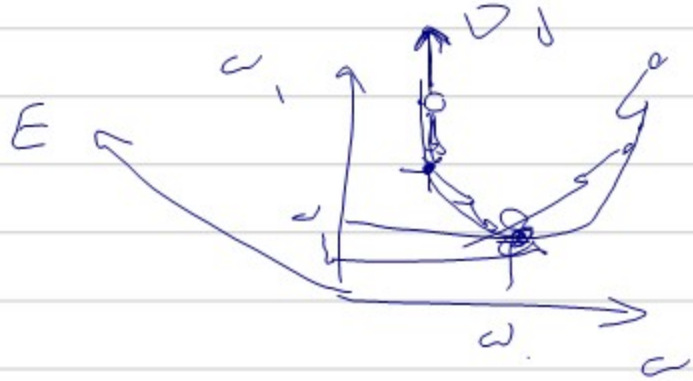
GD

جلسه ۳

$$MSE = J = \frac{\sum (y^{(i)} - \hat{y}^{(i)})^2}{m}$$

$$\frac{\partial J}{\partial w_0}, \frac{\partial J}{\partial w_1}$$



Gradient descent

$$J(w_0, w_1, \dots, w_n) \Rightarrow \nabla J = \begin{bmatrix} \frac{\partial J}{\partial w_0} \\ \frac{\partial J}{\partial w_1} \\ \vdots \\ \frac{\partial J}{\partial w_n} \end{bmatrix}$$

بردار گرادیان ←

الگوریتم GD

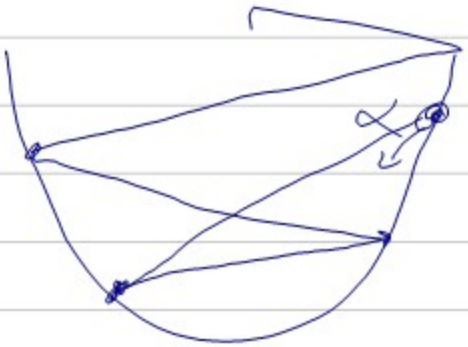
$w = w_0, w_1, \dots, w_n \leftarrow$ random

For steps in (100):

