EE5904 lec5

Mathematics Neuron → Mc & Pitts, 1943

↓

perceptron → linearly separable → Rosenblatt · 1958 → Perceptron Convergence thm

↓

Regression → linear

↓

MLP → not need of linearly separable → Rumelhart

→ magic: map input to feature

↓

hidden neuron 1

Learning Algorithm: Steepest Descent

$$\rightarrow w_{ji}^{(s)}(k+1) = w_{ji}^{(s)}(k) + \eta^{(s)} \delta_j^{(s)} x_{out\,i}^{(s-1)}$$

$\delta_j^{(s)}$: output error; $x_{out\,i}^{(s-1)}$: input signal

↓

Back propagation → easy to compute!

How to design?

① # of hidden layer { x-x-x ; x-x-x-x

② # of hidden neuron

③ activation $f^{(s)}$ (hidden)

④ activation f⃗ (out) { regression / PR

⑤ pre-process → input ⇒ Normalization

⑥ { Batch → try first { fast! / But local minima
Sequential → { slow / better results!

⑦ overfitting { minimal structure / regularization

MLP { regression / PR

↓

A way of approximation! ⇔ Real model is too complicated!

↑ universal approximator

desired output

{ Recognition → users-defined
Regression → observation

WEAKNESS

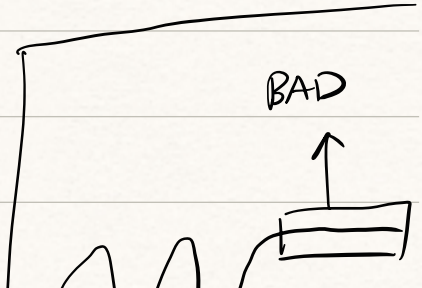
just approximation ↔ cannot explain! (no intuitive explanation)

↕

Black box

WEAKNESS 2

only focus on the input area



other space is flat line!

it can learn
But can not know
something new

generalization

Change activation f

our observation!

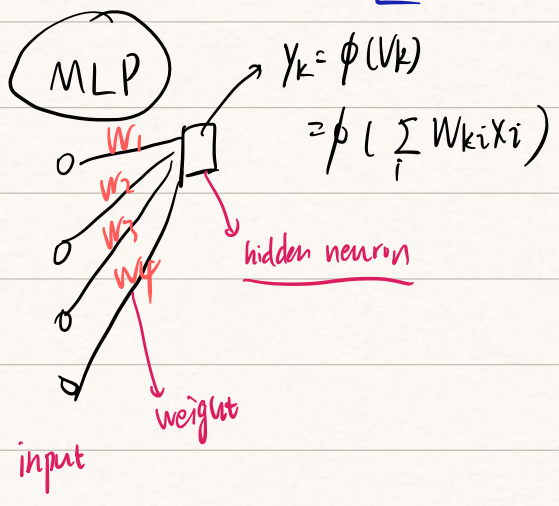
RBF

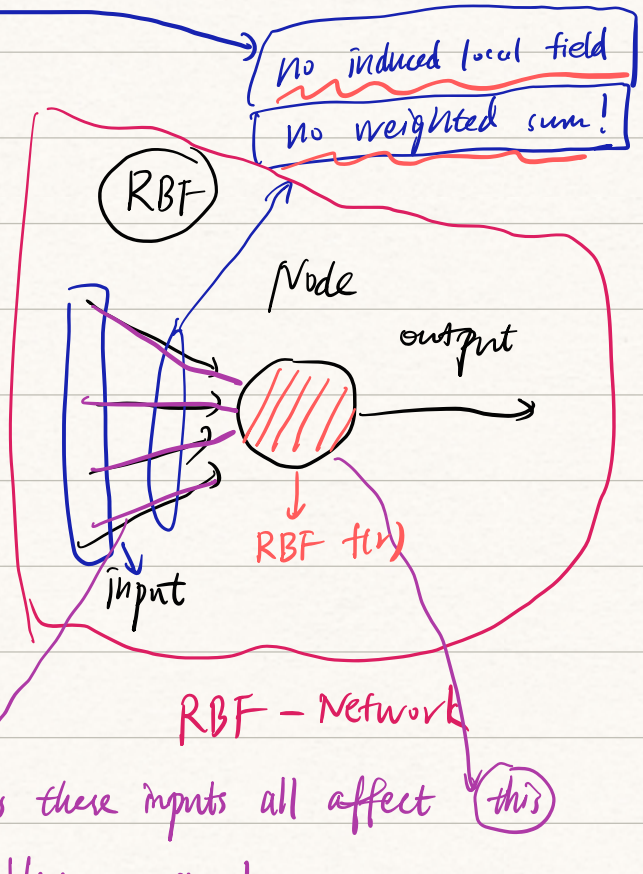
Radial Basis Function ⟷ function approximation
(RBF)

