Levenberg-Marquardt learning

- Multi-layer neural networks
- MLP (Multilayer Perceptrons)
- RBF (Radial Basis Functions)

Taylor expansion

$$E(\theta + \Delta\theta)$$

$$\simeq E(\theta) + E'(\theta)^T \Delta\theta$$

$$E(\theta + \Delta\theta)$$

$$\simeq E(\theta) + E'(\theta)^T \Delta\theta + \frac{1}{2} \Delta\theta^T E''(\theta) \Delta\theta$$

Function Approximation

- High-dimensional nonlinear function approximation
- Linear combination of basis functions
  - Projective basis function
  - Radial basis function

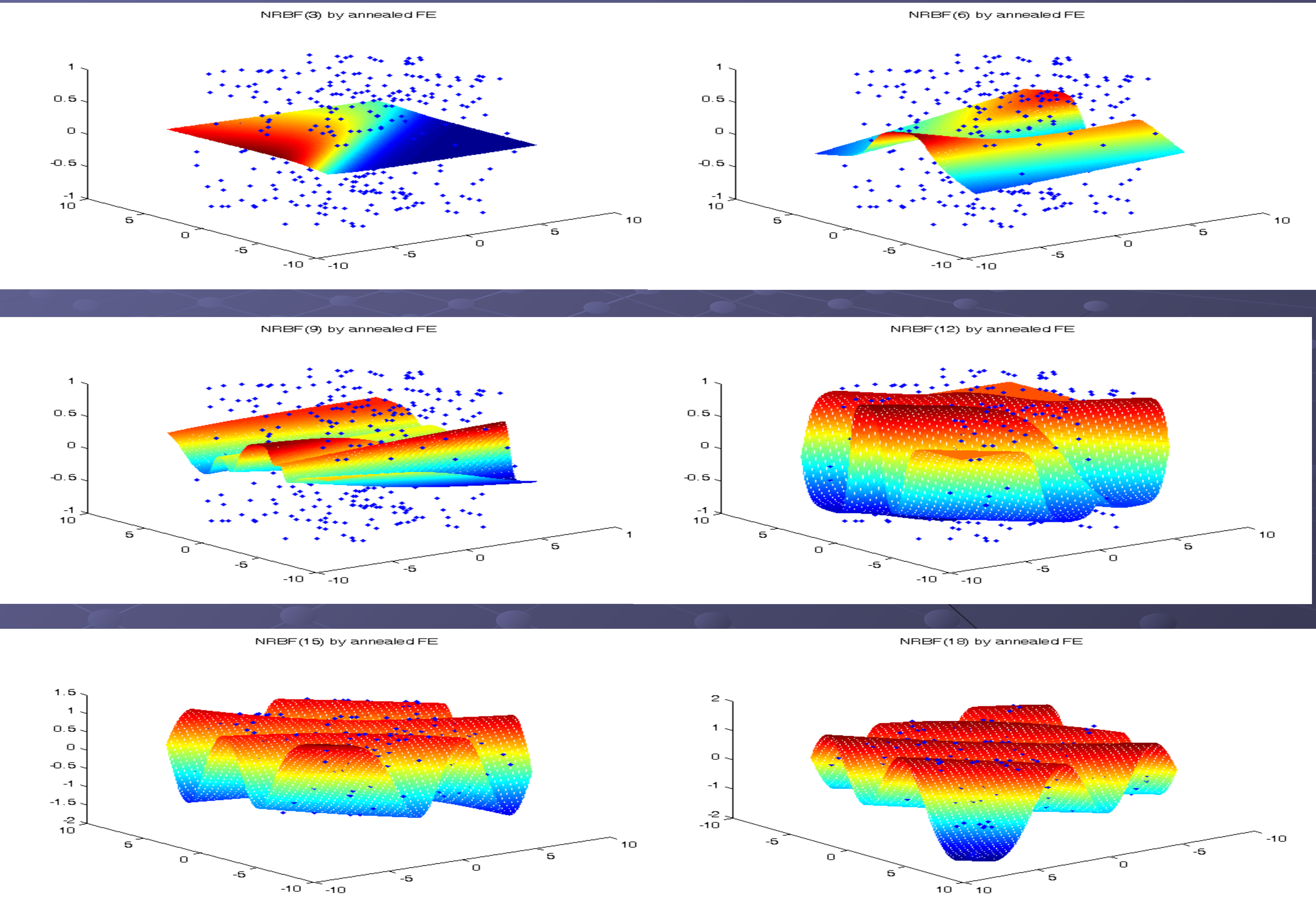
Function approximation

- Data oriented function approximation
- Paired data sampled from an unknown target function : $R^d \longrightarrow R$

  ($\mathbf{x}$[t], y[t]),t=1,…,N

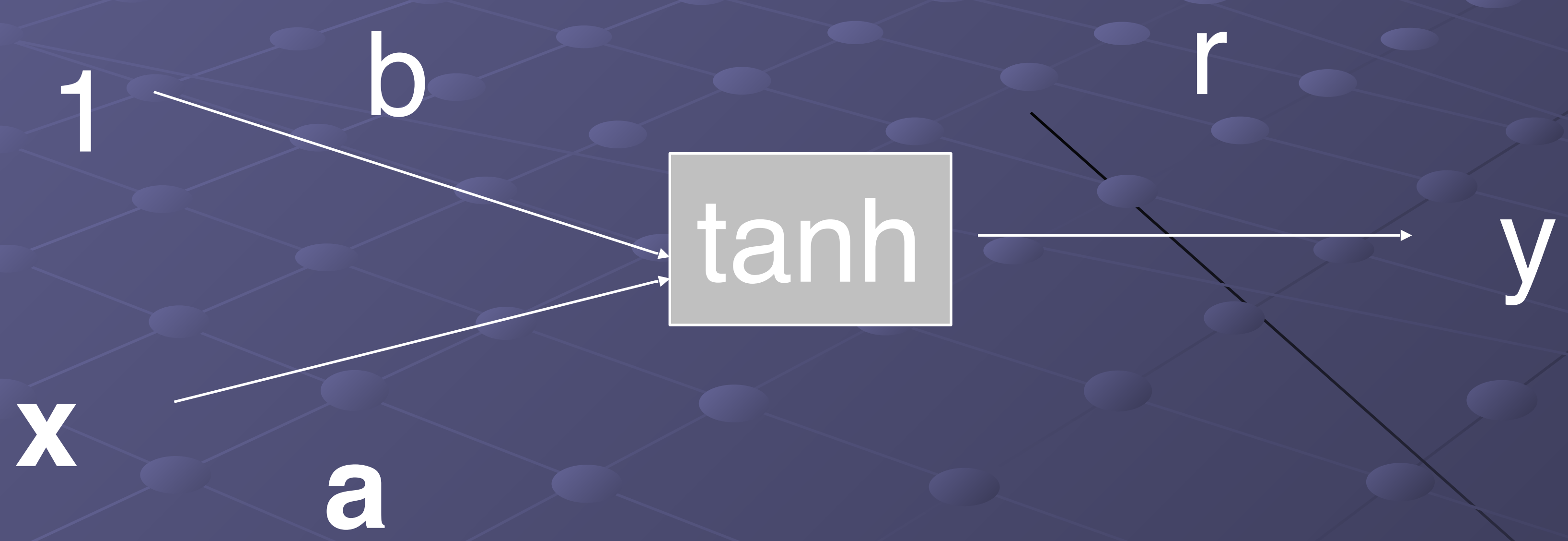
Desired target (output)

Predictor (input)


NRBF(3) by annealed FE
NRBF(6) by annealed FE
NRBF(9) by annealed FE
NRBF(12) by annealed FE
NRBF(15) by annealed FE
NRBF(18) by annealed FE