$$w_{new} = w_{old} - \alpha \nabla J$$

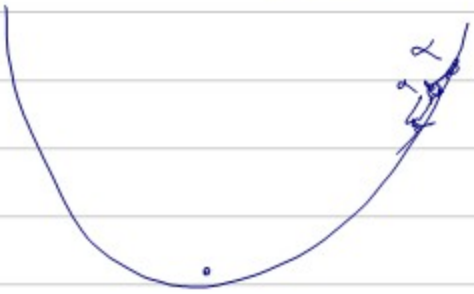
↓ descends

$\alpha \Rightarrow$ learning rate



α ↑ سرعت یادگیری ↑ احتمال گیر افتادن در بهینه محلی

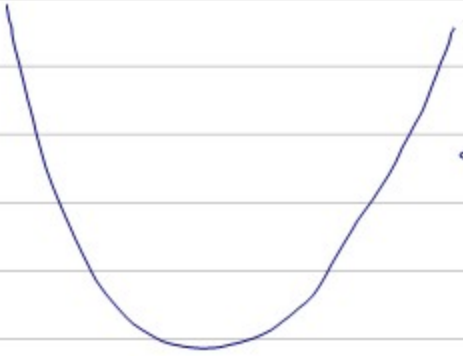
α ↓ زمان یادگیری ↑ احتمال گیر افتادن ↓



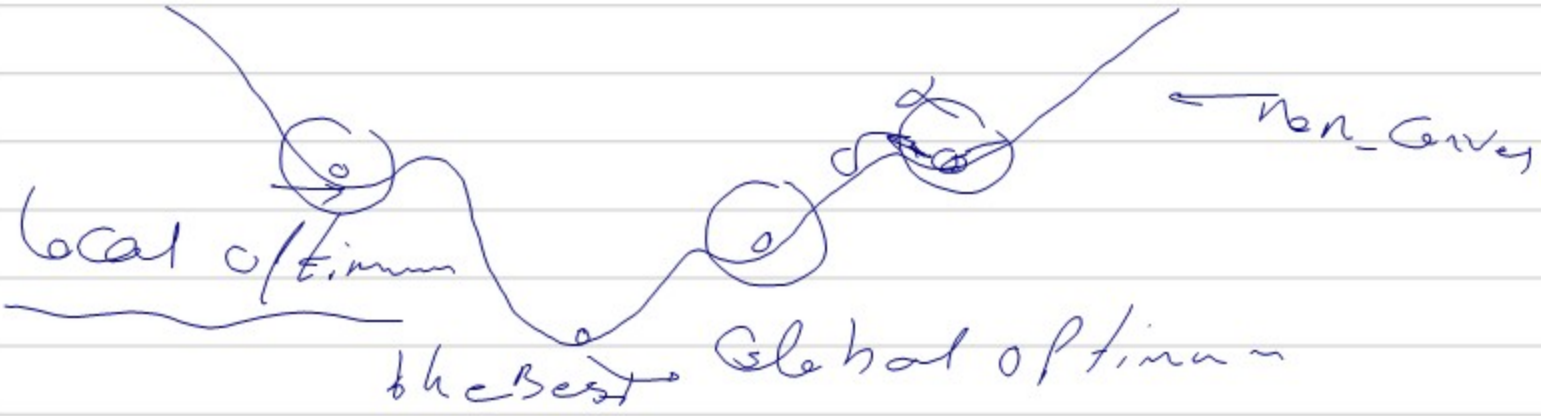
← Parameter

← HyperParameter

epoch



← Convex



← Non_Convex

local optimum

the Best → Global optimum

$$w^{t+1} = w^t \pm \alpha \nabla J$$

↓ G Descent

G Ascent

$\hat{y} = x^T \alpha$

$\hat{y} = w_0 + w_1 \alpha$

3 الگوریتم برای حل رابطه

② Batch Gradient Descent

در هر بار تکرار تمام داده‌ها را

بسیار کند

900

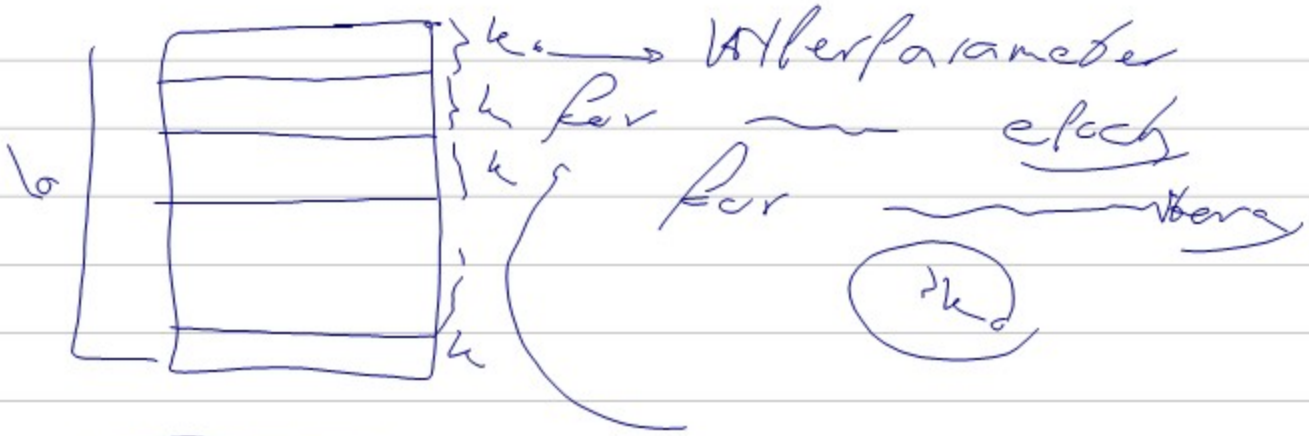
900

③ Stochastic GD

در هر بار تکرار یک داده را

آموزش می‌دهد

③ mini batch GD



$$\omega^{t+1} = \omega^t - \alpha * \nabla J$$

$$MSE = \frac{1}{b} \sum (\hat{y}^{(i)} - y^{(i)})^2 / m$$

$\hat{y} \rightarrow \omega_0 + \omega_1 x$

$$\nabla J \begin{cases} \frac{\partial J}{\partial \omega_0} = 2 \sum (\omega_0 + \omega_1 x - y) / m \\ \frac{\partial J}{\partial \omega_1} = 2 \sum x (\omega_0 + \omega_1 x - y) / m \end{cases}$$

