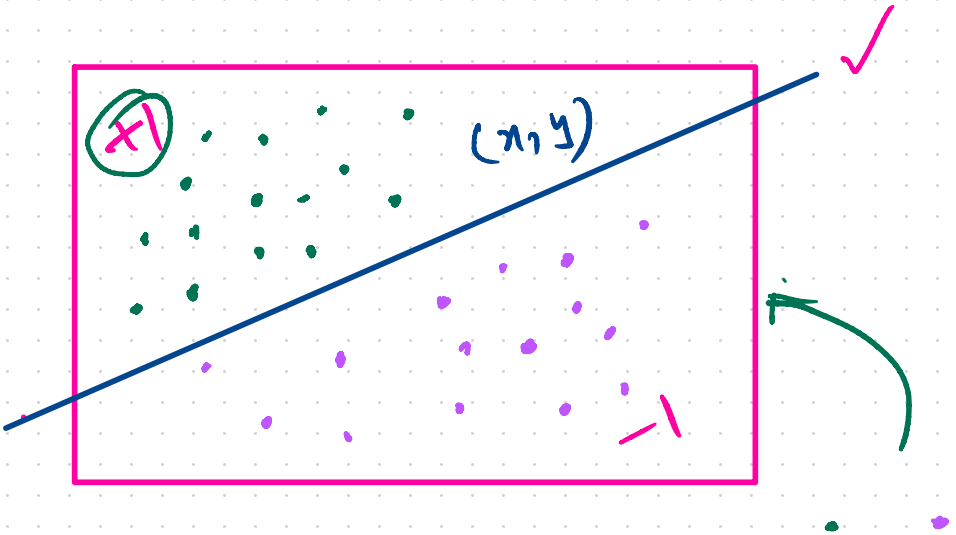
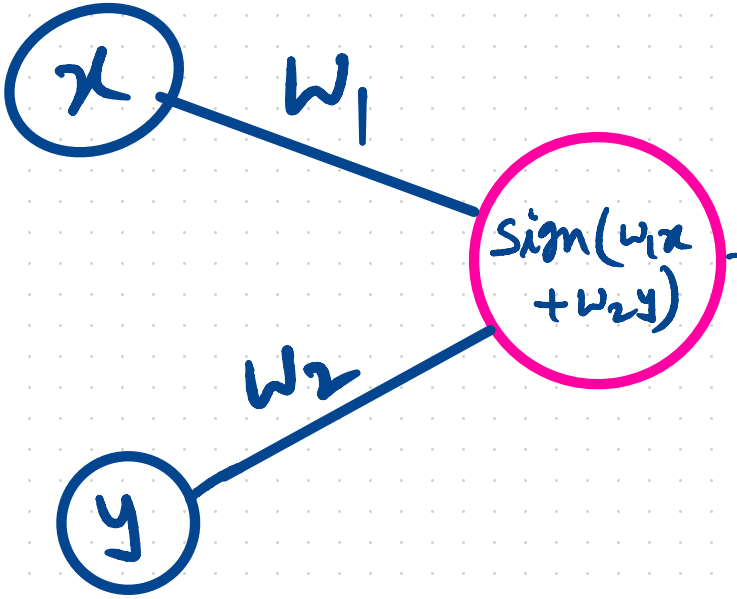
Day-9 1 JULY 2023

Break down the goal into tasks

- Understand terminology in ML
- Mathematical foundation of ML
  - Mathematical Definitions
  - Universal Approximation Theorem
- Mathematical Model of Artificial Neural Network(ANN)
  - Feed Forward
  - Backward Propagation
  - Activation and loss functions
- Implement algorithm for ANN in python
- Problem-1: Binary Classifier
- Introduction to python packages
- Problem-2: Digit Recognition
- Problem-3: Solving a Differential Equation