Perceptron
Projective basis function

$$y = r\tanh(\mathbf{a}^T\mathbf{x} + b)$$



Post-tanh projection

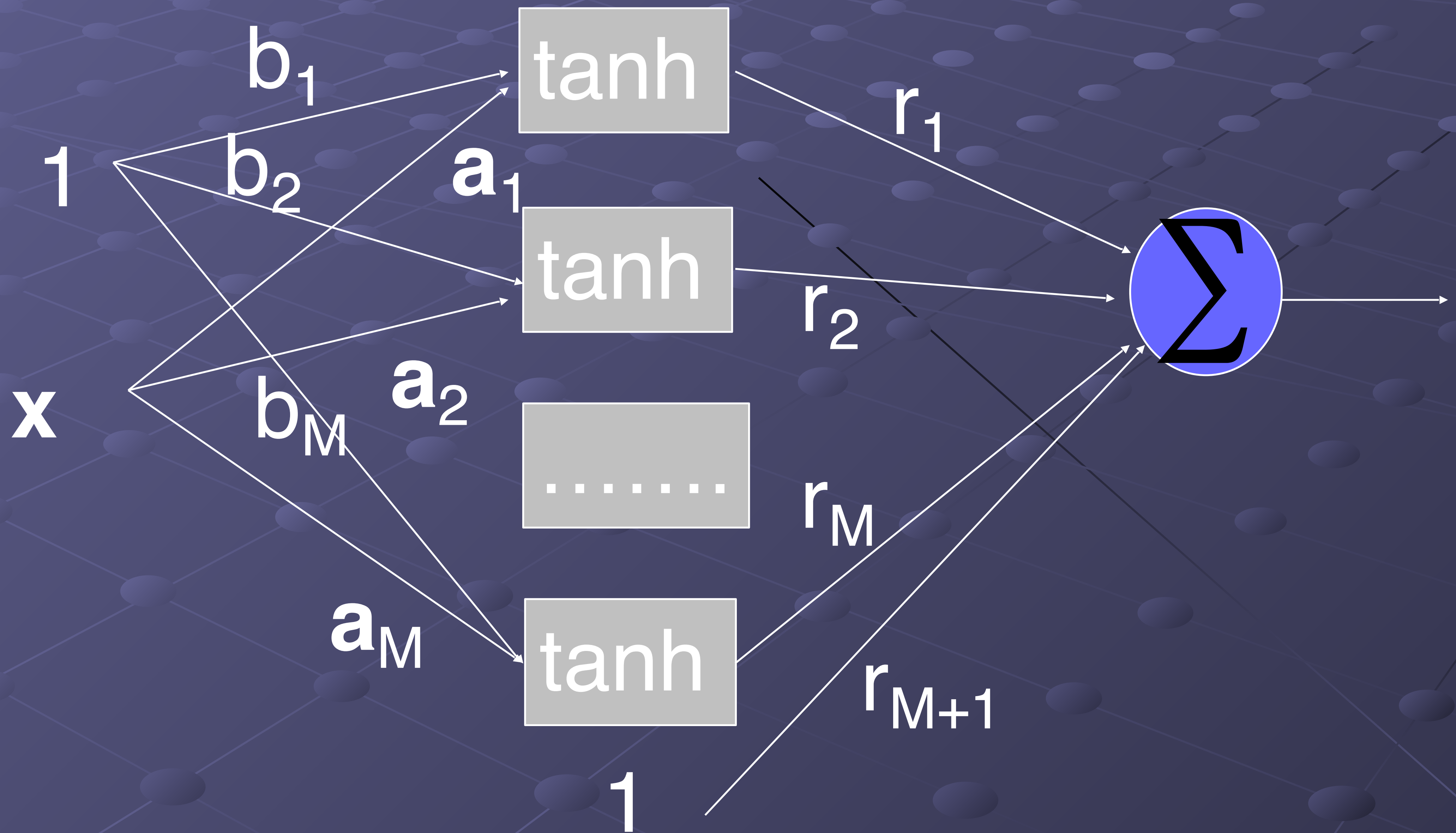
Posterior Weight

$$y = r\tanh(\mathbf{a}^T\mathbf{x} + b)$$

Linear projection

Post-nonlinear function

Multiple (Multi-layer) perceptrons



MLP (multilayer perceptrons)