$$\omega_0^{t+1} = \omega_0^t - \alpha \frac{\partial J}{\partial \omega_0}$$

$$\omega_1^{t+1} = \omega_1^t - \alpha \frac{\partial J}{\partial \omega_1}$$

from sklearn.linear_model import LinearRegression

pip install scikit-learn

model = L—()

model.fit(x_train, y_train)

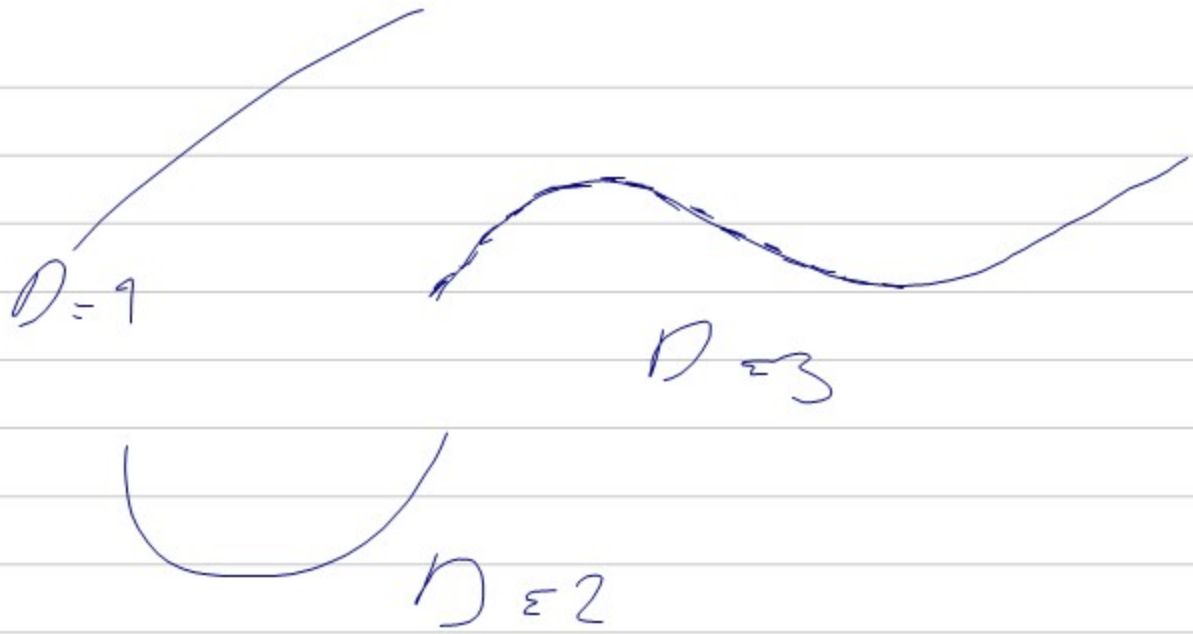
y_pred = model.predict(x_test)

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=10)

