EE5904 Multi-layer Perceptron (MLP)

Last week:

① simple perceptron $\leftrightarrow$ Pattern Recognition

linearly separable $\Longleftrightarrow$ $\exists\, w$, s.t classify correctly !

$\begin{cases} \text{off-line} \\ \text{on-line} \rightarrow \text{Perceptron Convergence Alg.} \end{cases}$

② Regression

input $\begin{cases} x_1(i) \\ x_2(i) \\ \vdots \\ x_m(i) \end{cases}$ $\longrightarrow$ Unknown System $\longrightarrow$ output $y(i)$

$y(i) = v(i) = \underline{w}^T(i)\, \underline{x}(i)$

learning framework

LLS $\leftarrow$ ① $E(w)$ cost f^n

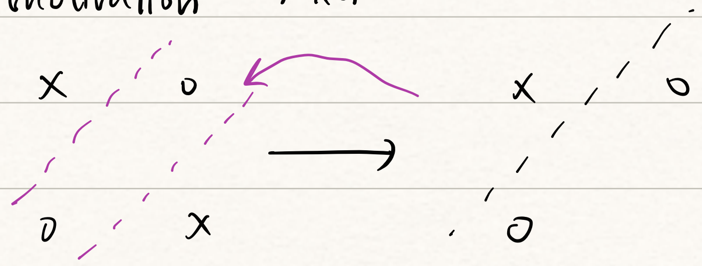
LMS $\leftarrow$ ② $\begin{cases} \frac{\partial E}{\partial w} = 0 \rightarrow \text{if task is easy } \left(\frac{\partial E}{\partial w} = 0 \text{ has closed form sol}^n\right) \\ \quad \rightarrow \text{solve directly} \\ \text{iterated method} \\ \quad \Downarrow \begin{cases} \text{GD} \\ \text{Newton's Method} \end{cases} \end{cases}$

Limits

$\begin{cases} ① \text{ classification} \rightarrow \text{linearly separable} \\ ② \text{ reg} \rightarrow \text{linear model} \end{cases}$

This week:

$\underset{\sim\sim}{\text{MLP}}$ → motivation → XOR



How to design? ⇒ in lecture note!

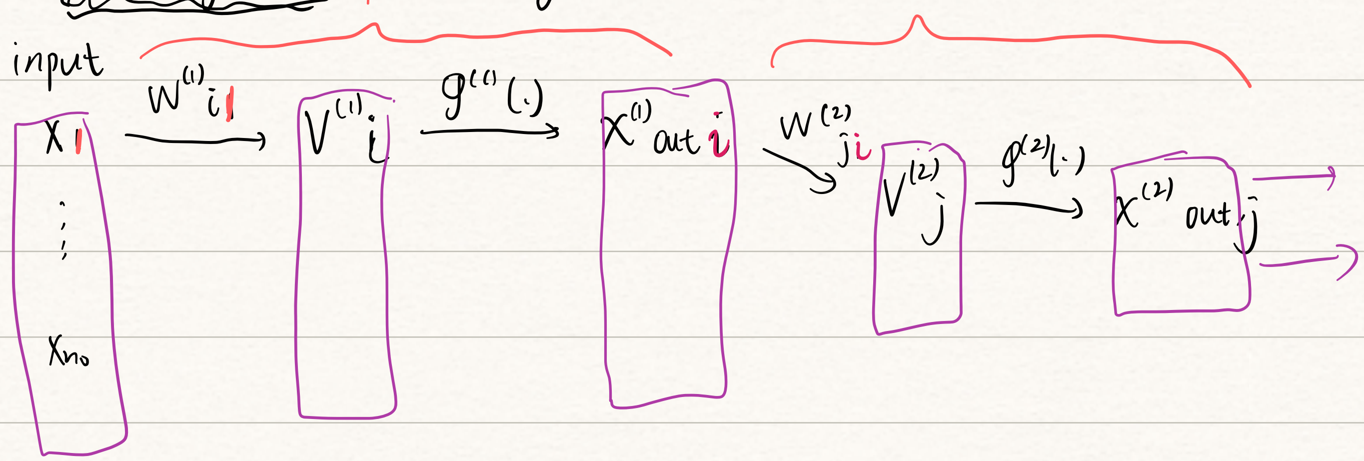
MLP ↔ David Rumelhart & PDP

① hidden layer ↔ neuron y_j (not include v_j → induced local field)

$$\underbrace{v_j \xrightarrow{\text{sigmoid}} y_j}_{\substack{\downarrow \\ \text{activation } f^{=}}}$$

1-st hidden neuron

② $\boxed{\text{BP Algorithm}}$ (Learning) 2-nd



$\underline{d}(n)$ → desired output $\underline{y}(n)$ → actual

→ $\underline{e}(n) = \underline{d}(n) - \underline{y}(n)$

we want $\underline{w}(n+1) = \underline{w}(n) + \underline{\Delta w}(n)$

$\Delta w(n) := -\eta \nabla E(n)$

Define cost $f^{\underline{n}}$ $E(n) = \frac{1}{2}\, \underline{e}(n)^T \underline{e}(n)$

$$\nabla_w E(n)^T = \frac{\partial E(n)}{\partial w(n)}$$

$$\Delta w_{ji}^{(3)}(n) = -\eta \frac{\partial E(n)}{\partial w_{ji}^{(3)}(n)}$$

$$\Delta w_{ji}^{(3)}(n) = -\eta \frac{\partial E(n)}{\partial w_{ji}^{(3)}(n)}$$

$$E(n) = \frac{1}{2}\sum_{j=1}^{n_3} e_j(n)^2 = \frac{1}{2}\sum_{j=1}^{n_3}\left(d_j(n) - x_{out,j}^{(3)}(n)\right)^2$$