- Weight sum of post-tanh projections

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

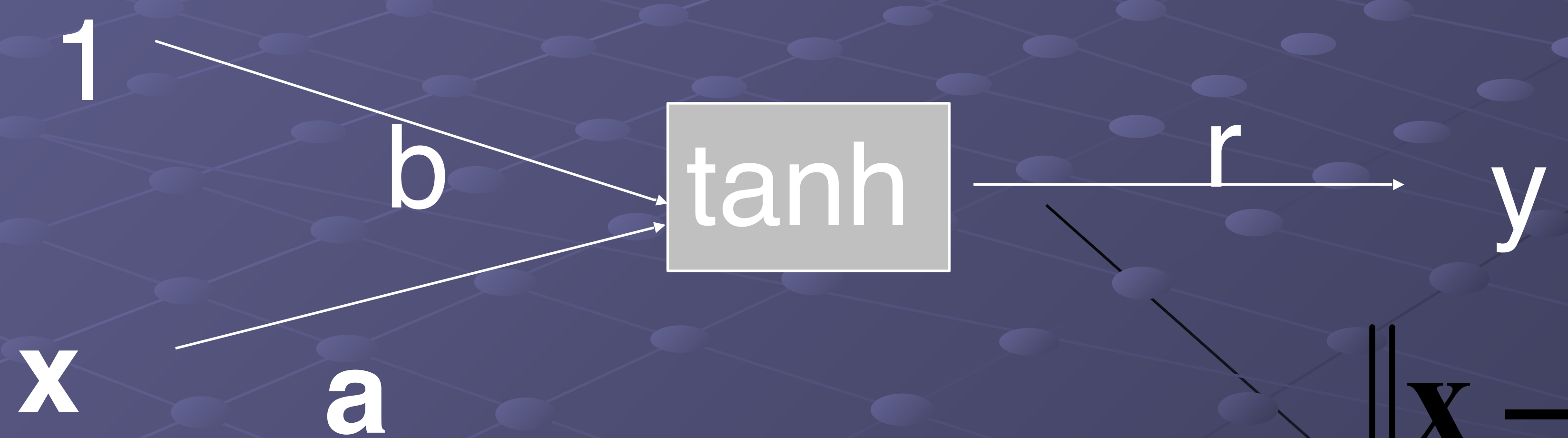
$$\theta = \{\mathbf{a}_m\} \cup \{b_m\} \cup \{r_m\}$$

Why Projection?

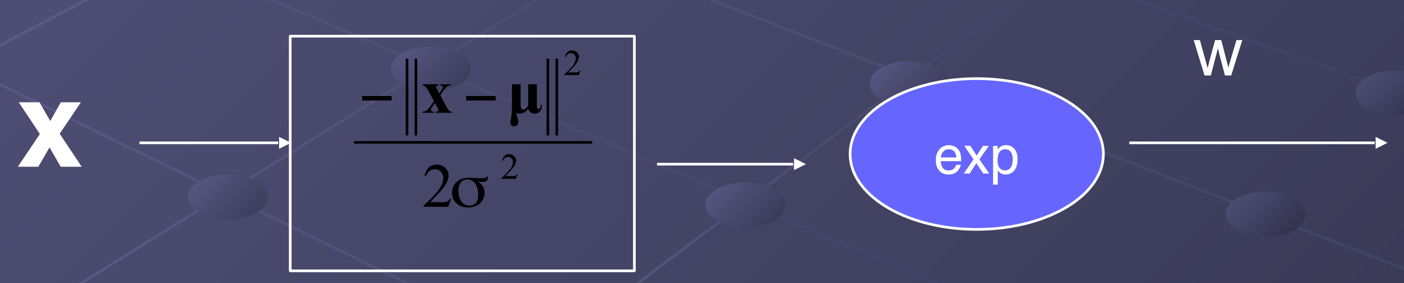
- Why projection?
  - Linear projection versus Radial basis function
- Why hyper-tangent
  - Hyper-tangent function versus exponential function