0.2 یعنی ۲۰٪ داده‌ها برای تست

random_state = 10


D=1
D=3
D=2

linear classification

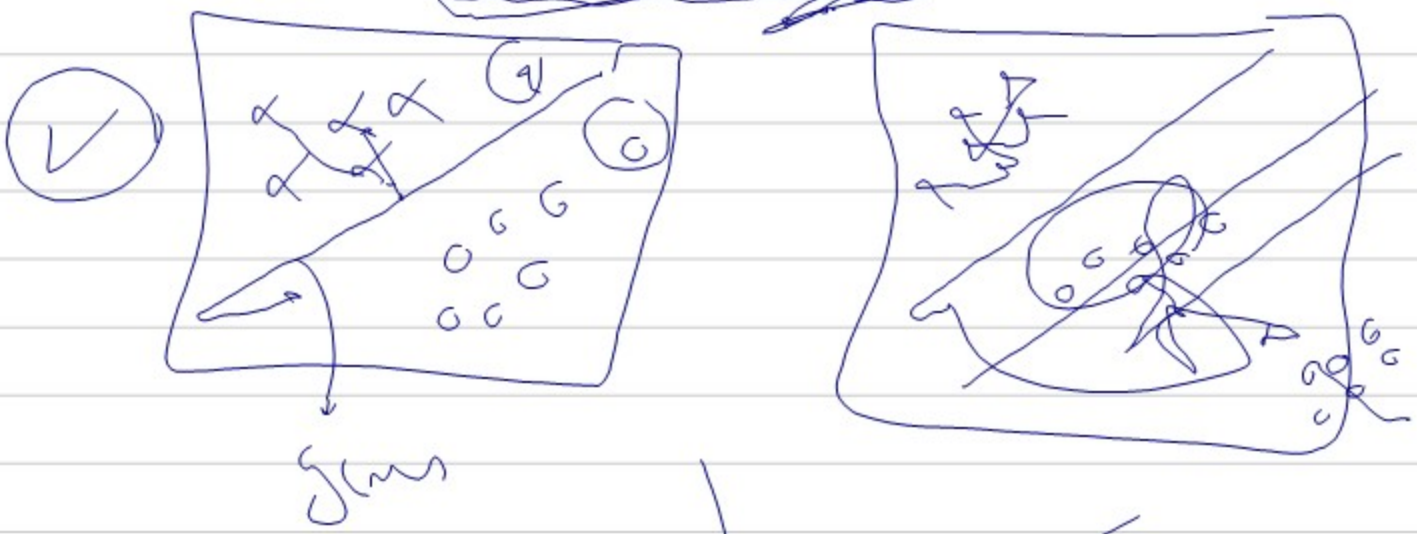
classification

$x \rightarrow y \in \{0, 1\}$

Regression

$x \rightarrow y \in \mathbb{R}$

$w_0 + w_1 x_1 + w_2 x_2 = 0$



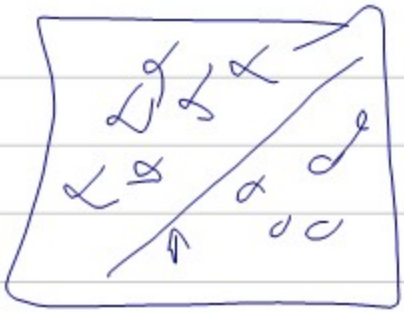
$g_2(x)$ $g_1(x)$

C_1 if $g_1(x) > g_2(x)$

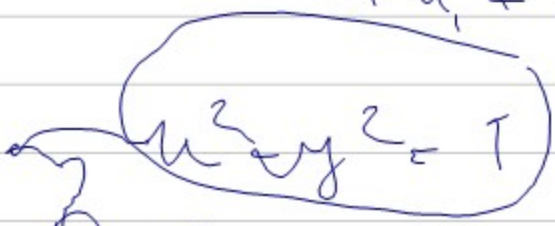
C_2 otherwise

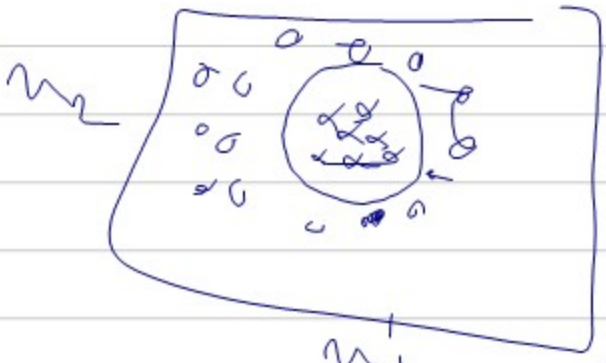
k-Class

$\hat{y} = \arg\max_i g_i(x)$



D = 1



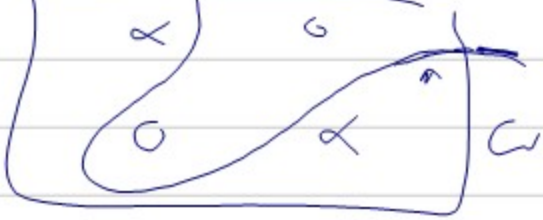


D = 2

$$\begin{bmatrix} 1 & x_1 & x_2 & x_1^2 & x_1 x_2 & x_2^2 \\ 1 & x_1 & x_2 & x_1^2 & x_1 x_2 & x_2^2 \\ 1 & x_1 & x_2 & x_1^2 & x_1 x_2 & x_2^2 \end{bmatrix}$$

