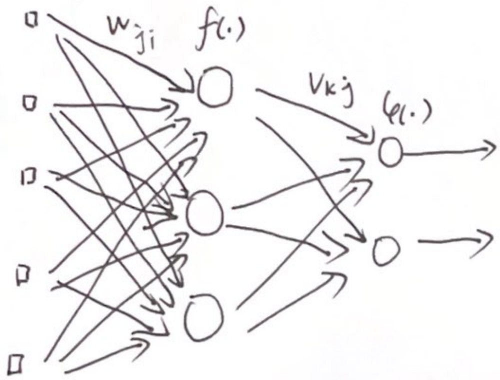
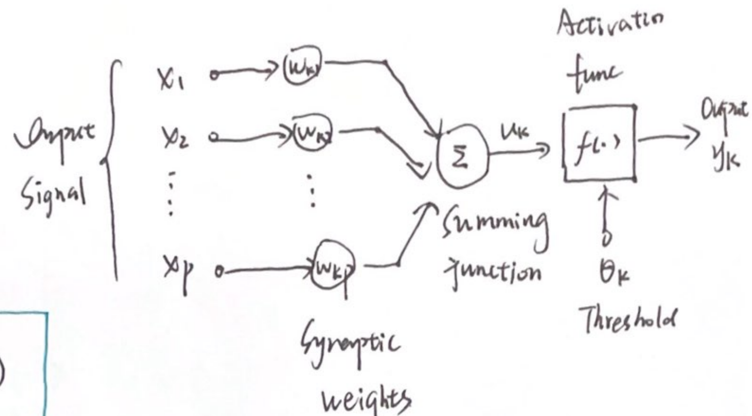
# EE7403 LEC14 Neural Networks and Deep Learning: From MLP to CNN

## 1. Neural Network and Deep CNN

① Network architecture of artificial neural networks (ANN), or simply, neural network.



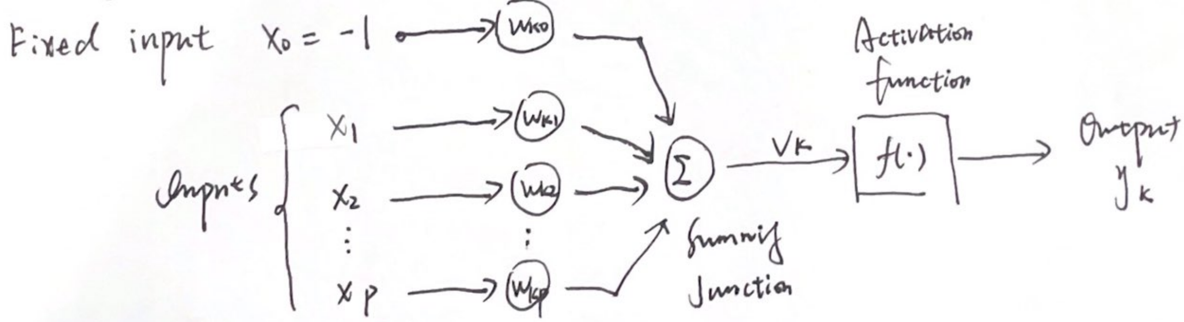




$$\boxed{u_k = \sum_{j=1}^{p} w_{kj} x_j \qquad y_k = f(u_k - \theta_k)}$$

$\theta_k$ is an external parameter, we can consider

$$x_0 = 1, \quad w_{k0} = -\theta_k$$

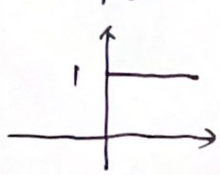
$$v_k = \sum_{j=0}^{p} w_{kj} x_j \qquad y_k = f(v_k)$$