Radial basis function

$$y = r\tanh(\mathbf{a}^T\mathbf{x} + b)$$



$$y = w\exp\left(-\frac{\|\mathbf{x}-\boldsymbol{\mu}\|^2}{2\sigma^2}\right)$$



RBF Network function

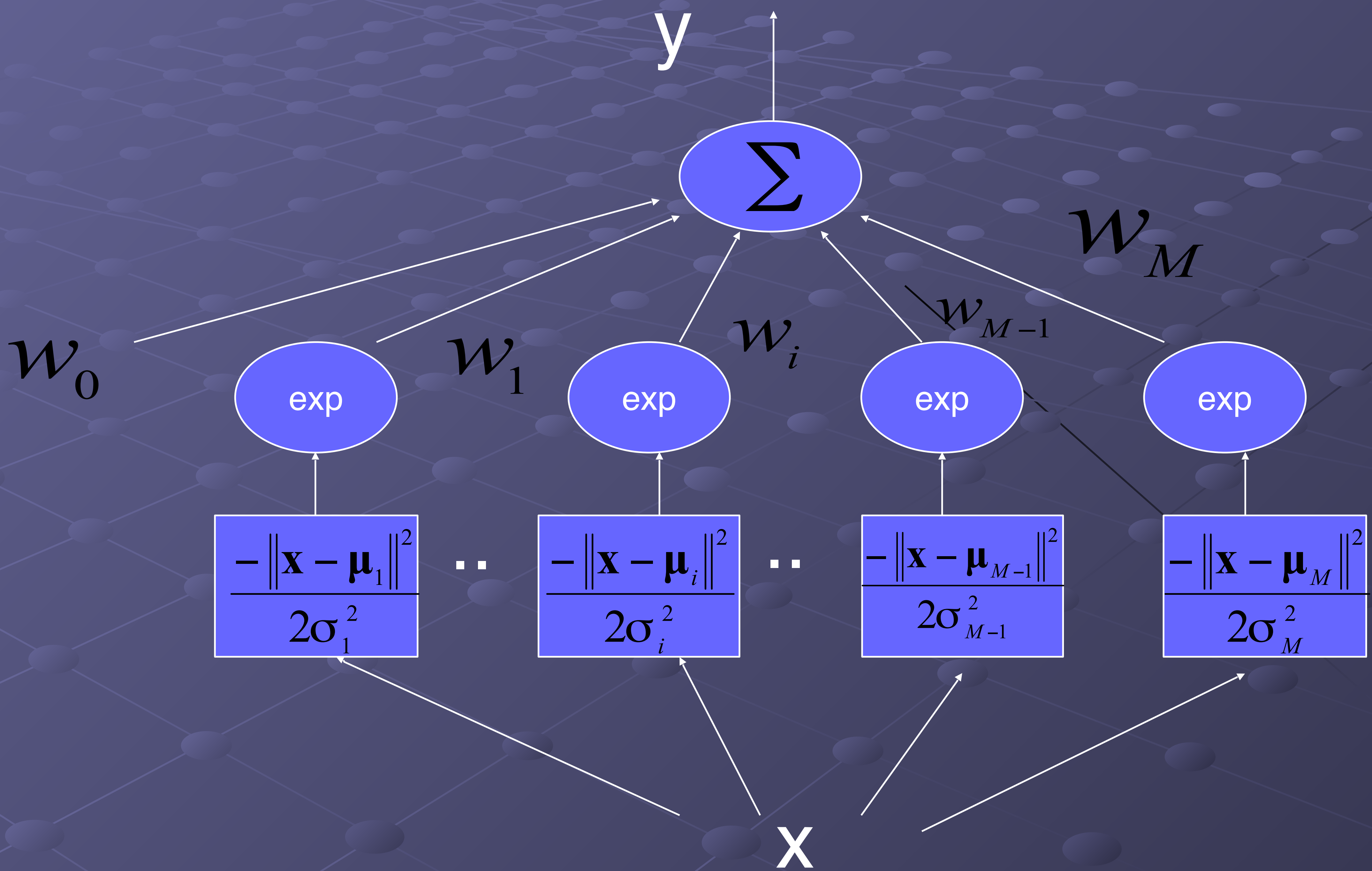
$$y(t\,|\,\theta) = G(\mathbf{x}[t]\,|\,\theta)$$