$$w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + \dots$$

D = 3



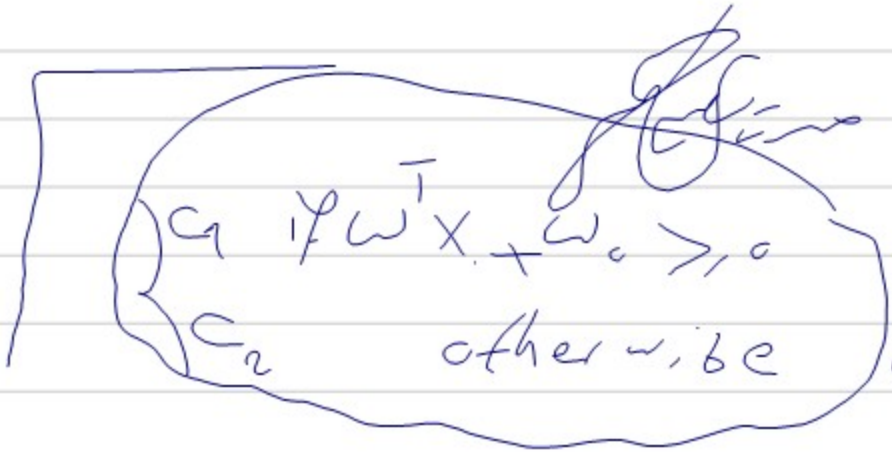
$$w_0 + w_1 x_1 + w_2 x_2 + \dots + w_D x_D = 0$$

