P4. Additivity

$\underline{X} = \begin{bmatrix} \underline{X}_1 \\ \underline{X}_2 \end{bmatrix}$, $\underline{X}_1$ and $\underline{X}_2$ are independent, i.e.

$$p(\underline{x}) = p_1(\underline{x}_1)\, p_2(\underline{x}_2)$$

$$q(\underline{x}) = q_1(\underline{x}_1)\, q_2(\underline{x}_2)$$

The $D_{KL}(p\|q) = D_{KL}(p_1\|q_1) + D_{KL}(p_2\|q_2)$

P5. Relation to cross entropy

Entropy $H(X) = H(p) = -E(\ln(p(x)))$

$$= -\sum_{i=1}^{M} p_i \ln(p_i) \geq 0$$

$$H(X) = H(p) = -E(\ln(p(x)))$$

$$= -\int_{-\infty}^{\infty} p(x) \ln(p(x))\, dx \geq 0$$

Cross Entropy (CE)

$$H(X,Y) = H(p,q) = -E_{X\sim p} \ln(q(x))$$

$$= -\int_{-\infty}^{\infty} p(x) \ln(q(x))\, dx \geq 0$$

$$D_{KL}(p\|q) = \int p \ln\frac{p}{q}\, d\underline{x}$$

$$= \int p \ln p\, d\underline{x} - \int p \ln q\, d\underline{x}$$

$$= -H(p) + H(p,q)$$

For a given $p(\underline{x})$:

$H(p)$ fixed, independent of $q(\underline{x})$,

$$\min_q D_{KL}(p\|q) \Longleftrightarrow \min_q H(p,q)$$

Not the case for the backward KL Divergence $D_{KL}(q\|p)$

Sometimes, the symmetric Jensen-Shannon divergence is used :

$$D_{JS}(p\|q) = \frac{1}{2}\left[D_{KL}\left(p\,\Big\|\,\frac{p+q}{2}\right) + D_{KL}\left(q\,\Big\|\,\frac{p+q}{2}\right)\right]$$
$$= D_{JS}(q\|p)$$

3.4 Probabilistic framework for supervised learning

Valid for
- Statistical processing and ML/DL
- regression and classification.

$p(\underline{x}/\underline{y})$: signal model

$$p(\underline{y}|\underline{x}) = p(\underline{x}/\underline{y}) \cdot \frac{p(\underline{y})}{p(\underline{x})}$$

Training in ML : model / calculate $p(\underline{y}|\underline{x}) \;\forall\; (\underline{x},\underline{y})$ from D_{train}

Inference in ML : Make a prediction about $\underline{y}$ for a given $\underline{x}$ based on the learnt $p(\underline{y}/\underline{x})$

Supervised learning

- $p(\underline{x}|\underline{y}), p(\underline{y})$ unknown $\rightarrow$ $p(\underline{y}|\underline{x})$ unknown
- approximate $p(\underline{y}|\underline{x})$ by a parameter posterior

  $q(\underline{y}|\underline{x};\underline{\theta})$ provided by a DNN with parameter vector $\underline{\theta}$

  $$\underline{x} \rightarrow \boxed{\text{DNN}, \underline{\theta}}$$
  $$q(\underline{y}|\underline{x};\underline{\theta})$$

  - function $q(\underline{y}|\underline{x};\underline{\theta})$ known $\longrightarrow$ given by DNN architecture
  - $\underline{\theta}$ unknown $\rightarrow$ DNN coefficients $\rightarrow$ learnt from data

- Learn $\underline{\theta}$ from a training set $D_{train} = \{ \underline{x}(n), \underline{y}(n) \mid 1 \leq n \leq N \}$

Learning criteria

$$\min_{\underline{\theta}} D_{KL}\left(p(\underline{x},\underline{y}) \,||\, q(\underline{x},\underline{y};\underline{\theta})\right) \qquad p(\,) \text{ fixed}$$

$$\parallel$$

$$\min_{\theta} H\left(p(\underline{x},\underline{y}), q(\underline{x},\underline{y};\underline{\theta})\right)$$

since $q(\underline{x},\underline{y};\underline{\theta}) = \underbrace{q(\underline{y}|\underline{x};\underline{\theta})}_{\text{DNN}} \cdot \underbrace{q(\underline{x})}_{\text{fixed}}$

$$\min_{\theta} H\left(p(\underline{x},\underline{y}), q(\underline{y}|\underline{x};\underline{\theta}) \cdot q(\underline{x})\right)$$

$$= -\int p(\underline{x},\underline{y}) \cdot \ln q(\underline{x},\underline{y};\underline{\theta})\, d\underline{x}\, d\underline{y}$$

$$= -\int p(\underline{x},\underline{y}) \ln q(\underline{y}|\underline{x};\underline{\theta})\, d\underline{x}\, d\underline{y} \;-\; \underbrace{\int p(\underline{x},\underline{y}) \ln q(\underline{y})\, d\underline{x}\, d\underline{y}}_{\substack{\text{independent of } \theta \\ \text{fixed}}}$$

$$= \underbrace{\int p(\underline{x},\underline{y}) \underbrace{\left(-\ln q(\underline{y}|\underline{x};\underline{\theta})\right)}_{\text{loss}} dx\, dy}_{\text{average loss, Bayesian risk}} + \text{const}$$

