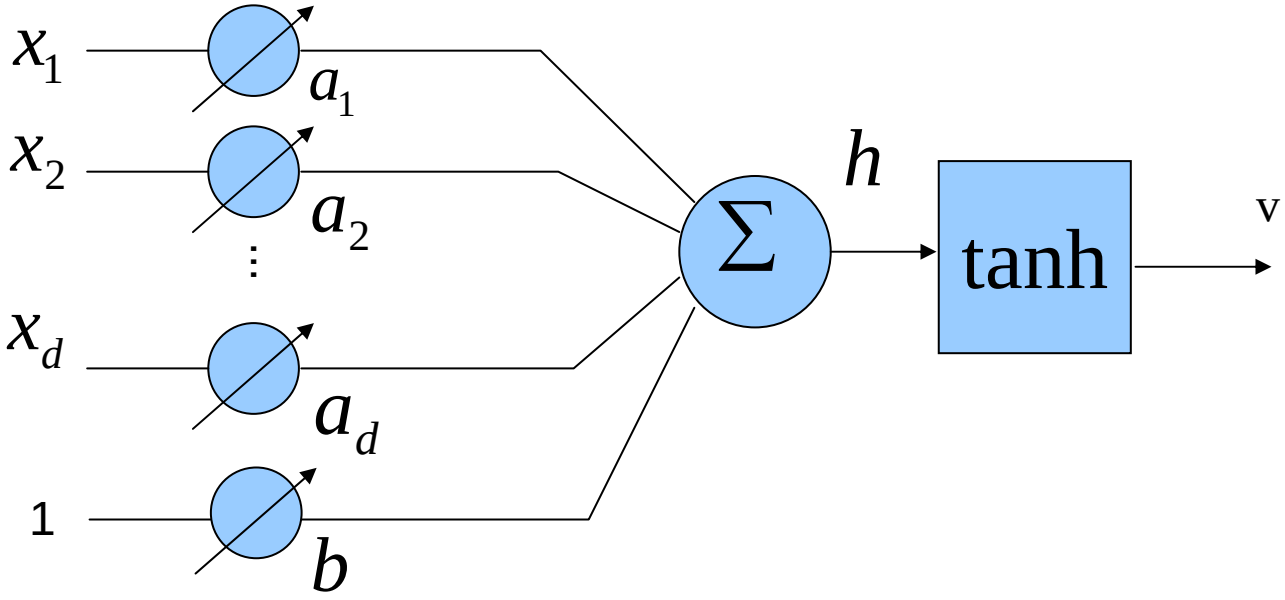
Levenberg-Marquardt Learning for RBF neural networks

Perceptrons

- Rosenblatt (1962), Widrow (1962)
- Post-tanh (sigmoid-like) projection

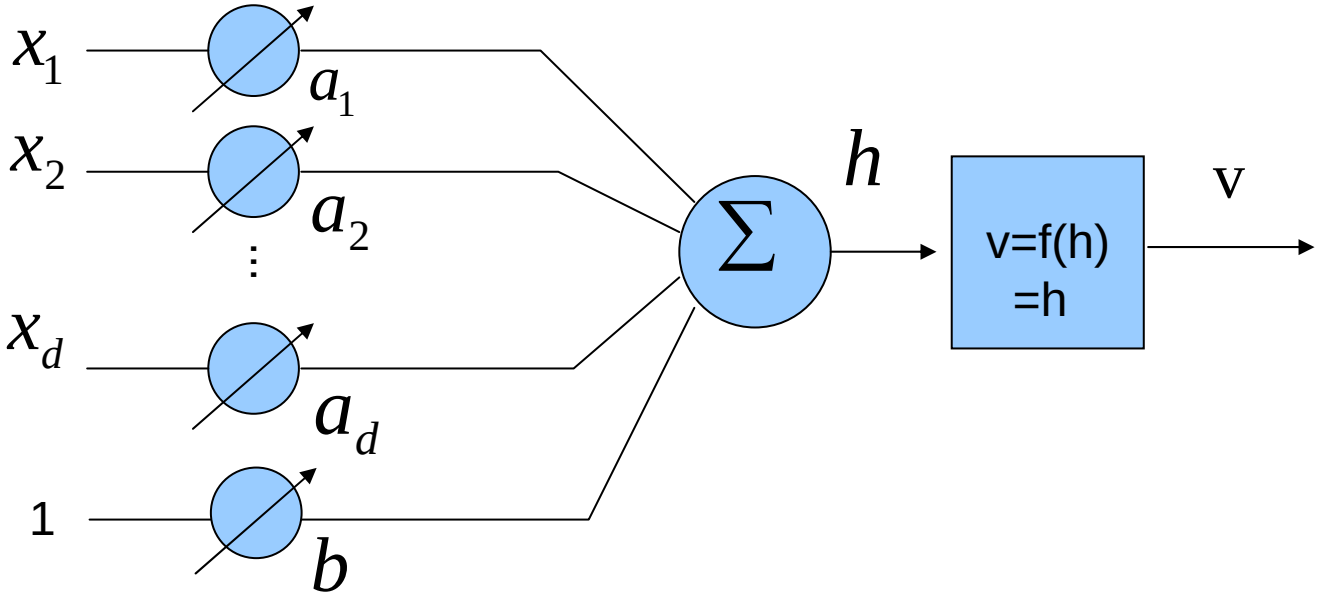
$$v = \tanh(h = a_1 x_1 + a_2 x_2 + \ldots + a_d x_d + b)$$