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

Network parameter

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

RBF Network



y
Σ
w_0
w_1
w_i
w_{M-1}
w_M
exp
exp
exp
exp
$\frac{-\|\mathbf{x}-\boldsymbol{\mu}_1\|^2}{2\sigma_1^2}$
..
$\frac{-\|\mathbf{x}-\boldsymbol{\mu}_i\|^2}{2\sigma_i^2}$
..
$\frac{-\|\mathbf{x}-\boldsymbol{\mu}_{M-1}\|^2}{2\sigma_{M-1}^2}$
$\frac{-\|\mathbf{x}-\boldsymbol{\mu}_M\|^2}{2\sigma_M^2}$
x

Posterior weights

- Two stages
- Calculation of radial basis functions
- A linear combination

Stage I

Radial basis functions

$$f_m(x \mid \mu_m, \delta_m)$$

$$= \exp\left(-\frac{\|x - \mu_m\|^2}{2\delta_m^2}\right)$$

$$o[t] = (o_1[t], \dots, o_M[t])^T$$

$$o_m[t] = f_m(x[t])$$

Stage II

Posterior weighting

$$y[t] = r^T o[t] + r_0$$

- - - - - - - - - -

$$x[t] \to o[t] \to y[t]$$

Given $\{u_m\}$, $\{\delta_m\}$

optimize r

$$O = \begin{bmatrix} o[1]^T \\ o[2]^T \\ \vdots \\ o[N]^T \end{bmatrix}, \quad N \times d$$

$$D = [\,O \quad \text{ones}(N,1)\,]$$

$$Dr = y$$

$$Or = y$$

$$O^T O r = O^T y$$

$$C(r) = r^T O^T O r - O^T y$$

$C(r)$

$= r^T O r - r^T y + \lambda r^T r$

O^T O^T

regularization

$$\frac{dc}{dr}$$

$$= O^T O r - O^T y + \lambda r = 0$$

$$(O^T O + \lambda I) r = O^T y$$

$$r = \mathrm{pinv}(D^T D + \lambda I)\, D^T y$$

optimization of r for given $\{\mu_m\}$ and $\{\sigma_m\}$

Ratsch Algorithm

Algorithm RBF-Net(K, λ, O)

Input:

Sequence of labeled training patterns $\mathbf{Z} = \langle(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_l, y_l)\rangle$

Number of RBF centers K

Regularization constant λ

Number of iterations O

Initialize:

Run K-means clustering to find initial values for μ_k and determine $\sigma_k, k = 1, \ldots, K$, as the distance between μ_k and the closest μ_i $(i \neq k)$.

Do for $o = 1 : O$,

1. Compute optimal output weights $\mathbf{w} = \left(G^\top G + 2\frac{\lambda}{l}\mathbf{I}\right)^{-1} G^\top \mathbf{y}$

2a. Compute gradients $\frac{\partial}{\partial \mu_k} E$ and $\frac{\partial}{\partial \sigma_k} E$ as in (28) and (27) with optimal $\mathbf{w}$ and form a gradient vector $\mathbf{v}$

2b. Estimate the conjugate direction $\overline{\mathbf{v}}$ with Fletcher-Reeves-Polak-Ribiere CG-Method (Press et al., 1992)

3a. Perform a line search to find the minimizing step size δ in direction $\overline{\mathbf{v}}$; in each evaluation of E recompute the optimal output weights $\mathbf{w}$ as in line 1