$w_1x + w_2y$



$e = t - o$

t	o	error
-1	-1	-1-(-1)=0 ✓
+1	+1	+1-(+1)=0 ✓
-1	+1	-1-(+1)=-2 ✓
+1	-1	+1-(-1)=2 ✓

$$w^{new} = w^{old} - \alpha \frac{dE}{dw}$$

$$\frac{dE}{dw} = -e \frac{df}{dO} x$$

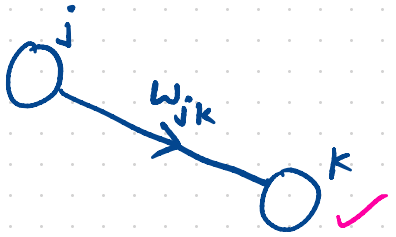
$$w^{new} = w^{old} + \Delta w$$

$$\Delta w = -e \cdot x$$

✓

$$w^{new} = w^{old} + \alpha (error)(input)$$

$$w_{jk}^{new} := w_{jk}^{old} - \alpha \frac{\partial E}{\partial w_{jk}}$$ ✓



$$w_{jk} = w_{jk} - \alpha ($$

$g'(o_k)$

$$\frac{\partial E}{\partial w_{jk}} = -e_k \, o_k (1 - o_k) \cdot o_j$$

$o_k (1 - o_k)$ ✓

$$sig(x) = sig(x)\,(1 - sig(x))$$

$$\frac{\partial E}{\partial w_{jk}} = -e_k g'(o_k) \cdot o_j$$

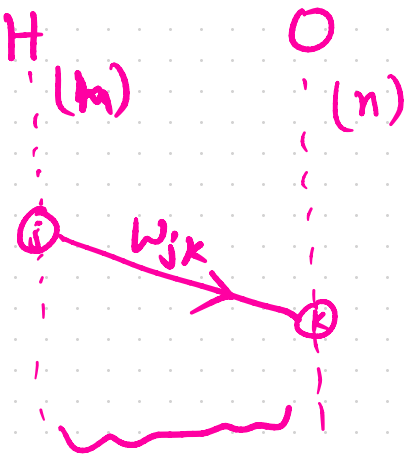
$$w_{jk} = w_{jk} - \frac{\partial E}{\partial w_{jk}}$$

$$w_{jk}^{n+1} = w_{jk}^{n} + e_k g'(o_k) \cdot o_j$$

$$\Delta w_{jk} = e_k g'(o_k) \cdot o_j$$

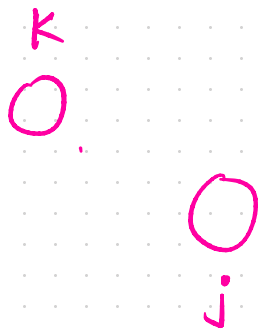
$$\begin{bmatrix} \Delta w_{11} & \Delta w_{21} & \cdots \\ \Delta w_{12} & \Delta w_{22} & \cdots \\ \vdots & \vdots & \\ & \Delta w_{jk} & \cdots \\ \vdots & \vdots & \end{bmatrix}_{n \times m}$$

$$= \begin{pmatrix} e_1 g'(o_1) \\ \vdots \\ e_k g'(o_k) \\ \vdots \end{pmatrix}_{n \times 1} \cdot \begin{pmatrix} o_1 & o_2 & \cdots & o_j & \cdots \end{pmatrix}_{1 \times m}$$



$$W_{HO}$$ $n \times m$

$$\Delta W = \alpha \cdot E_g^k \cdot O_j^T$$

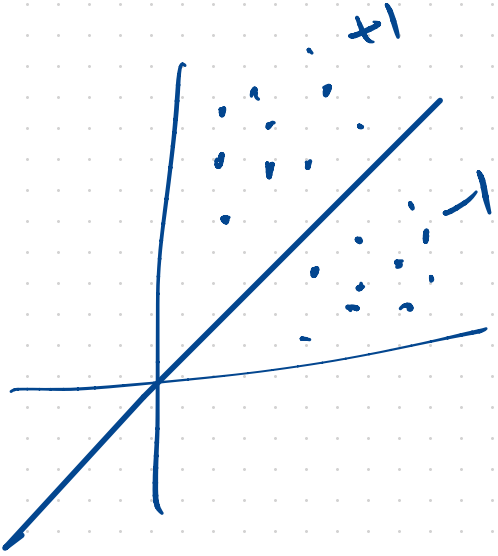


Point :