Linear projection

- Reduce to linear projection when tanh is removed

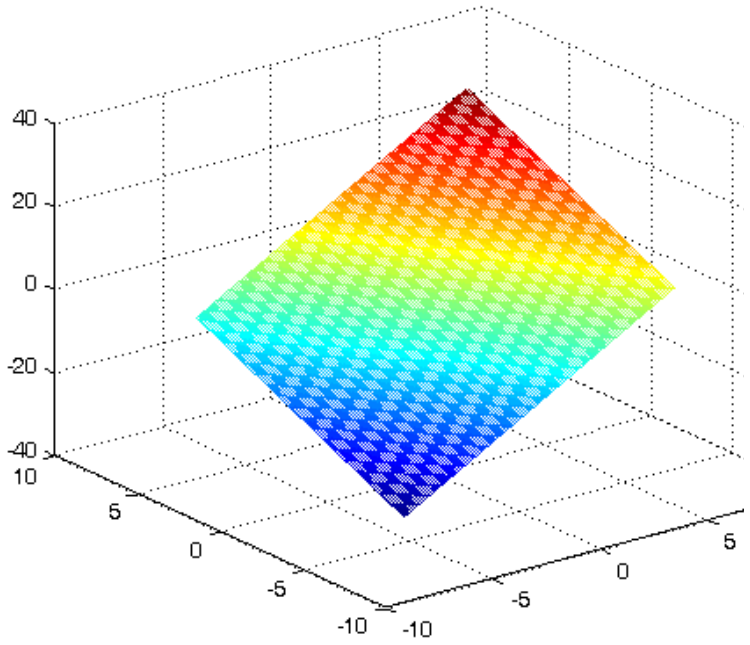
$$v = h = a_1 x_1 + a_2 x_2 + \ldots + a_d x_d + b$$