② Types of Activation Functions

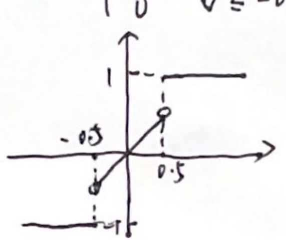
a. Binary function

$$f(v) = \begin{cases} 1, & v \geq 0 \\ 0, & v < 0 \end{cases}$$



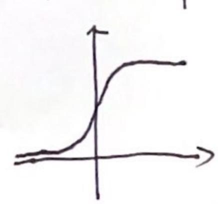

b. Piecewise-Linear

$$f(v) = \begin{cases} 1 & v \geq 0.5 \\ v & -0.5 < v < 0.5 \\ 0 & v \leq -0.5 \end{cases}$$




c. Sigmoid

$$f(v) = \frac{1}{1+\exp(-av)}$$



d. ReLu

$$f(v) = \max(0, v) = \begin{cases} v, & v \geq 0 \\ 0, & v < 0 \end{cases}$$

EE7403 LEC 14

2. ANN ① Multilayer Perceptron (MLP)

$$w_j = \begin{bmatrix} w_{j1} \\ w_{j2} \\ \vdots \\ w_{jn} \end{bmatrix} \quad v_k = \begin{bmatrix} v_{k1} \\ v_{k2} \\ \vdots \\ v_{kd} \end{bmatrix} \quad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad z = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_d \end{bmatrix} \quad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_c \end{bmatrix}$$

$z_j = w_j^T x$

$W = [w_1 \ w_2 \ \dots \ w_d] \Rightarrow z = W^T x$

$V = [v_1 \ v_2 \ \dots \ v_c]$ similarly $y = V^T z$

$z_j = f(\sum_{i=1}^{n} w_{ji} x_i) = f(w_j^T x)$

$y_k = f(\sum_{j=1}^{d} v_{kj} z_j) = f(v_k^T z)$

$= f[\sum_{j=1}^{d} v_{kj} f(\sum_{i=1}^{n} w_{ji} x_i)]$

$\Downarrow$

$y = f(V^T z) = f[V^T f(W^T x))$

If $f$ is a linear function: $y = f[V^T f(W^T x)] = V^T W^T x = (WV)^T x = \underline{U^T x}$ just same as single layer

where $U = WV$

To design a multi-layer network, the activation function should be nonlinear ☆

② Single-Layer Neural Network

$y = w^T x$ target output : $t$

error : $e = t - y = t - w^T x$

The mean square error : $J(w) = E\{e^2\} = E\{(t-y)^2\} = E\{(t - w^T x)^2\}$ $w = \frac{t}{x}$

least square method: quadratic : $= t^2 - 2w^T E\{xt\} + w^T E\{xx^T\} w$ only one minimum

$\frac{\partial E\{e\}}{\partial w} = \frac{\partial (t^2 - 2w^T E\{xt\} + w^T E\{xx^T\} w)}{\partial w} = 0 \Rightarrow E\{xx^T\} w = E\{xt\}$

$E\{xx^T\} \cong \frac{1}{q} \sum_{i=1}^{q} x_i x_i^T \quad E\{xt\} \cong \frac{1}{q} \sum_{i=1}^{q} x_i t_i$

Let $X = [x_1, x_2, \dots, x_q]$ $\quad t = [t_1 \ t_2 \ \dots \ t_q]$

$q E\{xx^T\} \cong \sum_{i=1}^{q} x_i x_i^T = XX^T \quad q \cdot E\{xt\} \cong \sum_{i=1}^{q} x_i t_i = Xt^T$

$E\{xx^T\} w = E\{xt\} \cong XX^T w = Xt^T \Rightarrow w = (XX^T)^{-1} X t^T$

EE 7403 LEC14

gradient descent method:

$w \Leftarrow w + \Delta w$

$\Delta w = -\eta \nabla J(w)$

$$\nabla J(w) = \frac{\partial J(w)}{\partial w} = \frac{\partial E\{(t-w^T x)^2\}}{\partial w} = -2E\{(t-w^T x)x\}$$