RBF: f only depends on the radial distance

r

different





input vector

Process

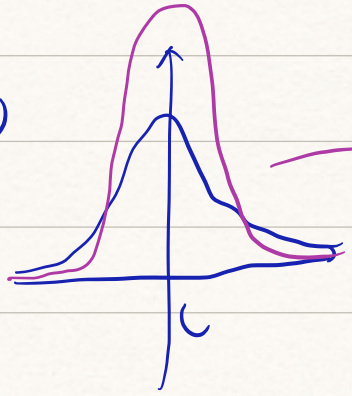
prototype vector

(node)

① $\varphi(\underline{x}) = \varphi(\|\underline{x}-\underline{c}\|) = \varphi(r)$

distance

② $\varphi(\cdot)$



→ Gaussian-like function

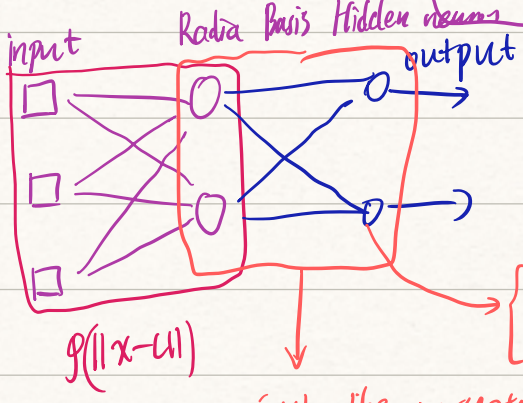
our goal: if $\underline{x}$ match $\underline{c}$

$\updownarrow$

$\|x-c\|$ small

then $\varphi(\|x-c\|)$ BIG!

Note

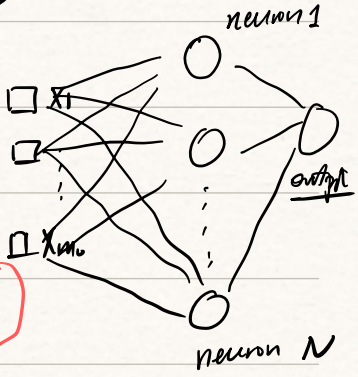


{ # of hidden neuron : # of data samples (N)

center : $\underline{x_i}$

$\underline{X}(1), \ldots, \underline{X}(N)$

$\underline{X}(i) \in \mathbb{R}^{m_0}$



$$f(x) = \sum_{i=1}^{N} w_i \, \phi(\|\underline{x} - \underline{x_i}\|$$

$$\Rightarrow f(\underline{x_j}) = \sum_{i=1}^{N} w_i \, \phi(\|\underline{x_j} - \underline{x_i}\|) = d_j$$

$\frac{N^2+N}{2}$

$$\Leftrightarrow \begin{pmatrix} d_1 \\ \vdots \\ d_N \end{pmatrix} = \begin{pmatrix} \phi_{11} & \cdots & \phi_{1N} \\ & \vdots & \\ \phi_{N1} & \cdots & \phi_{NN} \end{pmatrix} \begin{pmatrix} w_1 \\ \vdots \\ w_N \end{pmatrix}$$

$\phi_{ji} = \phi(\|x_j - x_i\|)$

$\varphi_{ji} = \varphi(\|x_j - x_i\|)$

$d = \Phi W \Rightarrow W = \Phi^{-1} d$ (if) Φ is invertible

However, if $\Phi \rightarrow$ nonivertible ?

thm: if $x(1), \ldots, x(n)$ in $\mathbb{R}^{m_0}$

distinct points

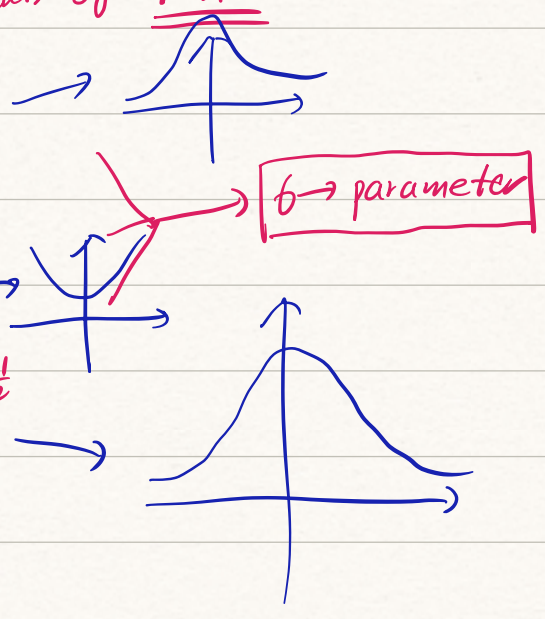
then $\varphi_{ji} := \varphi(\|x(j) - x(i)\|)$ non-singular