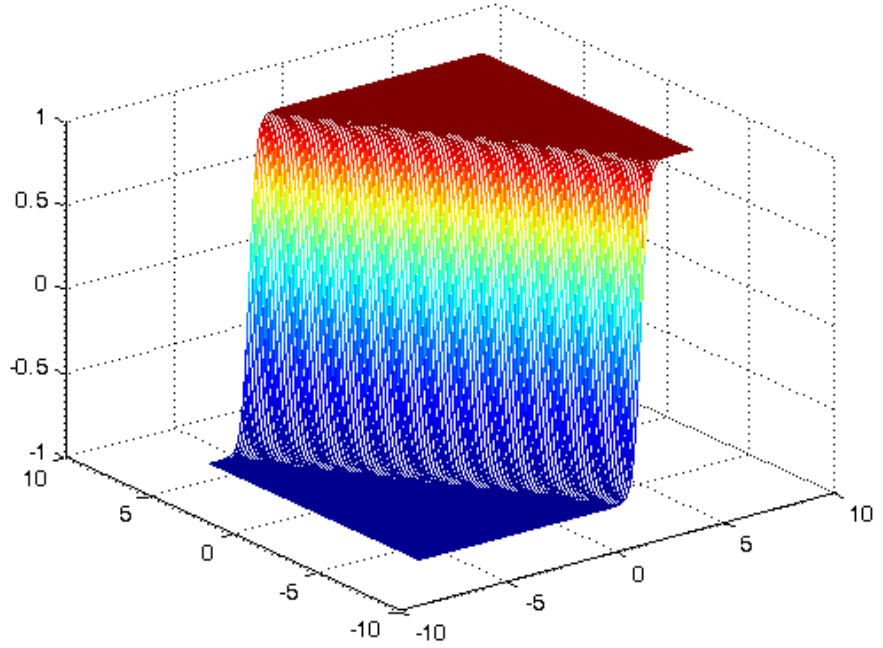


Linear projection vs post-tanh projection

d=2

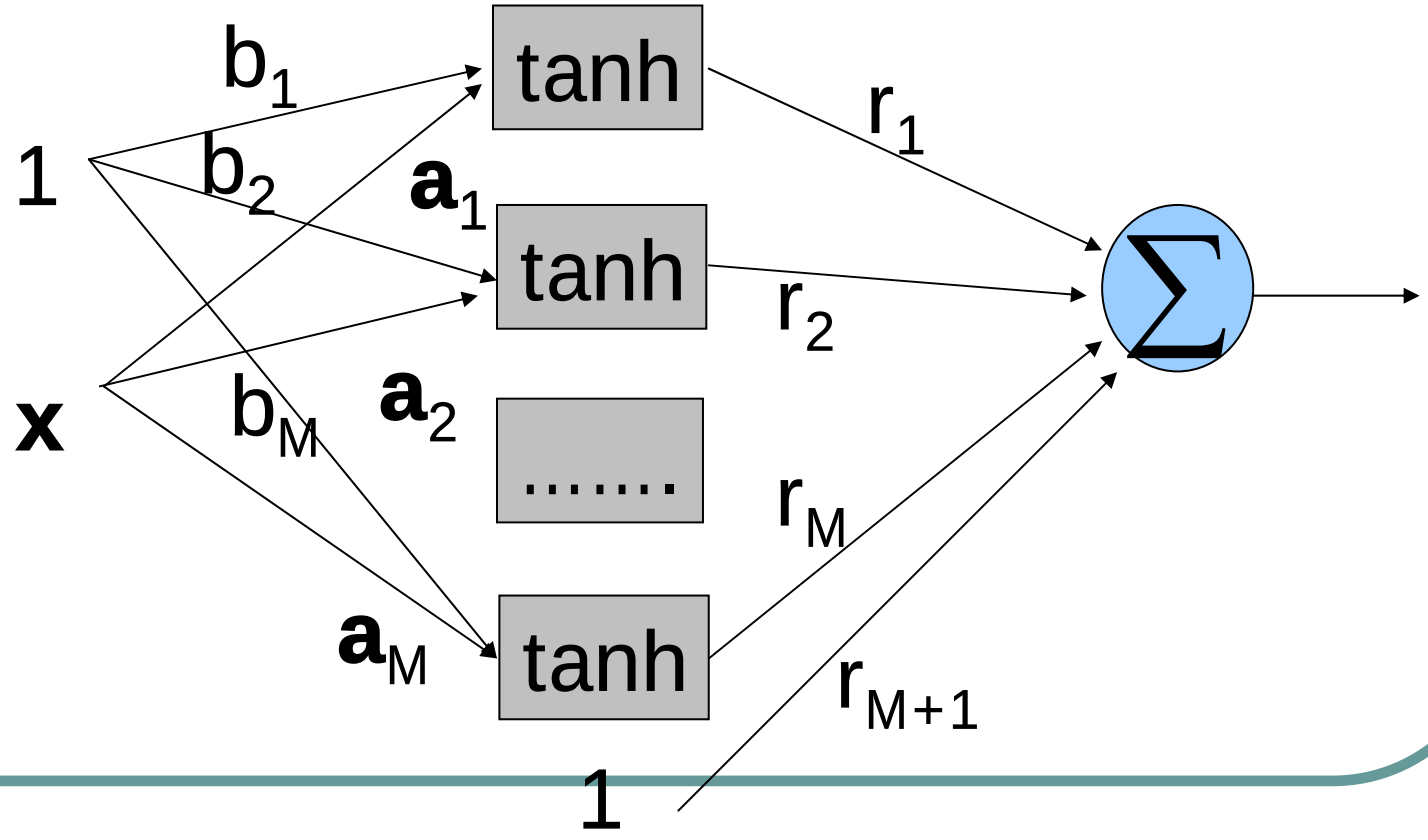
$$y = 2x_1 + 3x_2 + 1 \qquad y = \tanh(2x_1 + 3x_2 + 1)$$





Multiple perceptrons

Multilayer perceptrons (MLP)
(Rumelhart, 1986)



Network Function

$$f(\mathbf{x};\theta) = \sum_{m=1}^{M} r_m \tanh(\mathbf{a}_m^T \mathbf{x} + b_m) + r_0$$

