DSA5204 **LEC04**

Recap: FCNN

↓

hidden: $\begin{cases} h^{(i+1)} = \text{activation}(W^{(i+1)} h^{(i)} + b^{(i+1)}) \\ h^{(0)} = x \end{cases}$ $i = 0, \dots, T$

output: $\hat{y} = w^T h^{(T)} + b$

Observation. FCNN does not outperform other ML Algorithm

Today: CNN (Now also have Transformer)

(Weakness w.r.t Spatial / Temporal Data)

① Permutation Invariance of FCNN (not desirable)

↓

$\boxed{\text{Less Inductive Bias}}$ of spatial structure

Actually it is the Property of FCNN Hypothesis Space

$\Rightarrow$ $\boxed{\begin{array}{l} \forall f \in \mathcal{H}_{FCNN}, \text{ then } f_p \in \mathcal{H}_{FCNN} \\ \text{where } f_p(x_1, \dots, x_d) = f(x_{p(1)}, \dots, x_{p(d)}) \end{array}}$

Meaning: if we permutate the feature index,

then we can re-train a FCNN to behave as good

as the previous unpermutated data → on permutated data

(w.r.t Hypothesis Space)

Not desirable for sequential (spatial) data

Desirable for tabular feature (attributes)

② Conv. operation ("filter")

conv $\longrightarrow$ $x * w(t) = \int x(s)\, w(t-s)\, ds$

$$\boxed{w(x) = \begin{cases} e^{-x}, & x \geq 0 \\ 0, & o/w \end{cases}} \longrightarrow \text{Low-pass filter}$$

$\hookrightarrow$ Learnt by Data

Discrete Convolution

$$S(t) := x * w(t) = \sum_a x(a)\, w(t-a)$$
$$= \sum_a x(a)\, \tilde{w}(a-t) \qquad \boxed{\tilde{w}(t) = w(-t)}$$

a) Circular convolution $\rightarrow$ $x(-1) := x(n)$

one way of padding

3 2 5 1 4

$\downarrow$ pad

$\boxed{4\ 3\ 2\ 5\ 1\ 4\ 3}$

$\downarrow$ then do convolution

b) valid convolution $\rightarrow$ no padding

$\downarrow$

output size will decrease (we do not want)

☆

c) Zero-padding convolution $\Rightarrow$ maintain the output size

2D - conv

$$S(i,j) := \sum_{m,n} X(m,n)\, W(i-m, j-n)$$

Convolution $\Longleftrightarrow$ Cross-correlation

conv : $S(i,j) = \sum_{m,n} X(m,n)\, W(i-m, j-n)$

$= \sum_{m,n} X(m,n)\, \tilde{W}(m-i, n-j)$ → 往右下 shift (i,j) unit

cross-correlation : $S(i,j) = \sum_{m,n} X(m,n)\, \underbrace{W(m+i, n+j)}_{\text{shift index}}$

↳ implementation

(since NN just needs an element-wise calculation)

Conv is just one Linear Operation

$$w * x = \boxed{A(w)\, x}$$

↳ matrix multiplication

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

③ Meaning of conv (filter) ⇒ feature extractor

④ Property of conv → inductive bias

a) Sparsity → against over-fitting (only focus on neighbour input)

b) weight sharing → { extract features; save memory storage space }

Note : Receptive Field → when we stack MORE CNN Layers then it will enlarge Receptive Fields

3x3 conv → 3x3 conv → 3x3 conv

⇓

7x7 receptive field

c) translation - equivariance (of convolution operation)

→ $\boxed{g\text{ - equivariance}}$: $f(g(x)) = g(f(x))$

[e.g.] ReLu func is equivariant w.r.t scaling func

→ $\boxed{g\text{ - invariance}}$: $f(g(x)) = f(x)$

[e.g.]
- norm func. is in-variant w.r.t orthogonal projection
- max / $\boxed{\text{sum}}$ is in-variant w.r.t translation

special affine transformation

1. T_t : translation of signal x

   then we have $w * T_t(x) = T_t(w * x)$

Conclusion : ① for circular conv, it is translation - equivariance

② for other convolution (zero-padding etc.),

It is not strictly translation - equivariance for boundary.

But it is "almost" translation - equivariance

2. f_1, f_2 is g-equivariant. then $f_1 \circ f_2$ is also g-equivariance

Pf: $f_1 \circ f_2(g(x)) = f_1(g(f_2(x))) = g(f_1 \circ f_2(x))$

3. F is g-invariant. f is g-equivariant.

then $F \circ f$ is g-invariant

Pf: $F \circ f(g(x)) = F(g(f(x))) = F(f(x)) = F \circ f(x)$

☆ **Corollary**: $f_i \rightarrow$ Convolution layer i

$F \rightarrow$ pooling layer

then $F \circ (f_T \circ \cdots \circ f_1)$ is translation-invariant

$\downarrow$

$\boxed{\text{CNN stack}}$

⑤ Infintely Strong Prior

$\downarrow$

$\mathcal{H} := \{ f : f(T(x)) \approx f(x) \} \longrightarrow$ T-invariant Hypothesis

$\Downarrow$

Make sense for image classification

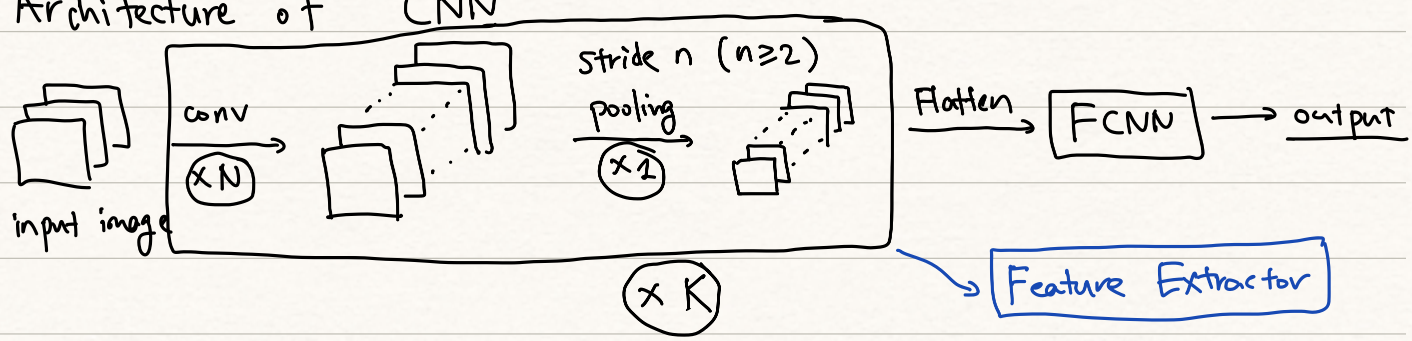
⑥ Pooling Layer $\longrightarrow$ "almost" $\boxed{\text{translation-invariant}}$

$\Downarrow$

$\boxed{\text{for small translation}}$

- type of pooling
  - average
  - max
  - ⋮
- stride

⑦ Architecture of CNN



⇒ This is why we can use FCNN in the last layer

since those inputs can be viewed as "attributes of original image"