$2^{10} = 1024$$

Cup

10