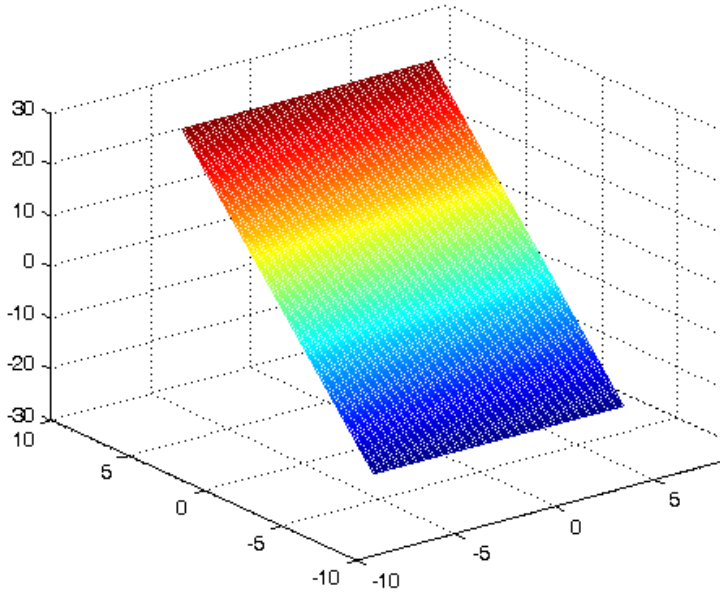
$$\theta = \{\mathbf{a}_m\} \cup \{b_m\} \cup \{r_m\}$$

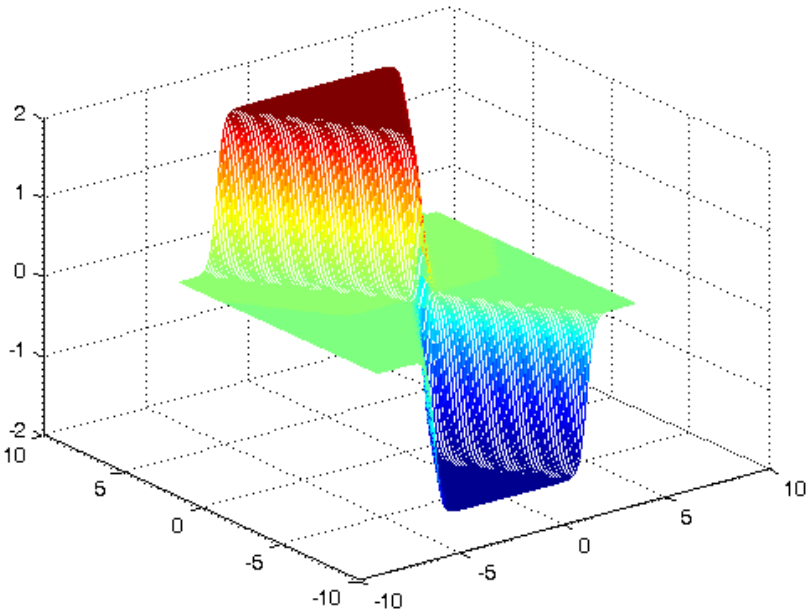
Multiple post-tanh projections

M=2,d=2

f(x1,x2)= (2*x1+3*x2+1)
+(2*x1-3*x2+2)

f(x1,x2)=tanh(2*x1+3*x2+1)
+tanh(2*x1-3*x2+2)