$$w^T x$$

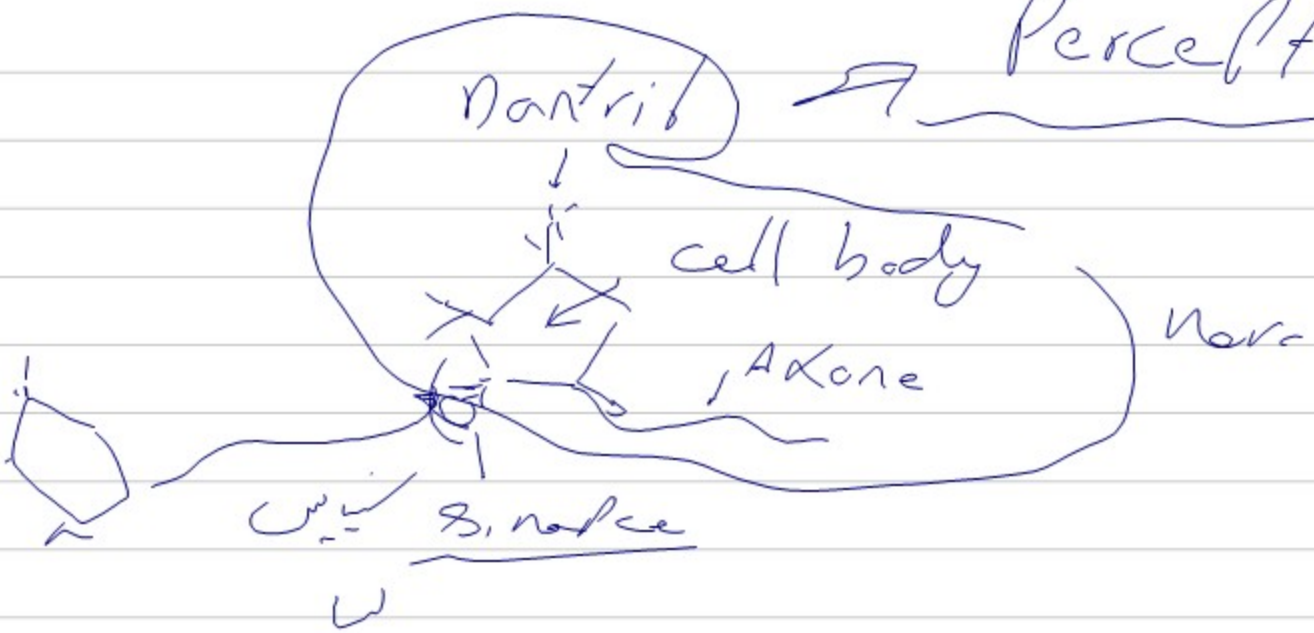
Decision boundary

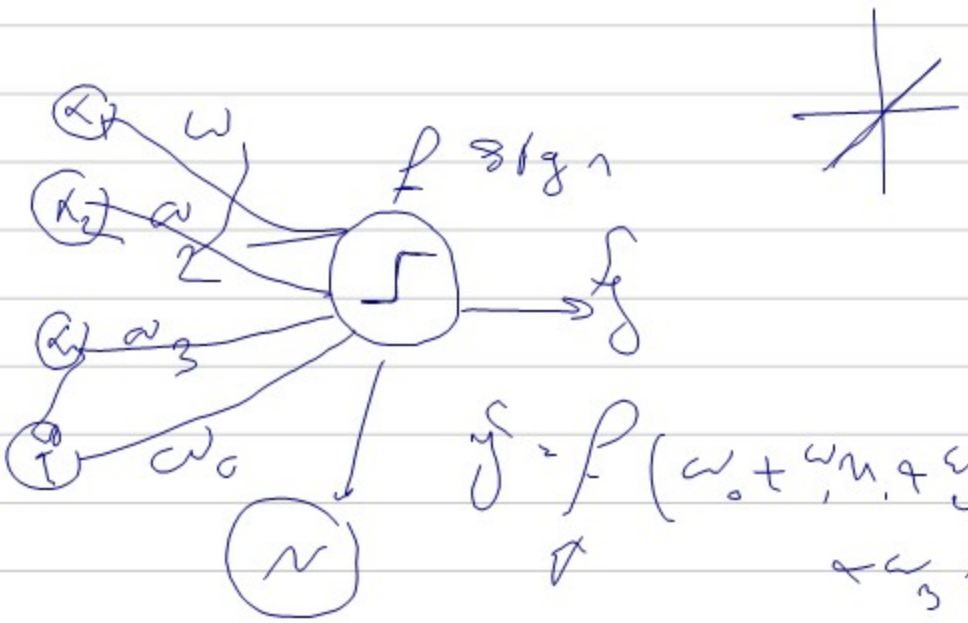
$$x = [1\ x_1\ x_2 \dots]$$

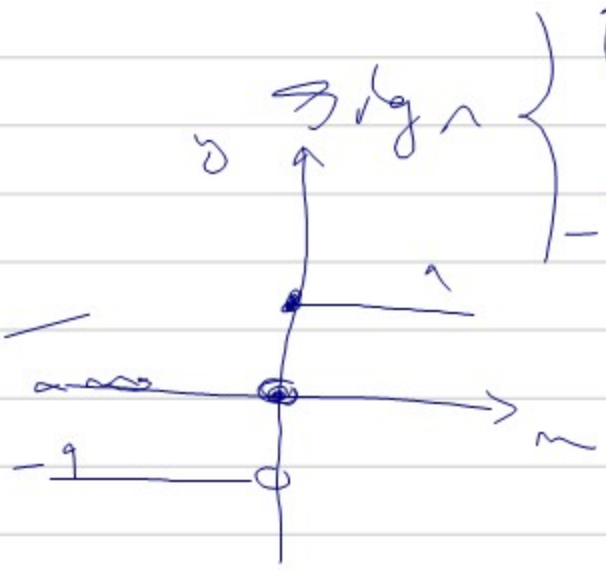
$$w = [w_0\ w_1\ w_2 \dots]$$

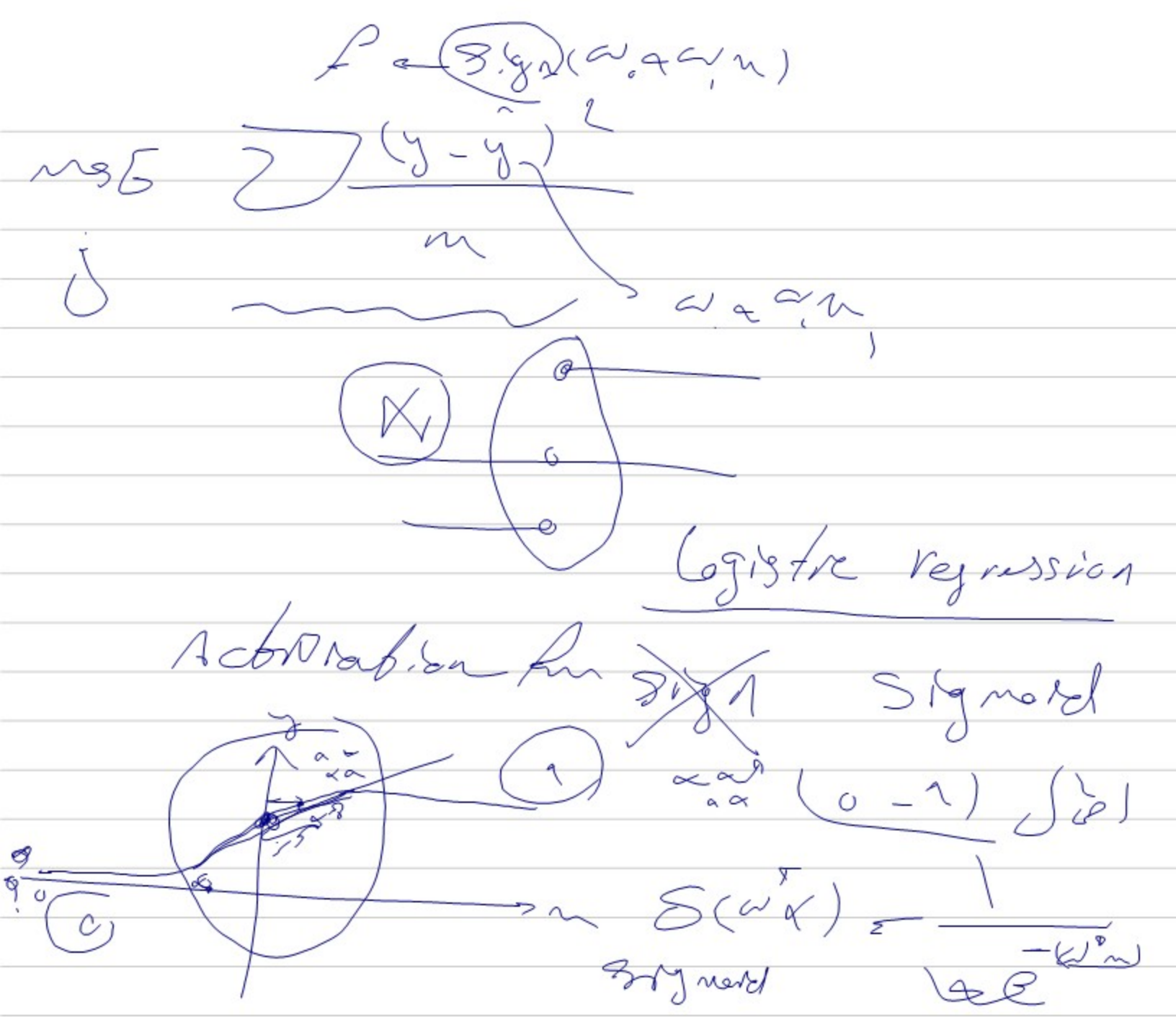


Perceptron


Dendrite
cell body
Axone
Nucleus
Synapse


sign
$y = f(w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3)$


sign
1
-1
$f(x) \geq 0$
$f(x) < 0$



Logistic regression

Activation function: Sigmoid

(0 - 1) الخرج

$$\delta(w^T x) = \frac{1}{1 + e^{-w^T x}}$$

Sigmoid

$$\sum (y - \delta(w^T x))^2$$

$$\nabla J \quad \left[\frac{\partial J}{\partial w} \right]$$

Preproc

MinMaxScaler (2)