$$= -2E\{ex\} \triangleq -2e_i x_i$$

$w \Leftarrow w + \eta e_i x_i = w + \eta(t_i - y_i)x_i$

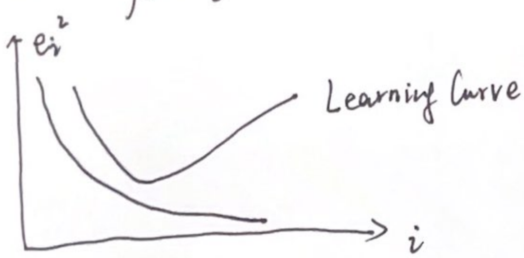
$\rightarrow w = (XX^T)^{-1} X t^T$

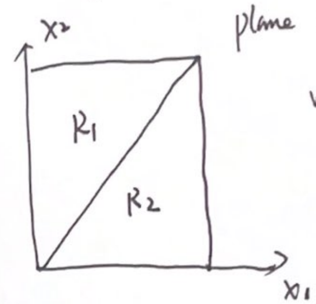
③ Learning Curve and Decision Boundary

$w \Leftarrow w + \eta(t_i - y_i)x_i$

decision boundary is a straight line or plane or hyper-plane

$w^T x = 0$




④ backpropagation

$y = f[V^T f(W^T x)]$

$J(w,v) = E(e^2) = E\{\|t-y\|^2\} = E\{\|t - f[V^T f(W^T x)]\|^2\}$

$$= \frac{1}{2}\sum_{k=1}^{c}[t_k - f(q_k)]^2 = \frac{1}{2}\sum_{k=1}^{c}\Big[t_k - f\Big(\sum_{k=1}^{d} v_{kj} z_j\Big)\Big]^2$$

$\Big(q_k = \sum_{j=1}^{d} v_{kj} z_j\Big)$

$w \Leftarrow w - \eta \nabla J(w)$

$v \Leftarrow v - \eta \nabla J(v)$

$$\frac{\partial J(w,v)}{\partial v_{kj}} = \frac{\partial J}{\partial q_k} \cdot \frac{\partial q_k}{\partial v_{kj}} = -(t_k - y_k)\cdot f'(q_k) z_j$$

Similarly, $J(w,v) = E\{\|t-y\|^2\} = \frac{1}{2}\sum_{k=1}^{c}[t_k - f(q_k)]^2 = \frac{1}{2}\sum_{k=1}^{c}\Big[t_k - f\Big(\sum_{j=1}^{d} v_{kj} z_j\Big)\Big]^2$

$$= \frac{1}{2}\sum_{k=1}^{c}\Big[t_k - f\Big[\sum_{j=1}^{d} v_{kj} f(q_j)\Big]\Big]^2 = \frac{1}{2}\sum_{k=1}^{c}\Big[t_k - f\Big[\sum_{j=1}^{d} v_{kj} f\Big(\sum_{i=1}^{n} w_{ji} x_i\Big)\Big]\Big]^2$$

$$\frac{\partial J(w,v)}{\partial w_{ji}} = \frac{\partial J}{\partial z_j} \cdot \frac{\partial z_j}{\partial q_j} \cdot \frac{\partial q_j}{\partial w_{ji}} = -\Big[\sum_{k=1}^{c}(t_k - y_k)\cdot f'(q_k) v_{kj}\Big] f'(q_j) x_i$$

$v_{kj} \Leftarrow v_{kj} - \eta \frac{\partial J(w,v)}{\partial v_{kj}}$ $\qquad w_{ji} \Leftarrow w_{ji} - \eta \frac{\partial J(w,v)}{\partial w_{ji}}$

$v_{kj} \Leftarrow v_{kj} + \eta(t_k - y_k)\cdot f'(q_k) z_j$

$w_{ji} \Leftarrow w_{ji} + \eta\Big[\sum_{k=1}^{c}(t_k - y_k) f'(q_k) v_{kj}\Big] f'(q_j) x_i$

EE7403 LEC 14

$$f(q) = \frac{1}{1+\exp(-aq)} = \frac{a\exp(-aq)}{[1+\exp(-aq)]^2} = a f(q)[1-f(q)]$$

propagation procedure:

Assume n training samples, no prior info.

(1) Initialization

(2) Presentation of training samples should be randomized from epoch to epoch

(3) Forward computation

(4) Backward computation

(5) Iteration. criterion can be the number of iterations or the rate of change of the average error small enough

Stop training at minimum of the error on the validation set.