Paired data for function approximation

- Target function $\mathrm{F}: R^d \rightarrow R$
- Paired Data

$$(\mathbf{x}[t], y[t]),\ t = 1,\ldots, N$$

- Predictors

$$\mathbf{x}[t] = (x_1[t],\ldots, x_d[t])^T$$

- Targets

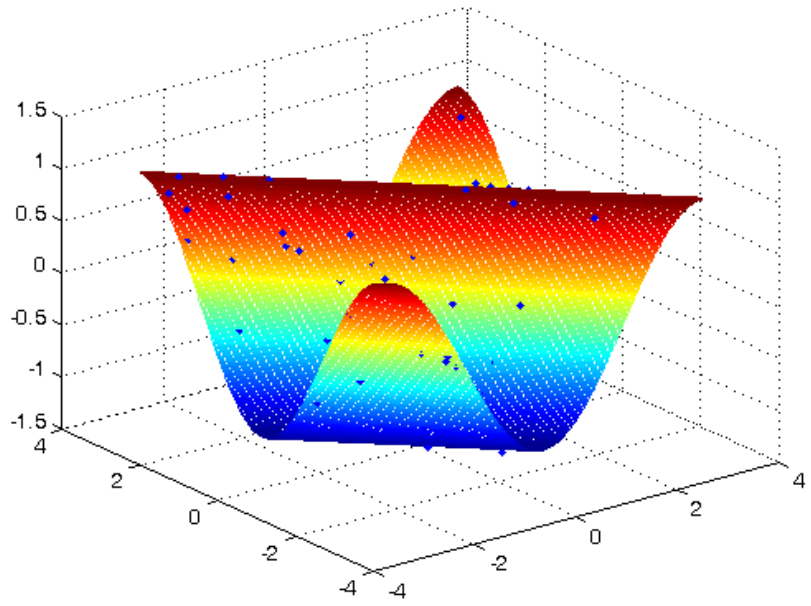
$$y[t] = \mathrm{F}(\mathbf{x}[t]) + noise$$

Paired data

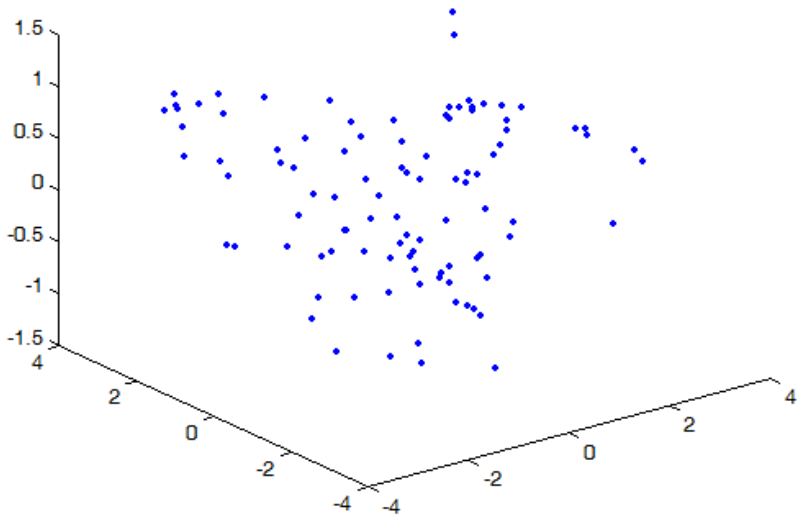
$$d = 2$$

$$\mathrm{F}(x_1 + x_2) = \cos(x_1 + x_2)$$

Target function



Paired data



Hyper-plane Fitting

MLP_Tool.fig MLP_Tool.m

mcc –m MLP_Tool.m

MLP_Tool.ctf MLP_Tool.exe

Function approximation

- Approximating function :
  - Multilayer perceptrons
  - Equivalently, multiple post-tanh projections
- Subject to given paired data, optimize network parameters, $\theta=\{\mathbf{a},\mathbf{b},\mathbf{r}\}$, to reconstruct the unknown target function.

Approximating error

$$\mathrm{E}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}(y[t] - f(\mathbf{x}[t];\boldsymbol{\theta}))^2$$

Unconstrained optimization

$$\boldsymbol{\theta}_{opt} = \arg\min_{\boldsymbol{\theta}} \mathrm{E}(\boldsymbol{\theta})$$

$$\mathrm{E}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}(y[t] - f(\mathbf{x}[t];\boldsymbol{\theta}))^2$$

Target functions I

(2009,Wu)

Table 1. Target functions

$f_1(\mathbf{x}) = \sin(x_1+x_2)$	
$f_2(\mathbf{x}) = x_1^2+x_2^2$	
$f_3(\mathbf{x}) = 0.5x_1^2-0.9x_2^2$	
$f_4(\mathbf{x}) = \exp(-0.05x_1^2-0.09x_2^2)$	noise
$f_5(\mathbf{x}) = \sin([1,-1]^T x)+\exp(-x^T Ax)$	$A = \begin{bmatrix} 0.08 & 0.015 \\ 0.02 & 0.075 \end{bmatrix}$
$f_6(\mathbf{x}) = \tanh(0.8x_1+0.2x_2)+\tanh(0.3x_1-0.9x_2)$	
$f_7(\mathbf{x}) = 0.5\sin(x_1+x_2)+0.2x_1-0.2x_2$	
$f_8(\mathbf{x}) = \exp(-(\mathbf{x}-\mathbf{w}_1)^T A(\mathbf{x}-\mathbf{w}_1))+\exp(-(\mathbf{x}-\mathbf{w}_2)^T B(\mathbf{x}-\mathbf{w}_1))$	$B = \begin{bmatrix} 0.12 & -0.02 \\ 0.035 & 0.075 \end{bmatrix}$
$f_9(\mathbf{x}) = f_8(\mathbf{x})+0.5sin(x_1+0.3x_2)+0.5sin(0.2x_1-0.8x_2)$	