In practice, $p(\underline{x},\underline{y})$ approximated by emperical distribution

$$\hat{p}(\underline{x},\underline{y}) = \frac{1}{N}\sum_{n=1}^{N} \delta\left(\underline{x}-\underline{x}(n), \underline{y}-\underline{y}(n)\right)$$

3.2.3 Sampling property of $\delta(\,)$

$$\Rightarrow \min_{\underline{\theta}} H(\hat{p}, q) = \underset{\substack{\uparrow \\ \text{we ignore this}}}{\text{constant}} + \underbrace{\frac{1}{N}\sum_{n=1}^{N} \underbrace{-\ln q\left(\underline{y}(n) \mid \underline{x}(n), \underline{\theta}\right)}_{\text{loss for } (\underline{x}(n), \underline{y}(n))}}_{\text{cost function } L}$$

Role of DNN:

- approximate true posterior $p(\underline{y}|\underline{x})$ by $q(\underline{y}|\underline{x};\underline{\theta})$
- learn $\underline{\theta}$ from D_{train}

  see more details in Ch 4.6

$-\ln q()$: negative log-likelihood (NLL) loss.

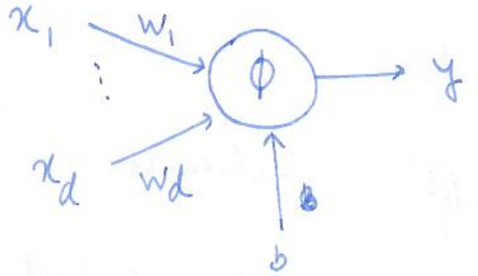
4. Dense neural network

A general model for estimating $q(\underline{y} \mid \underline{x}; \theta)$

- should be able to approximate non-linear mapping
- suitable for both regression and classification tasks

Artificial NN: mimic biological NN (brain)

4.1 Neuron



$\underline{x} = [x_i] \in \mathbb{R}^d$: input vector or image, tensor.

$\underline{w} = [w_i] \in \mathbb{R}^d$: weights

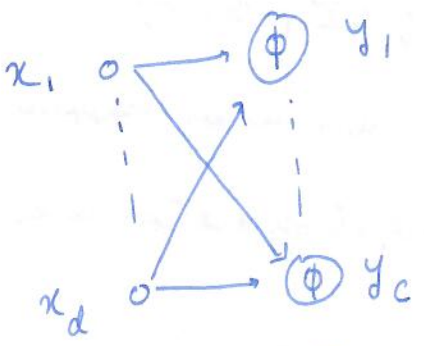
$b \in \mathbb{R}$: bias

$a = \underline{w}^T \underline{x} + b \in \mathbb{R}$: activation, an affine f^n of the input $\underline{x}$

$\phi() : \mathbb{R} \to \mathbb{R}$: activation function

$y = \phi(a) = \phi(\underline{w}^T \underline{x} + b)$: output

4.2 Layer



c neurons $\rightarrow$ all are connected to all

$\underline{x} \in \mathbb{R}^d$: input

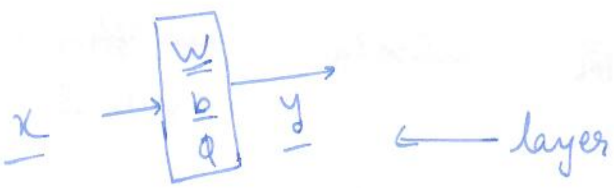
$\underline{\underline{W}} = [w_{ij}] \in \mathbb{R}^{c \times d}$: weights

$b = [b_i] \in \mathbb{R}^c$: bias

$\underline{a} = \underline{\underline{W}}^T \underline{x} + \underline{b} \in \mathbb{R}^c$: activation

$\underline{y} = [y_i] = \phi(\underline{a}) \in \mathbb{R}^c$: output

$\phi(\,)$: activation function



2 meanings of $\phi()$

*) one neuron : $\phi(a) : \mathbb{R} \longrightarrow \mathbb{R}$

*) one layer : $\phi(\underline{a}) : \mathbb{R}^c \longrightarrow \mathbb{R}^c$

•) element wise : $\phi(\underline{a}) = \begin{bmatrix} \phi(a_1) \\ \vdots \\ \phi(a_c) \end{bmatrix}$

•) or not : $\phi(\underline{a}) = \begin{bmatrix} \phi_1(\underline{a}) \\ \vdots \\ \phi_1(\underline{a}_c) \end{bmatrix}$

see Ch 4.4

<u>Comments:</u>

- no inter connections b/w neurons in the same layer
- fully connected, dense layer :
  each input x_j connected to each input y_i
  $\Rightarrow$ cd weights w_{ij} and c bias values b_i
  i.e $c(d+1)$ parameters.

4.3 Feed forward neural network.

A cascade of dense layer.