↳ for a class of RBF

localized ← ① $\varphi(r) = \exp(-\frac{r^2}{2b^2})$

non-localized ← ② $\varphi(r) = (r^2 + b^2)^{\frac{1}{2}}$

③ $\varphi(r) = (r^2 + b^2)^{-\frac{1}{2}}$



the algorithm is quite easy!

↳ Φ, $d \Rightarrow W = \Phi^{-1} d$

Limitation!

↓

over-fitting! ⟷ noisy data!

Strategy: shrink the hidden neuron!

How to improve?

↓

- not fix the # of hidden units
- not fix the center of RBF ← learn!
- spread parameter? ← learn

we can also add BIAS term W_0

Gaussian RBF

$$\varphi_i(\|x-\mu_i\|) = \exp\left(-\frac{\|x-\mu_i\|^2}{2\sigma_i^2}\right)$$

μ_i, σ_i, w_i → How to find?

↓

$$y(x) = \sum_{i=1}^{M} w_i \varphi_i(\|x-\mu_i\|)$$

And it also has Universal Approximation Theorem for RBFN

↕

just like MLP

only existence

we still consider $E(w) = \sum_{i=1}^{N} e(i)^2 = \sum_{i=1}^{N} (d(i) - y(i))^2$

① Steepest descent

↑ calculate $\frac{\partial E(w)}{\partial \mu_i}$ $\frac{\partial E(w)}{\partial \sigma_i}$ $\frac{\partial E(w)}{\partial w_i}$

↑

we don't have BP Alg. here!

↓

no efficient algorithm to calculate these!

② our method

↓

分 BP 和 LS train

① μ_i & σ_i

② $w_1 \cdots w_M$, w_0

according to μ_i & σ_i

Algorithm:

Stage 1: { para of hidden layer
- M → # of hidden neuron ⇒ unsupervised
- μ_i
- σ_i

⇗ weights

para between hidden & output

Stage 2 { $w_1 \cdots w_M$, w_0 ⇒ supervised learning

Firstly, stage 2 fix $\begin{cases} M \\ \mu_1 \cdots \mu_M \\ G_1 \cdots G_M \end{cases}$

$$y(x_j) = \sum_{i=0}^{M} w_i \varphi_i(x_j) = d_j$$

$$\Rightarrow \underbrace{\begin{pmatrix} \phi_0(x_1) & \cdots & \phi_M(x_1) \\ \phi_0(x_N) & \cdots & \phi_M(x_N) \end{pmatrix}}_{N \times (M+1)} \begin{pmatrix} w_0 \\ \vdots \\ w_M \end{pmatrix} = \begin{pmatrix} d_1 \\ \vdots \\ d_N \end{pmatrix}$$

$$\Rightarrow \phi w = d$$

样本大 ~~hidden neuron~~ hidden neuron ↯

if $N > M+1$, → we can not find $\hat{f}$ pass through all data samples!

↓

MSE

$$E(w) = (\phi w - d)^T (\phi w - d)$$

$$\overset{\text{const}}{=} w^T \phi^T \phi w - 2 d^T \phi w$$

optimality condition: $\nabla E(w) = 0$

$$\Rightarrow \nabla E(w) = 2\phi^T \phi w - 2\phi^T d = 0$$

$$\Leftrightarrow \boxed{w = (\phi^T \phi)^{-1} \phi^T d}$$

LLS solution

Stage 1.

① fix centers randomly

② clustering algorithm → k-means

---

supervised learning of RBFN

↓

well result

great cost

↓

like MLP

$$E = \frac{1}{2}\sum_{i=1}^{N}(e_i)^2$$

$$g_n = \exp\left(-\frac{\|\underline{x}_i - \underline{\mu}_n\|^2}{2\sigma_n^2}\right)$$

$$e_i = d_i - y(\underline{x}_i) = d_i - \sum_{n=1}^{M} w_n g_n(\underline{x}_i; \underline{\mu}_n, \sigma_n)$$

Gradient Descent → second order method is also OK

$$\begin{cases} \Delta w_i = -\eta_w \dfrac{\partial E}{\partial w_i} \\ \Delta \underline{\mu}_i = -\eta_{\underline{\mu}} \dfrac{\partial E}{\partial \underline{\mu}_i} \\ \Delta \sigma_i = -\eta_\sigma \dfrac{\partial E}{\partial \sigma_i} \end{cases}$$

Q: How to choose # of hidden neurons?