Target function II

(2006 Wu et al)

TABLE III
TARGET FUNCTIONS

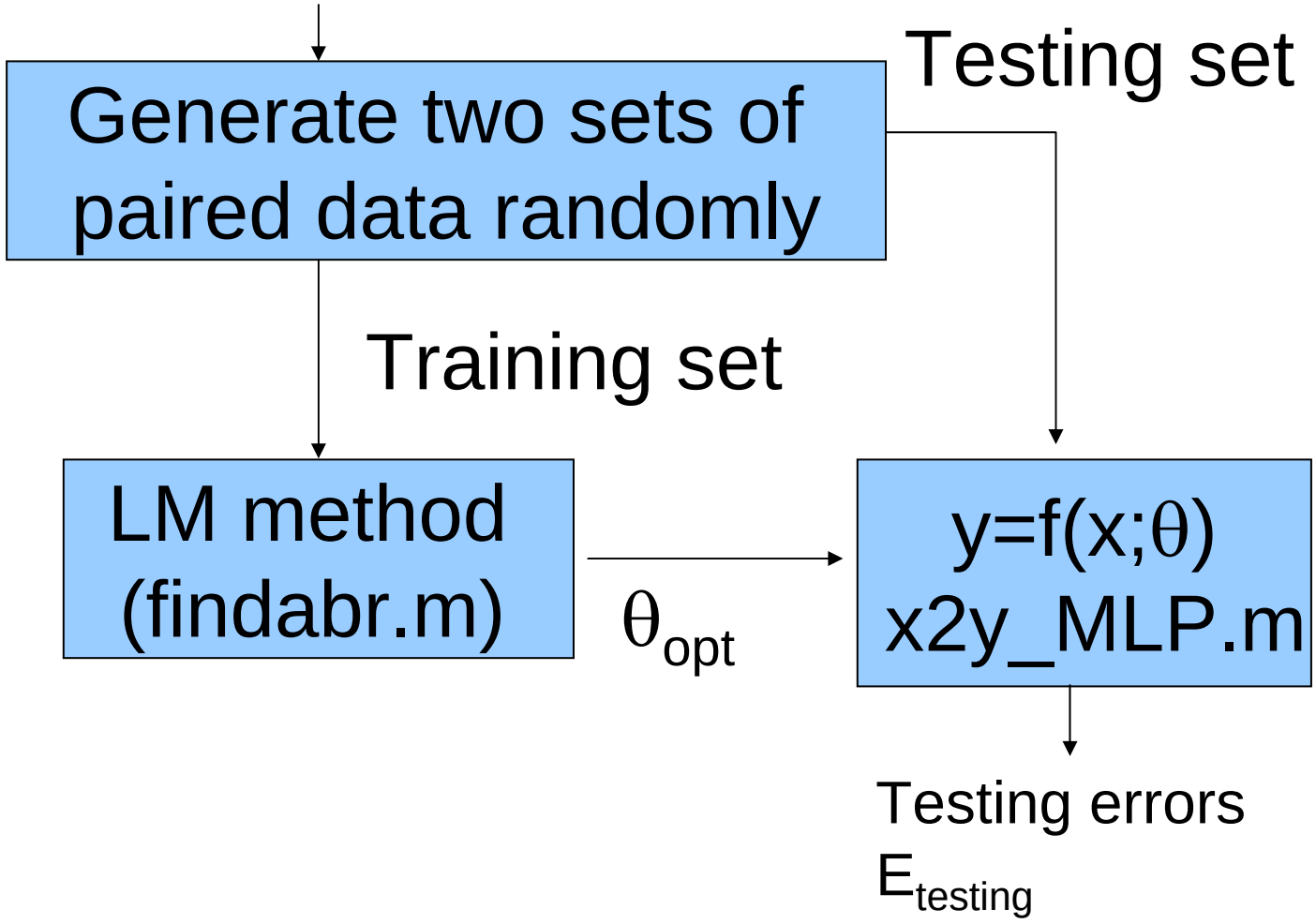
Target function	Description
$y_1(z) = \psi(z) + n$	One-dimensional function with uniform noise $n \in [-0.5, 0.5]$
$y_2(\mathbf{z}) = 0.3z_1 + z_2 - 0.5z_3 + 2z_4 - 0.7z_5$	Single linear projection
$y_3(\mathbf{z}) = \tanh(0.3z_1 + z_2 - 0.5z_3 + 2z_4 - 0.7z_5)$	Single post-nonlinear projection
$y_4(\mathbf{z}) = \tanh(0.8z_1 + 0.2z_2) + \tanh(0.3z_1 - 0.9z_2)$	Sum of two PNL projections
$y_5(\mathbf{z}) = \tanh(0.8z_1 + 0.2z_2 + 0z_3 + 0z_4 + 0z_5) + \tanh(0.3z_1 - 0.9z_2 + 0z_3 + 0z_4 + 0z_5)$	Sum of two PNL projections with ineffective attributes
$y_6(\mathbf{z}) = 0.5z_1^2 - 0.9z_2^2$	Quadratic function
$y_7(\mathbf{z}) = \exp(-\frac{z_1^2}{\sigma_1} - \frac{z_2^2}{\sigma_2}) \cos(\kappa z_1 + \phi)$	Gabor function

