The Process of Learning
1.
2.
3.

ground truth

① Supervised Learning
- teacher → desired response
- learning system → actual response

→ error signal

key

adjustment

not a teacher

no desired response

② Reinforcement Learning
- critic ⇒ heuristic reinforcement
  - reward → You do a good job
  - penalty → You are bad!
- learning system

↓

train a robot!

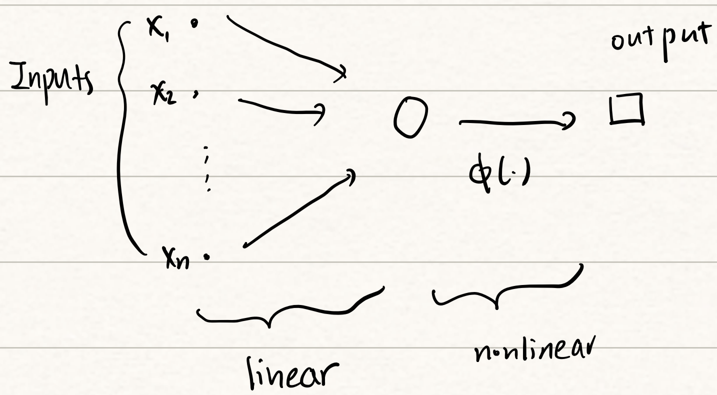
③ unsupervised learning ↕ self-organized

environment → learning system

Simplest NN → Perceptron

↓ Rosenblatt

Perceptron Convergence Theorem



Goal : Pattern Classification

Defn : (Notation)

$$\underline{x}(n) = [+1, x_1(n), \dots, x_m(n)]^T \in \mathbb{R}^{m+1}$$

$$\underline{w}(n) = [b(n), w_1(n), \dots, w_m(n)]^T \in \mathbb{R}^{m+1}$$

Decision Boundary ⟶ must be linear

① dimension = 1 ⇒ one point

② dimension = 2 ⇒ line

③ dimension = 3 ⇒ plane

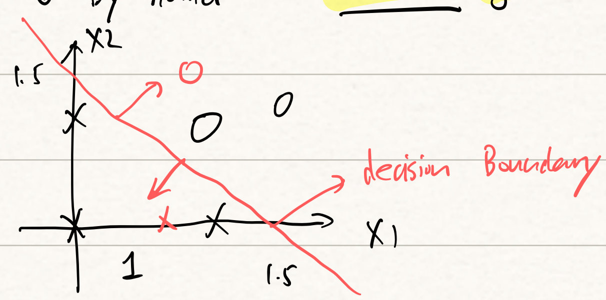
④ dimension ≥ 4 ⇒ hyper plane

How to learn weight?

① By hand

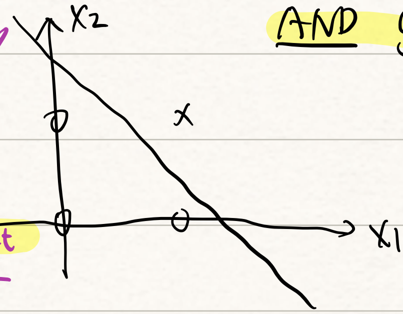
② Learning Process

① By hand NAND gate



$$-x_1 - x_2 + 1.5 = 0$$

AND gate



same decision boundary

different weight

for the opposite output

Reason ↑

fix

(logic gate)

$$x_1 + x_2 - 1.5 = 0$$

如果是人为设定的 pattern, then we don't care

the # of the output (1 or 0)

Limitation → Perceptron only works on linearly-separable dataset

↓

not work on 'XOR' dataset

② Learning process → $\boxed{\text{Trial and Error}}$

↑

work on high-dimension dataset ($d \geq 4$)

$\boxed{\text{How?}}$

$$w^{(k)} = w^{(k-1)} + y x \quad \text{if } y = \pm 1$$

now $y = 0/1$

error signal $e = d - y \Rightarrow \begin{cases} d=1,\ y=0 \Rightarrow e=1 \\ d=0,\ y=1 \Rightarrow e=-1 \end{cases}$

$$\Rightarrow w^{(k)} = w^{(k-1)} + e x$$

KEY POINT

① 更新 ⇔ $e(n) \cdot \langle w(n), x(n) \rangle < 0$

② $e(n) \ \langle w^*(n), x(n) \rangle \geq 0$

考虑 $\dfrac{\langle w^*, w(n+1) \rangle}{\|w\|^* \, \|w(n+1)\|} \leq 1$

w^* is the truth-weight

learning rate

we can choose μ big enough to make sure answer can be obtained in one step

this does not mean BIG IS GOOD!

Regression

$$D = \{ ((\underline{x}(i), d(i)) \}_{i=1}^{n}$$

$e(i) = d(i) - y(i) \rightarrow$ error signal

$y(i) = \text{function}(\underline{x}(i))$

optimization problem with the help of cost function

$$E(w) = \sum_{i=1}^{n} e(i)^2 = \sum_{i=1}^{n} (d(i) - y(i))^2$$

$$E(w) = e^T e \qquad e = \begin{pmatrix} e(1) \\ \vdots \\ e(n) \end{pmatrix}$$

$$= (d-y)^T (d-y) \qquad y = \begin{bmatrix} x(1)^T w \\ \vdots \\ x(n)^T w \end{bmatrix}$$

$$\overset{c}{=} y^T y - 2 d^T y$$

$$\underbrace{X = \begin{pmatrix} x(1)^T \\ \vdots \\ x(n)^T \end{pmatrix}_{n \times (m+1)}} \qquad = w^T X^T X w - 2 d^T X w \qquad = Xw$$

$$\nabla E(w) = 2 X^T X w - 2 X^T d = 0 \Rightarrow \underbrace{w = (X^T X)^{-1} X^T d}$$

② $\nabla E(w) = \nabla_w e \, \nabla_e E$

$\Rightarrow \nabla E(w) = X^T \cdot 2e$

$$= X^T(2y-2d)$$

X^TX invertible as long as we have enough data

$$= 2X^TXw - 2X^Td$$

LMS Algorithm

$$e = d - y = d - w^Tx$$

$E(w) = \frac{1}{2}e^2(n) \rightarrow$ one error signal

$$\frac{\partial E}{\partial w} = \frac{\partial E}{\partial e}\frac{\partial e}{\partial w} = -e(n) \cdot x^T(n)$$

$$g(n) = \left(\frac{\partial E}{\partial w}\right)^T = -e(n)\,x(n)$$

$$\Downarrow$$

$$w(n+1) = w(n) + \underset{(n)}{\eta}\, e(n)\, x(n)$$

Both Perceptron & LMS Alg. $\rightarrow$ error-correction-learning