LEC 6

$\boxed{\text{Recap}}$

1. NN → universal approximator
   - Shallow NN ⟺ $\boxed{\text{MLP}}$
   - Deep NN

3. (Also, we talks about NN will not suffer curse of dimensionality)

2. Optimization
   - $\boxed{\text{GD}}$
   - $\boxed{\text{SGD with mini-batch}}$
     - → learning rate ⟺ Trade-off { convergence rate / quality of solution }
     - → batch size

$\boxed{\text{Architecture of NN}}$

→ $\boxed{\text{Last Week}}$ → fully-connected NN architecture (FCNN)

→ simulate the neuron structure

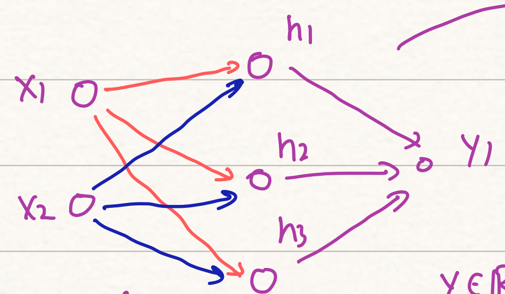
$\boxed{\text{Issue}}$ → permutation Invariance (naturally)

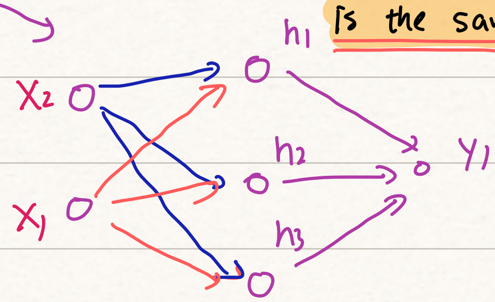
$\{1,2,3\} \to \{3,1,2\}$

FCNN Invariance → $\boxed{本质}$ ⇒ ① $w^T x$ is permutation invariant

② after permutation, the hypothesis space is the same!!!

$\boxed{\text{permutation}}$





$X = (x_1, x_2) \in \mathbb{R}^2$ input

$h \in \mathbb{R}^3$ hidden

$y \in \mathbb{R}$ output

$\boxed{\text{Procedure}}$

$$\begin{cases} x_i = (x_i^1, \ldots, x_i^d) \in \mathbb{R}^d \\ y_i \in \mathbb{R} \end{cases}$$

$$\boxed{\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N}$$

$\hat{\theta}_1$

① we train the NN by minimizing $R_{emp}(\underline{\theta}) = \frac{1}{N}\sum_{i=1}^{N} \ell(f_\theta(x_i), y_i) \quad \theta \in \mathcal{H}$

② if we apply PERMUTATION to $x_i \rightsquigarrow \boxed{x_i^p}$

$$\boxed{\min_{\underline{\theta} \in \mathcal{H}} : R^p_{emp}(\underline{\theta}) = \frac{1}{N}\sum_{i=1}^{N} \ell(f_\theta(x_i^p), y_i)} \rightarrow \hat{\theta}_2$$

then $\hat{\theta}_1$ & $\hat{\theta}_2$ is different only under permutation

---

Mathematically, FCNN invariance is:

$\Rightarrow$ $f \in \mathcal{H}$, then for any permutation p,

$f_p(x_1, \dots, x_d) := f(x_{p(1)}, \dots, x_{p(d)})$, then $f_p \in \mathcal{H}$

this is very bad when inputs have natural order (sequence) (spatial/temporal structure)