Install NNSYSID (2001)

1. Install The NNSYSID Toolbox
2. Download findabr.m x2y_MLP.m

   (Levenberg Marquardt method)
3. Set path to recruit the directory of where NNSYSID is installed

Training and Testing

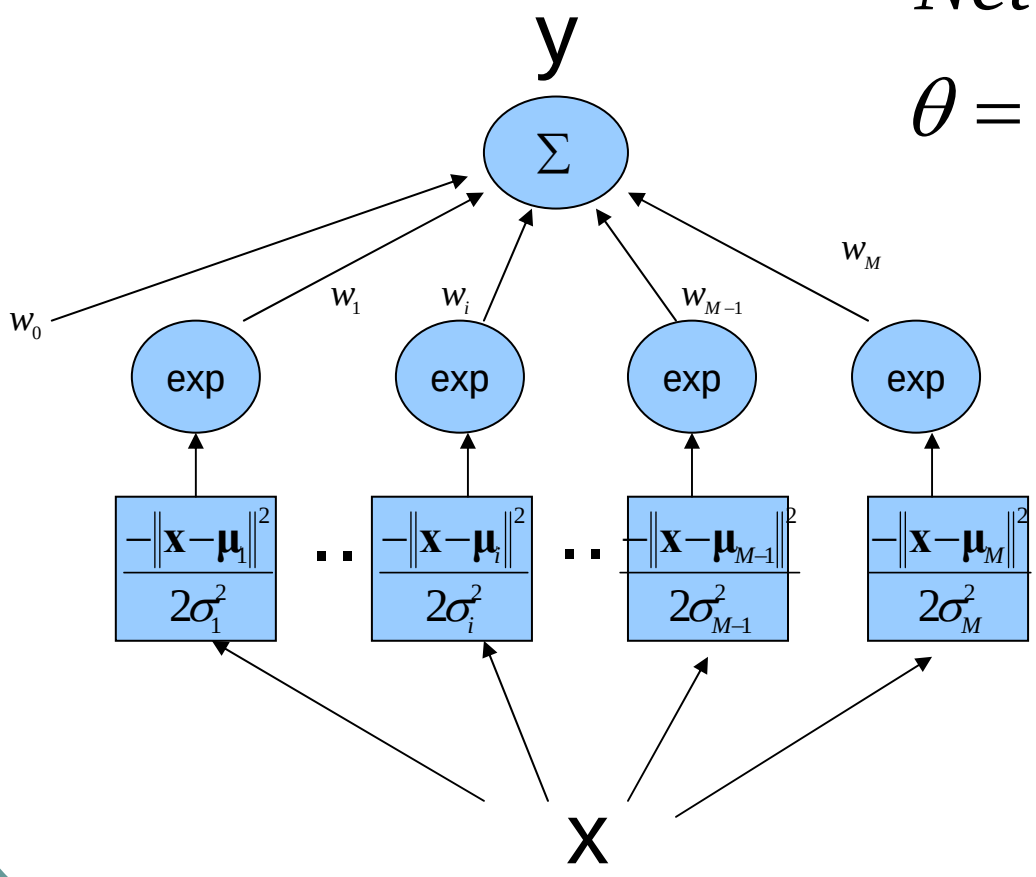


Network function

$$y(t \mid \theta) = G(\mathbf{x}[t] \mid \theta)$$

$$= w_0 + \sum_{m=1}^{M} w_m \exp\left(-\frac{\left\|\mathbf{x}[t] - \boldsymbol{\mu}_m\right\|^2}{2\sigma_m^2}\right)$$