↓

1. minimal structure

2. regularization

Goal → overcome overfitting

↓

restrict the value of weight

$$F = E_D + \lambda E_W$$

$\frac{1}{2}\sum_j e_j^2$ — penalty on complexity (smoothness)

Question: How to measure the smoothness?

↓

derivatives (gradient)!

$$f(x) = \sum_i w_i \varphi(\|x - \mu_i\|) \qquad \varphi_i(x) = \exp\left(-\frac{\|x-\mu_i\|^2}{2\sigma_i^2}\right)$$

$$\frac{\partial f}{\partial x} = \sum_{i=0}^{M} w_i \frac{\partial \varphi(\|x-\mu_i\|)}{\partial x}$$

w_i → influence the gradient!

w_i big ⇒ less smoothness!

- a measurement of SMOOTHNESS
- the motivation of minimizing weights!

$$\frac{\partial e^T e}{\partial w} = \frac{\partial e^T e}{\partial e}\frac{\partial e}{\partial w} = 2e^T \cdot \Phi$$

$$F(w) = \frac{1}{2}\sum e_i^2 + \frac{1}{2}\lambda w^T w = \frac{1}{2}(\Phi w - d)^T(\Phi w - d) - \frac{1}{2}\lambda w^T w$$

$$= 2(\Phi w - d)^T \Phi$$

$$= 2w^T\Phi^T\Phi - 2d^T\Phi$$

$$\Phi = \begin{pmatrix} \varphi_0(x_1) & \cdots & \varphi_M(x_1) \\ \vdots & & \vdots \end{pmatrix} \qquad W = \begin{pmatrix} w_0 \\ \vdots \end{pmatrix}$$

$\varphi_0(x_N) \cdots \varphi_M(x_N)$ $\quad (w_M)$

$\nabla F(w) = \Phi^T \Phi w - \Phi^T d + \lambda w$

$= (\Phi^T \Phi + \lambda I) w - \Phi^T d$

$\Rightarrow w = (\Phi^T \Phi + \lambda I)^{-1} \Phi^T d$

$\frac{\partial F}{\partial w} = \left(\frac{\partial e^T e}{\partial w} + \frac{\partial \lambda w^T w}{\partial w} \right) \frac{1}{2}$

$= 2 e^T \Phi + 2 \lambda w^T$

- MLP – Regularization → H/3RD
- RBFN – Reg → EASY

---

for classification

↓

easy to design output

↓

k-class ⇒ k-output $\underbrace{\begin{bmatrix}1\\0\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}\cdots\begin{bmatrix}0\\1\end{bmatrix}}_{k}$ $k\times 1$

Note:

All output neurons → linear neuron

All power lies in hidden neurons → RBF

↓

input pattern → (nonlinear map) → hidden units

↓ usually low-dim ↓ usually high-dim

linear map

$\begin{pmatrix} \varphi_0(x_N) & \cdots & \varphi_M(x_N) \end{pmatrix}$

Cover's theorem

output

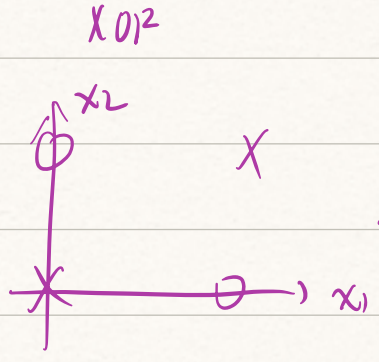
A complex pattern classification cast in a high-dim space. (nonlinearly)
then it is more likely to be linearly separable than in low-dim space!

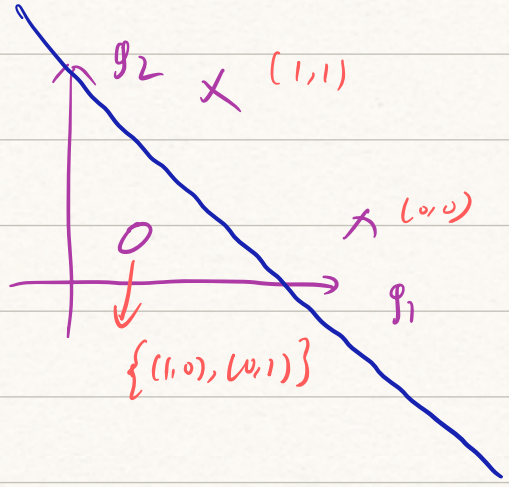
then we can solve it easily!