$\boxed{\text{not a good idea to use FCNN}}$

- images
- stoke prices
- text

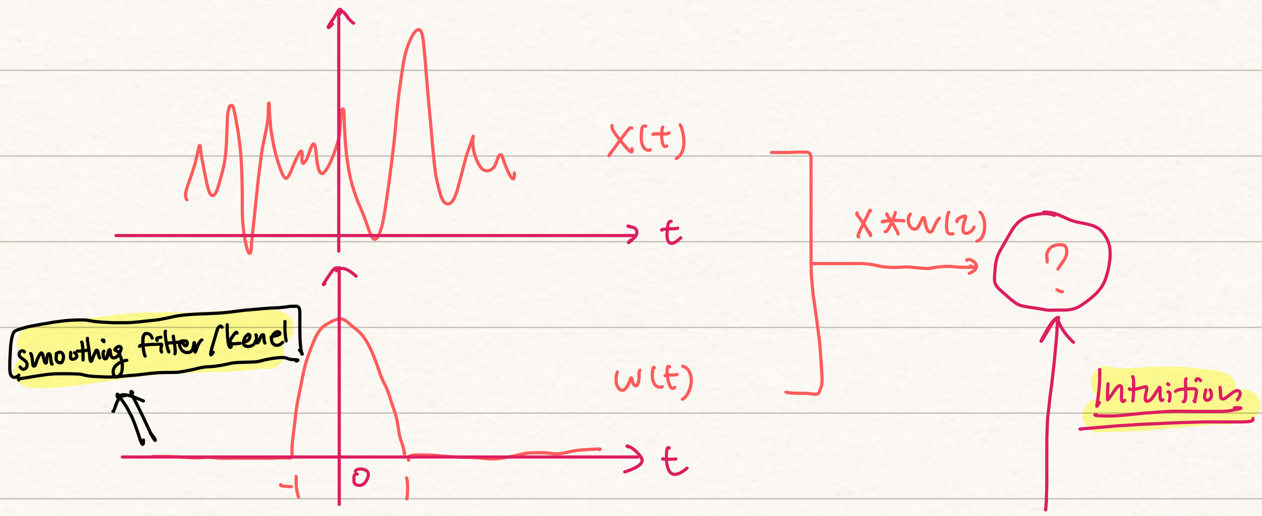
$\boxed{\text{CNN}}$

① convolution operation

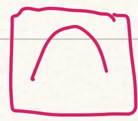
a) $w * x(\tau) = \int_{-\infty}^{+\infty} w(t)\, x(\tau - t)\, dt = \int_{-\infty}^{+\infty} x(t)\, w(\tau - t)\, dt = x * w(\tau)$

$$\rightarrow \begin{cases} \text{commutative} \Rightarrow w * x(\tau) = x * w(\tau) \\ \text{linearity} \Rightarrow w * (\lambda_1 x_1 + \lambda_2 x_2)(\tau) = \lambda_1 w * x_1(\tau) + \lambda_2 w * x_2(\tau) \end{cases}$$

b) what can conv. do?



$$w * x(\tau) = \int_{-1}^{1} \underline{w(t)}\, x(\tau - t)\, dt \rightarrow \text{average the interval } [-1+\tau, 1+\tau] \text{ of } x(t) \text{ to achieve } x * w(\tau)$$



$\Uparrow$

use $w(t)$ to smoothen $x(t)$, achieving $w * x(t)$ which is a smoothing signal

c) Discrete Convolution $\longrightarrow$ real-life application

$$\Rightarrow (w * x)(k) = \sum_{i=-\infty}^{+\infty} w(i)\, x(k-i) \qquad \text{(discrete convolution)}$$

(discrete Cross-Correlation) $\longrightarrow$ applied in CNN

$$\Rightarrow (w * x)(k) = \sum_{i=-\infty}^{+\infty} w(i)\, x(k+i)$$

truncate

$$(w * x)(k) = \sum_{i=1}^{m} w(i)\, x(k+i) \qquad \boxed{k = 0, 1, 2, \dots, d-m}$$

where $w \in \mathbb{R}^m$ $x \in \mathbb{R}^d$ $\boxed{m < d}$ $w * x \in \mathbb{R}^{d-m+1}$

kernel signal

dimension is decreasing

↓ BAD

cannot have large number of conv.

solved by padding (zero padding)

$$\rightarrow x \in \mathbb{R}^d \xrightarrow{\text{pad}} x_{pad} \in \mathbb{R}^{d+m-1} \xrightarrow{w(\cdot)} w * x_{pad} \in \mathbb{R}^d$$

填充 m-1 个 0

dimension preserve

In CNN

channel 1

channel 2

channel 3

w_{11}, w_{12}, w_{13}

$\sigma(\sum_k w_{1k} * x_k)$

d) multi-dimension conv.