× overfit

3. Problems of Machine Learning

Problems of local minima (under-fitting). $\boxed{\text{poor generalization (overfitting)}}$

① Why deep layer though 2 layers are enough?:

a. deep layer NN is easier to train. a shallow NN with many neurons is hard to optimize. fewer neurons per layer are often with better gradient flow.

b. deep layer needs fewer parameters. there're fewer neuron per layer. reuse parameters, making it more efficient.

c. learns hierarchical features Better generalization. deep layer structure like feature extractor.

② $\min_{\hat{f}} \hat{e}^2 = \min \sum_{\forall i} \| y_i - \hat{f}(x_i) \|_2^2 \Rightarrow 0$ , Can only find $y_i = \hat{f}(x_i)$, not $y = f(x)$.

how to make $\hat{f}(x) \Rightarrow f(x)$ ? Regulation.

EE7403 LEC14

4. Convolutional Neural Networks, CNN

① different from MLP.

1) network architecture

2) Convolution

3) simple nonlinear activation function ReLU.

4) pooling

5) deep. large number of layer.

Input size P×Q. C channels.

Output size P×Q. D channels.

$x_{i,j,k}$. $1 \le i \le P$, $1 \le j \le Q$, $1 \le k \le C$.

$y_{i,j,k}$, $1 \le i \le P$. $1 \le j \le Q$. $1 \le k \le D$

$$y_{i,j,k} = \sum_{n=1}^{C} \sum_{m=-1}^{1} \sum_{l=-1}^{1} w_{l,m,n,k} \cdot x_{i-l,j-m,n} + b_k \quad 1 \le i \le P.\ 1 \le j \le Q.\ 1 \le k \le D$$

input channel

CNN capture the local feature. includes a bias term

Along channel dim, all inputs of different channels are all fully connected.

② Characteristics

1) Local Perceptive Fields

2) Shared Weights (gives translation invariance. num of parameters↓)

3) Full channel connected.

4) Bias Term

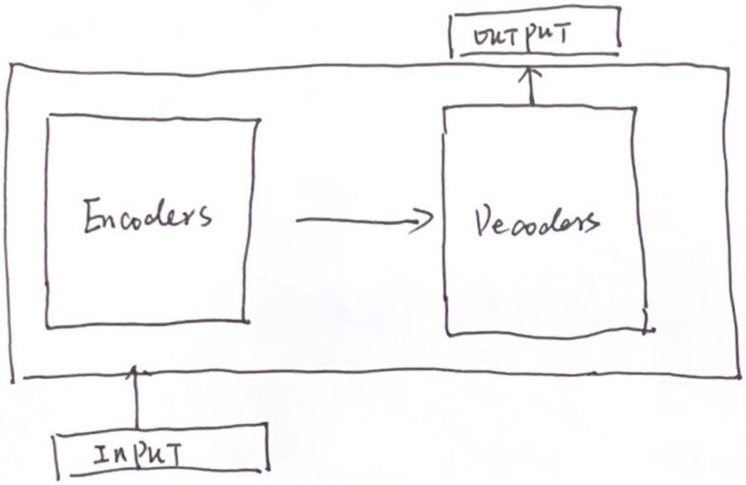
5) Hierarchy Features

A CNN is essentially a regularized version of MLP.

# EE7403 LEC 15 Deep Learning: From CNN to Transformer.

Attention is all you need.

Transformer captures the relationships between each word/token in a sequence with every other word/token.




The Transformer.