→ x

→ y

→ l



Perceptron :

→ weights

→ train(Pt) ✓

→ estimate(Pt) ✓

1. What if replace activation function with sigmoid ?

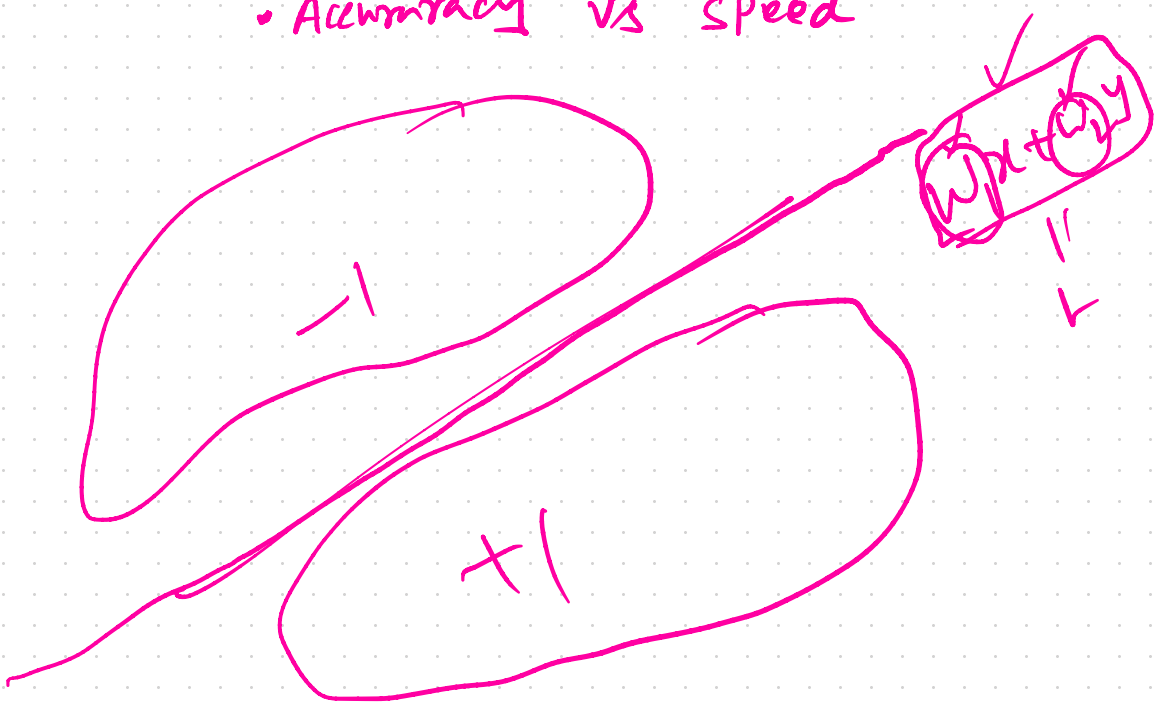
2. Does it better than Sign function?