$$y = \frac{(x - x_{min})}{x_{max} - x_{min}}$$

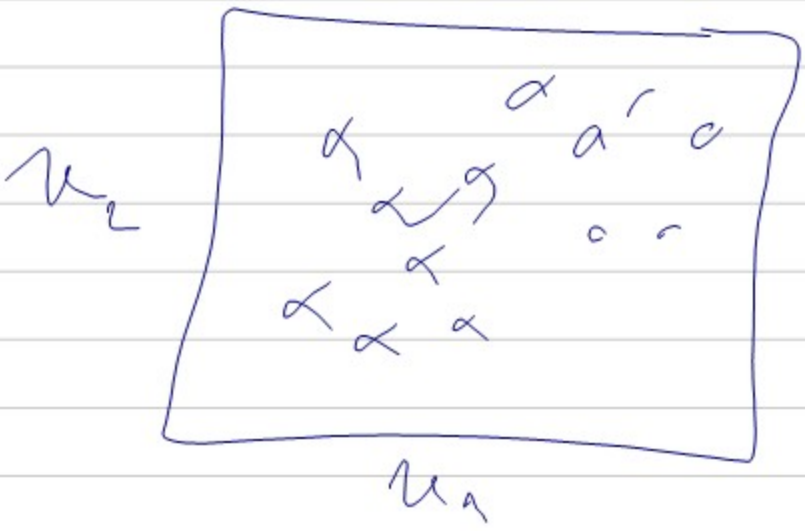
```
from sklearn.preprocessing import
                    MinMaxScaler

sc = MinMaxScaler()
sc = sc.fit_transformer(X_train)
                        X_test)
```

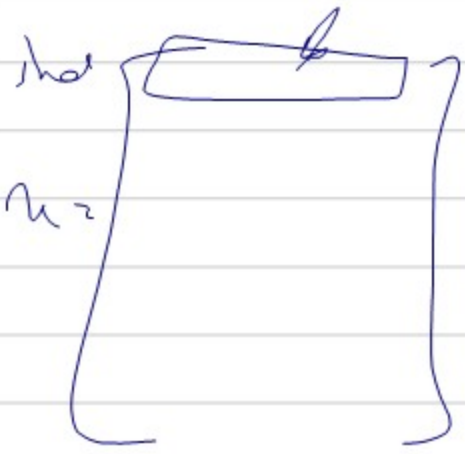
② Standard Scaler : نرمال‌سازی داده‌ها

میانگین صفر و واریانس یک

$$y = \frac{(x - \text{mean})}{\text{Std}}$$

Standard deviation

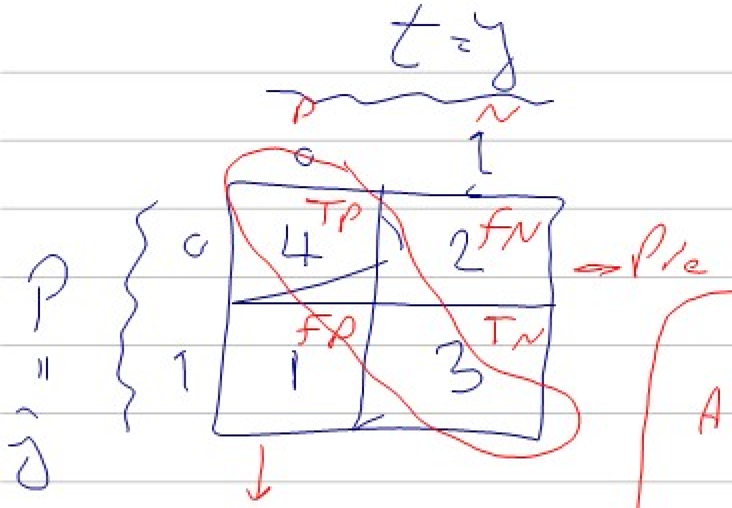




n → 10 → 5 → 0
5 → 1

Confusion matrix

2 ~ ~ → 2x2



$$Acc = \frac{TP + TN}{TP + FP + TN + FN}$$

$$sensitivity = \frac{TP}{TP + FP}$$

$$Precision = \frac{TP}{TP + FN}$$