① $$\frac{\partial E(n)}{\partial x_{out,j}^{(3)}(n)} = \left(d_j(n) - x_{out,j}^{(3)}(n)\right)(-1)$$

$$= -e_j(n)$$

② $y_j(n) = x_{out,j}^{(3)} = \phi^{(3)}\left(v_j^{(3)}(n)\right)$ $\quad \frac{\partial x_{out,j}^{(3)}(n)}{\partial v_j^{(3)}(n)} = \phi^{(3)\prime}\left(v_j^{(3)}(n)\right)$

③ $$\frac{\partial v_j^{(3)}(n)}{\partial w_{ji}^{(3)}(n)} = x_{out,i}^{(2)}(n)$$

$$v_j^{(3)}(n) = \sum_{i=1}^{n_2} w_{ji}^{(3)}(n)\, x_{out,i}^{(2)}(n)$$

$$\frac{\partial E(n)}{\partial w_{ji}^{(3)}(n)} = \frac{\partial E(n)}{\partial x_{out,j}^{(3)}(n)} \cdot \frac{\partial x_{out,j}^{(3)}(n)}{\partial v_j^{(3)}(n)} \frac{\partial v_j^{(3)}(n)}{\partial w_{ji}^{(3)}(n)}$$

$$= -e_j(n)\, \phi^{(3)\prime}\left(v_j^{(3)}(n)\right) x_{out,i}^{(2)}(n)$$

$$:= -\delta_j^{(3)}(n)\, X_{out,i}^{(2)}(n)$$

$$\Rightarrow \Delta W_{ji}^{(3)}(n) = -\eta \frac{\partial E(n)}{\partial W_{ji}^{(3)}(n)}$$

similar to $g(n)$

$$= \eta \underbrace{\delta_j^{(3)}(n)}_{\text{output}} \underbrace{X_{out,i}^{(2)}(n)}_{\text{input}}$$

$W^{(2)} \rightarrow$ 更复杂了

$$\frac{\partial E(n)}{\partial W_{ji}^{(2)}(n)} = \sum_{k=1}^{n_3} \left(\frac{\partial E_n}{\partial X_{out,k}^{(3)}(n)} \frac{\partial X_{out,k}^{(3)}(n)}{\partial V_k^{(3)}(n)} \cdot \frac{\partial V_k^{(3)}(n)}{\partial X_{out,j}^{(2)}(n)} \cdot \frac{\partial X_{out,j}^{(2)}(n)}{\partial V_j^{(2)}(n)} \cdot \frac{\partial V_j^{(2)}(n)}{\partial W_{ji}^{(2)}(n)} \right)$$

$$= \sum_{k=1}^{n_3} \left(-\delta_k^{(3)}(n)\, W_{kj}^{(3)}(n)\, \phi^{(2)\prime}\left(V_j^{(2)}(n)\right) X_{out,i}^{(1)}(n) \right)$$

$$= \phi^{(2)\prime}\left(V_j^{(2)}(n)\right) X_{out,i}^{(1)}(n) \sum_{k=1}^{n_3} \left(-\delta_k^{(3)}(n)\, W_{kj}^{(3)}(n)\right)$$

$$:= -\delta_j^{(2)}(n)\, X_{out,i}^{(1)}(n)$$

$$\Rightarrow W_{ji}^{(2)}(n+1) = W_{ji}^{(2)}(n) + \eta\, \delta_j^{(2)}(n)\, X_{out,i}^{(1)}(n)$$

output input

$$\delta_j^{(3)} = \underbrace{(d_j - X_{out,j}^{(3)})}_{\text{network error}} \underbrace{\phi'(V_j^{(3)})}_{\text{gradient}}$$ ⇒ output layer

↓

output error

error propagate through network

$$\delta_j^{(2)} = \underbrace{\left(\sum_{k=1}^{n_3} \delta_k^{(3)} W_{kj}^{(3)}\right)}_{\text{output error}} \underbrace{\phi'(V_j^{(2)})}_{\text{gradient}}$$

error propagate through network

$$\delta_j^{(1)} = \left(\sum_{k=1}^{n_2} \delta_k^{(2)} W_{kj}^{(2)}\right) \phi'(V_j^{(1)})$$

hidden layer → output error propagate backwards

BR

Rate of Learning

$\boxed{\text{Momentum}}$

Origin: $W_{ji}^{(s)}(n+1) = W_{ji}^{(s)}(n) + \eta^{(s)} \delta_j^{(s)}(n) X_{out,i}^{(s-1)}$

↓

Mom: $\Delta W_{ji}^{(s)}(k) = \alpha \underbrace{\Delta W_{ji}^{(s)}(n-1)}_{\text{history}} + \eta^{(s)} \delta_j^{(s)}(n) X_{out,i}^{(s-1)}(n)$

epoch: use whole training samples!

Stopping criteria

Summary

Initialization

↓

present training samples

↓

forward computation

↓

Backward propagation

↓

whether stop

↓

terminate

$\boxed{\text{Limitation:}}$ → Black-Box

↓

可解释性差