Architecture



Network parameter

$$\theta = \{\mathrm{w}_i\}_i \cup \{\boldsymbol{\mu}_i\}_i \cup \{\sigma_i\}_i$$

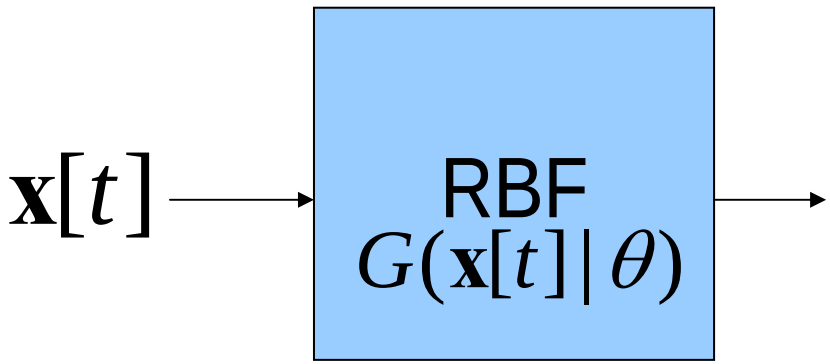
Network parameter

$$\theta = [\mathbf{u}_1^T\ \mathbf{u}_2^T \cdots \mathbf{u}_M^T\ \sigma_1\ \sigma_2 \cdots \sigma_{\mathrm{M}}\ \mathrm{w}_0\ \mathrm{w}_1\ \mathrm{w}_2 \cdots \mathrm{w}_{\mathrm{M}}]^T$$

$$= [\theta_1, \ldots, \theta_{M*d+2M+1}]$$

Training set

$$S = \{(\mathbf{x}[t], y[t])\}_{t=1}^{N}$$



Desired output

Network output

Training

Mean square error

$$E_S(\theta) = \frac{1}{2N}\sum_{t=1}^{N}(y[t] - y(t \mid \theta))^2$$

Unconstrained optimization

$$\widehat{\boldsymbol{\theta}} = \arg\min_{\{\boldsymbol{\theta}\}} E_S(\boldsymbol{\theta})$$

Find θ to minimize $\mathrm{E}_s(\theta)$

Iterative approaches

- Gradient method
- Newton-Gauss method
- Levenberg-Marquardt method

Iterative approach

1. Initialize $\boldsymbol{\theta}_i$ with i=0

2. Determine $\Delta\theta_i$
3. Update network parameters

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i + \boldsymbol{\Delta\theta}_i$$

4. If halting condition holds, exit
   otherwise i=i+1, go to step 2

Objective function

$$E_S(\boldsymbol{\theta}) = \frac{1}{2N}\sum_{t=1}^{N}\varepsilon^2(t,\boldsymbol{\theta})$$

$$\varepsilon(t,\boldsymbol{\theta}) \quad = \quad y\,[t] - y(t|\boldsymbol{\theta})$$

Network output

Desired output

Gradient method

$$\Delta\boldsymbol{\theta}_i \propto -\frac{dE_S(\boldsymbol{\theta})}{d\boldsymbol{\theta}}\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_i}$$

Gradient

$$E_S(\theta) = \frac{1}{2N}\sum_{t=1}^{N}(y[t] - y(t\,|\,\theta))^2$$

$$\frac{dE_S(\theta)}{d\theta} = \frac{-1}{N}\sum_{t=1}^{N}(y[t] - y(t\,|\,\theta))\frac{dy(t\,|\,\theta)}{d\theta}$$

$$\varepsilon(t,\boldsymbol{\theta}) \quad = \quad y\,[t] - y(t|\boldsymbol{\theta})$$

derivative

$$\psi(t,\boldsymbol{\theta}) = \frac{dy(t|\boldsymbol{\theta})}{d\boldsymbol{\theta}}$$

$$\frac{dE_S(\theta)}{d\theta}\Big|_{\theta=\theta_i} = \nabla(\boldsymbol{\theta}_i) = \frac{-1}{N}\sum_{t=1}^{N}\varepsilon(t,\boldsymbol{\theta}_i)\psi(t,\boldsymbol{\theta}_i)$$

$$\Delta\boldsymbol{\theta}_i \propto -\frac{dE_S(\boldsymbol{\theta})}{d\boldsymbol{\theta}}\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_i}$$

$$= \quad -\boldsymbol{\nabla}(\boldsymbol{\theta}_i)$$

Exercise

- Function approximation
  - Learning radial basis functions
  - 2D