3. About learning rate

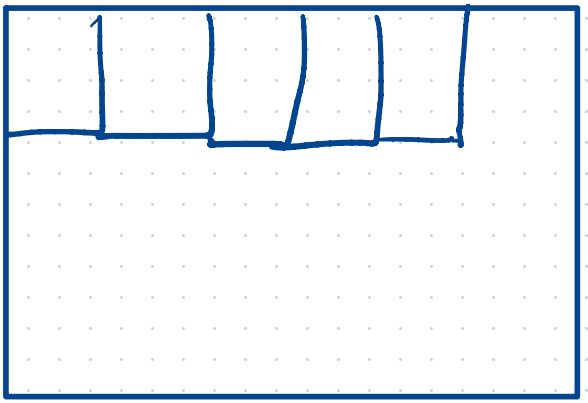
[0.2, 0.1, 0.05, 0.01, 0.001]

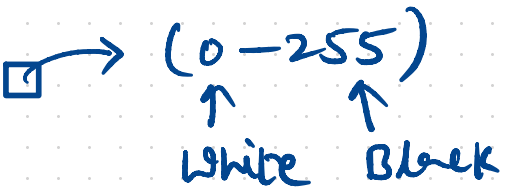
? ?

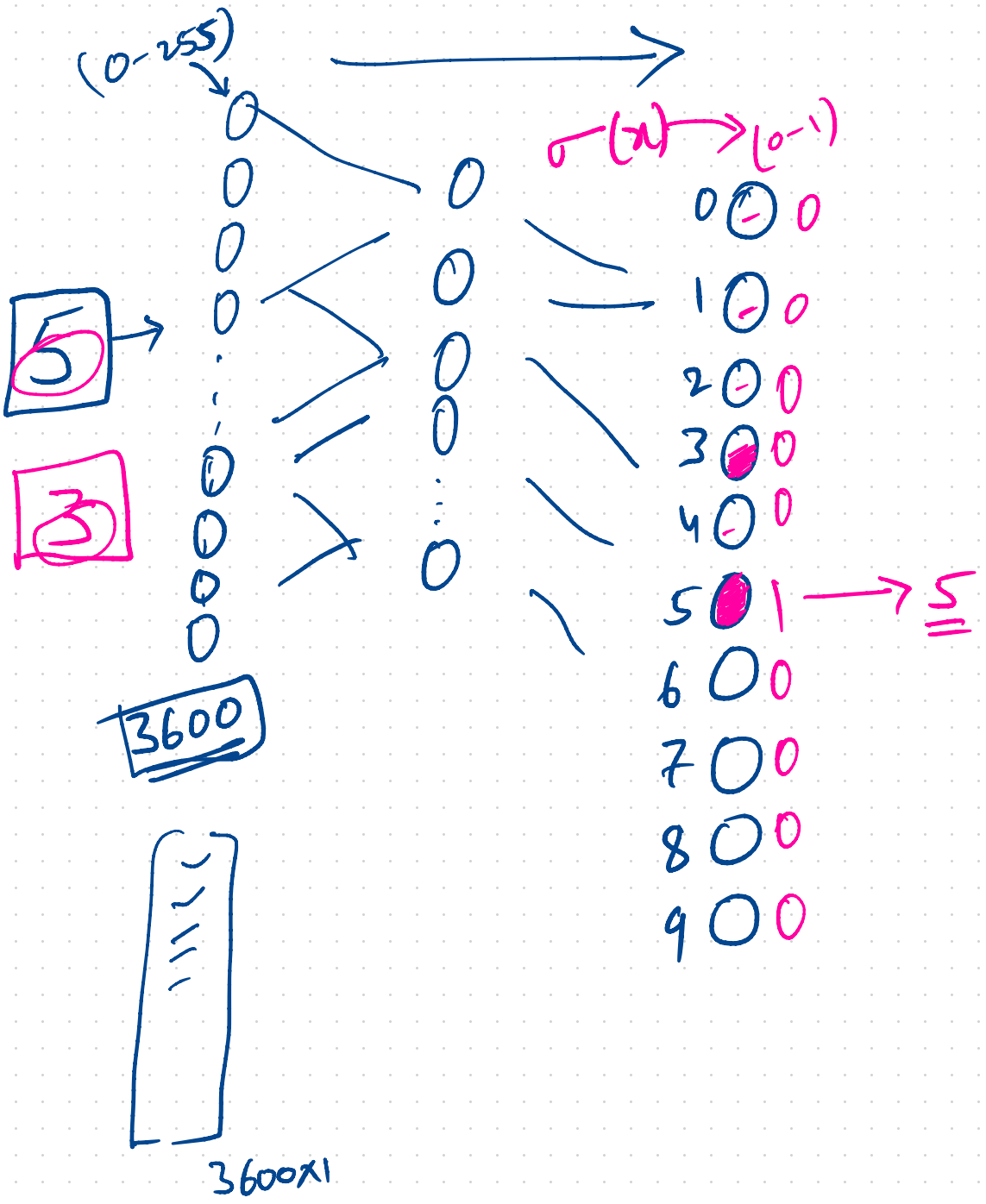
- Accuracy vs speed




5'
60X60



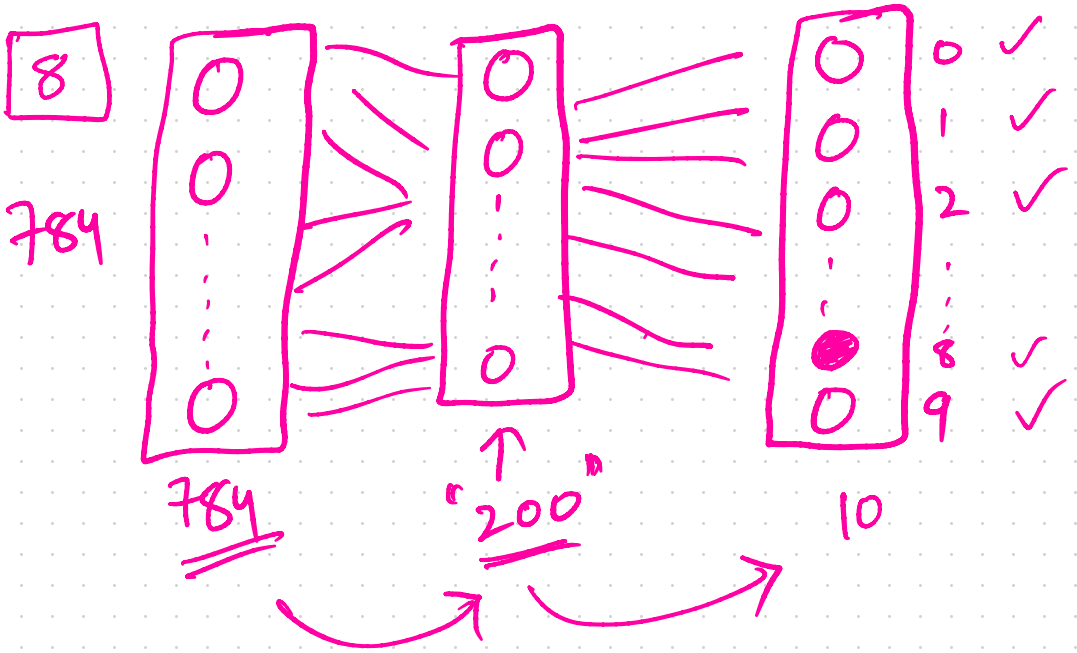

(0-255)
White
Black


(0-255)
5
3
3600
3600x1
σ(x)→(0-1)
0 0 0
1 0 0
2 0 0
3 0 0
4 0 0
5 0 1 → 5
6 0 0
7 0 0
8 0 0
9 0 0

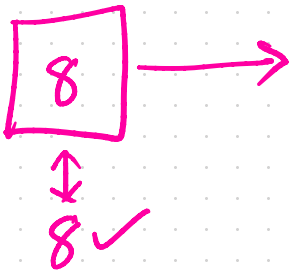
"mnist"

$28 \times 28 = 784$

(0-255)



`dr = newralNetwork(784, 200, 10, 0.1)`



0 1 2 3 4 5 6 7 8 9

target = [0, 0, 0, 0, 0, 0, 0, 0, 1, 0]

0.01 0.99