2-D conv: $(w * x)(k,\ell) = \sum_{i=-\infty}^{+\infty} \sum_{j=-\infty}^{+\infty} w(i,j)\, x(k+i, \ell+j)$

multi-channels

↓

image with channels

(RGB) → 3 channels

$\sigma\left(\sum_k W_{\ell k} * x_k + b_\ell\right)$ → the feature map $(d \times d)$ given by ℓ-th kernel

FCNN

FCNN: $\sigma\left(\sum_k W_{\ell k} \cdot X_k + b_\ell\right)$ ⇒ hidden unit h_ℓ

e) weight sharing

$w * x$ in 1D ⟷ $W \cdot x$

Matrix computation

$$= \begin{pmatrix} w_1 & w_2 & w_3 & 0 & 0 \\ 0 & w_1 & w_2 & w_3 & 0 \\ 0 & 0 & w_1 & w_2 & w_3 \\ w_3 & 0 & 0 & w_1 & w_2 \\ w_2 & w_3 & 0 & 0 & w_1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix}$$

Recall: FCNN invariance

(permutation invariance issues)

‖

☆ ⇒ CNN solves the Invariance Issue (permutation)

permutation variant !!!

Since convolution has its own structure of Matrix W

→ not so much freedom

Compare: FCNN : $Wx = \begin{pmatrix} w_{11} & & \\ & \ddots & \\ & & w_{55} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_5 \end{pmatrix}$

↓

all is free to choose

⇓

Permutation Invariant !

f) feature extractor → we can design different kernels/convolutions to achieve some function $\begin{cases} \text{blur} \\ \text{vertical edges} \\ \vdots \end{cases}$

Defn: Equi-variance (f is equi-var with Translation T)

$$f(T(x)) = T(f(x))$$

Conclusion : Conv. is Equi-variance with Shift - Translation

$\boxed{x \xrightarrow{\text{conv}} h}$ → Equi-variance

Defn: Invariance:

$$\boxed{f(T(x)) = f(x)}$$

[e.g.] $f(x) = \sum_{i,j}^{n} x(i,j)$ $\quad X \in \mathbb{R}^{n\times n}$

$$f(x) = \prod_{i,j}^{n} x(i,j)$$

☆

Conclusion: for some translation T, if $\begin{cases} f \to \text{invariant} \\ g \to \text{equi-variant} \end{cases}$

then $f \circ g$ is invariant

→ firstly apply equi-variant function g. → [e.g.] conv.

secondly apply in-variant function f.

$\boxed{f \circ g_1 \circ \cdots \circ g_j}$ → in-variant!

~~[e.g.] FCNN~~

FCNN is permutation invariant !!!

$$f_{FC}(x) = \sum_{i=1}^{m} V_i\, \sigma(W_i^T x + b_i)$$

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$$

$\tilde{x}_i = x_i + a = T(x_i)$

$$\mathcal{D}_s = \{(\tilde{x}_i, y_i)\}_{i=1}^{N}$$

$$f_{FC}(x) = \sum_{i=1}^{M} V_i\, \sigma(w_i^T x + b_i)$$

$$= \sum_{i=1}^{M} v_i \sigma(w_i^T \tilde{x} + \tilde{b}_i) \quad \underline{\text{where } \tilde{b}_i = b_i - w_i^T a}$$

$$= f_{FC}(\tilde{x})$$

$$= f_{FC}(T(x))$$

<u>Shrink Hyperthesis class</u> $\Longrightarrow \mathcal{H}_{conv} = \{ f : f(T(x)) = f(x) \}$

$\boxed{\text{invariant function class}}$

$\boxed{\text{Pooling Layer}} \rightarrow \boxed{\text{Max Pooling}} \rightarrow$ can shrink dimensionality

(with a stride p)