Encoders + Decoders

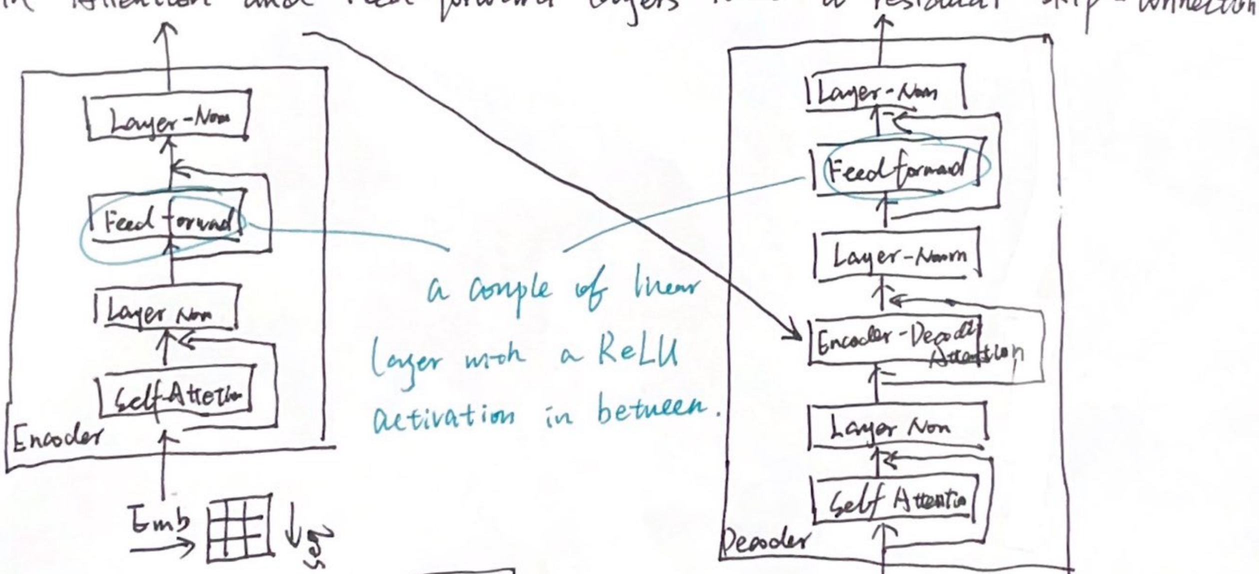
1. Embedding

Embed each input token into a feature vector.

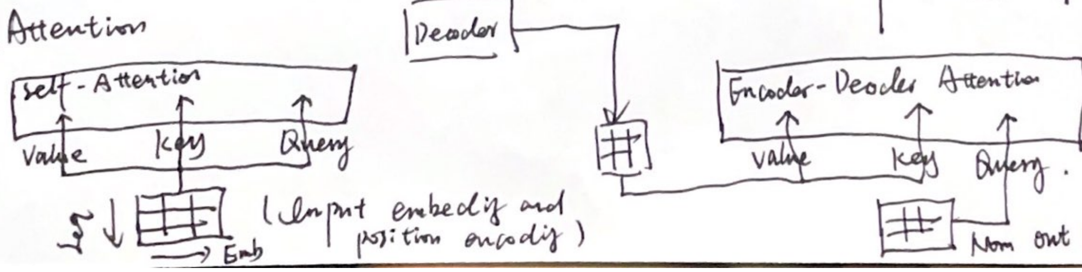
1 token is a vector

2. Encoder & Decoder

Both Attention and Feed-forward layers have a residual skip-connection




3. Attention



EE7403 LEC15

Three copies of each word/token are generated for self-attention by linear projection.

trainable $W_q . W_k . W_v$

Let $X = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}$, $Q = \begin{pmatrix} Q_1 \\ Q_2 \\ Q_3 \\ Q_4 \end{pmatrix}$, $K = \begin{pmatrix} K_1 \\ K_2 \\ K_3 \\ K_4 \end{pmatrix}$, $V = \begin{pmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \end{pmatrix}$

Then: $Q = XW_q$, $K = XW_K$, $V = XW_V$ each 'row' of these matrices correspond to one token in the source sequence

① $R = QK^T$ the relevance between words.

Dot product generates similarity between words.

used as factors

② $Z = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$ $Q . K . V . Z \in R^{n \times k}$

dimension k

Attention: $W = QK^T$

Output $Z = WV$ weighted sum of all input token. Capture global infor.

X learnt from training data but generated by specific test input.

Attention itself is learnable.

4. Learnable feed forward networks

$Y = h(h(h(XW_1)W_2)W_3)$

$(N \times K)(K \times K) \Rightarrow (N \times K)$

all tokens in inputs/outputs share same learnable parameters

each single learnable parameter go through all points