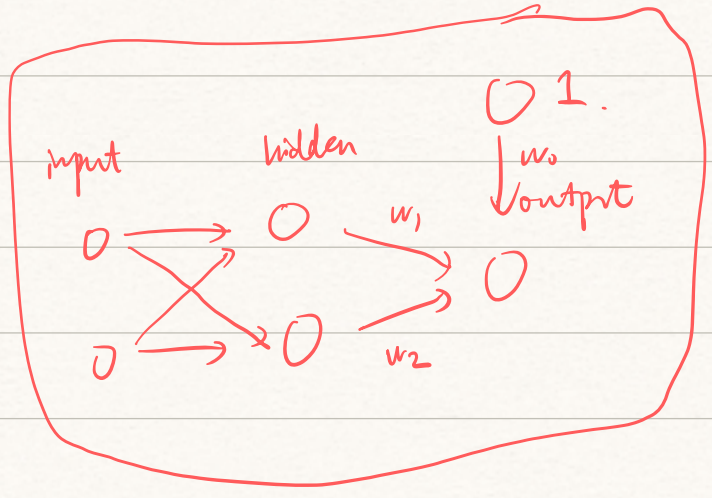
$\mu_1 = (0,0)$
$\mu_2 = (1,1)$

$g_1(\underline{x}) = \exp(-\|x-\mu_1\|^2)$

$g_2(\underline{x}) = \exp(-\|x-\mu_2\|^2)$







RBFN vs MLP

Similarity
① nonlinear feed-forward network
② universal approximators
③ in similar application areas

difference: 1. RBFN cannot have 2 (or more) hidden neurons

MLP ≥2 is OKAY!

② argument of activation f^n $\phi(\cdot)$

2. RBFN hidden neuron ✗ output neuron — very differently.

MLP hidden neuron ≈ output neuron

① { 流程相似
activation f° 可能稍有不同

distance between input & center → "weight"

induced local field (weighted sum)

DRAMATIC Difference

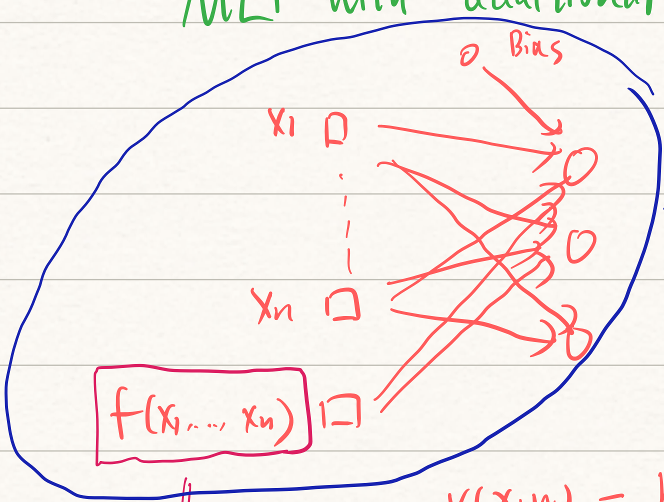
3. training differently

MLP → BP

RBFN → { 2-stage / GD

faster

MLP with additional input — MLP → RBFN



$\sum_{i=1}^{n} x_i^2$

→ CBP { Both MLP & RBFN

↓ Circular BP

$$v(x,w) = b + \sum_{i=1}^{n} w_i x_i + w_{n+1} f(x)$$

$x_1^2 + \dots + x_n^2$

one input case

$b + wx + \boxed{(x-c)^2 + D}$

$\approx (x-a)^2 := r^2 \qquad r = \|x-a\|$

$= g(\|x-a\|)$

$$V(\underline{x}, \underline{w}) = w_{n+1} \sum_{i=1}^{n} \left(x_i + \frac{w_i}{2w_{n+1}}\right)^2 + **$$

$$\boxed{V(\underline{x}) = g(\|\underline{x} - \underline{c}\|^2 + \theta)}$$

where $g = w_{n+1} \qquad \underline{c} = \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix} \quad c_i = -\frac{w_i}{2w_{n+1}}$

$$\theta = \frac{b}{w_{n+1}} - \sum_{i=1}^{n} \frac{w_i^2}{4w_{n+1}^2}$$

let $r = \|x-c\|$

RBF $\leftarrow$ $V(x) = g(\|x-c\|^2 + \theta) = gr^2 + a$ $\rightarrow$ just find one homogenous f^n

$\Downarrow$

$\phi(V(x)) \rightarrow$ RBF