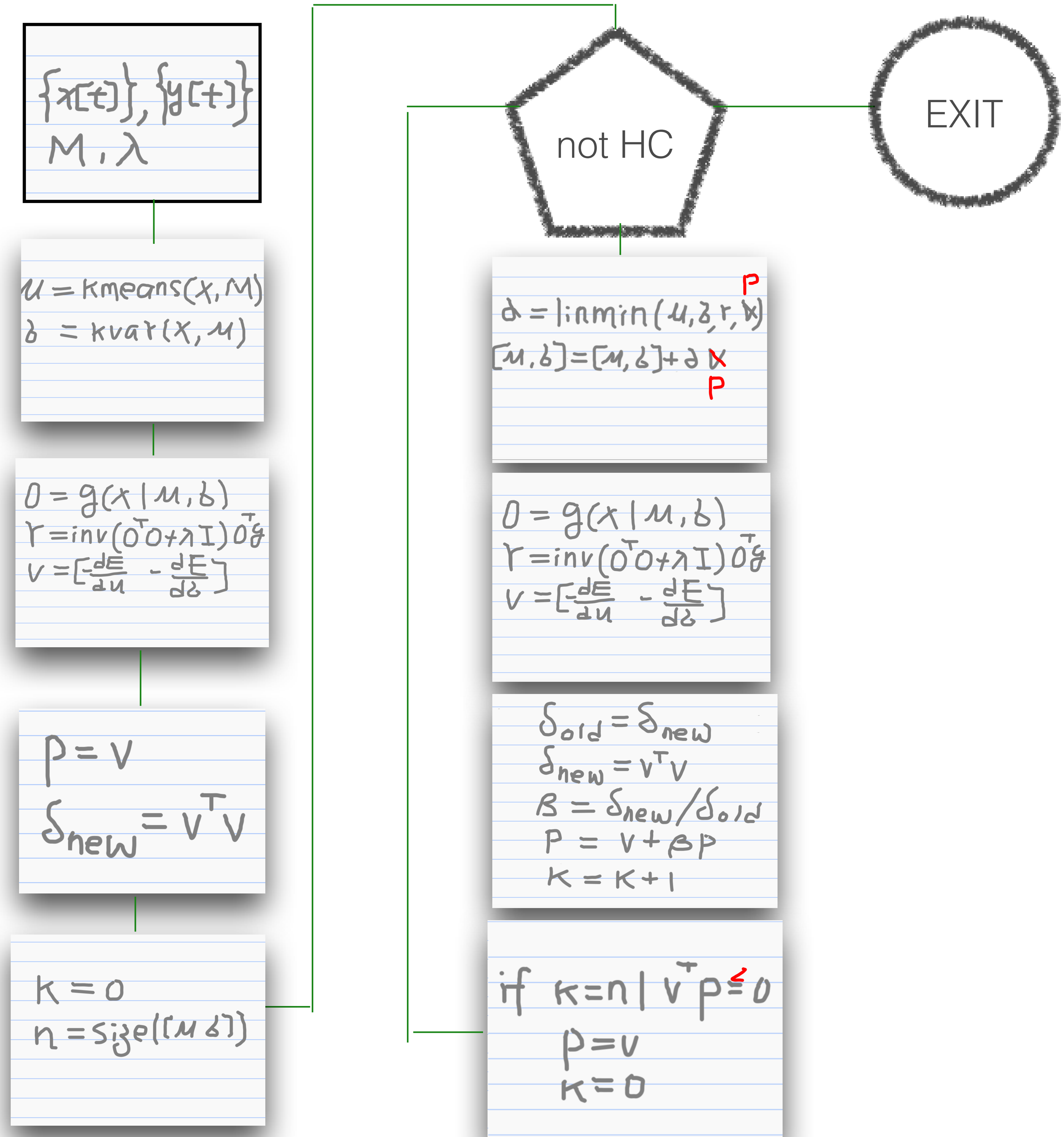
3b. Update μ_k and σ_k with $\overline{\mathbf{v}}$ and δ

Output: Optimized RBF net

K Means

- Given {x[t]}, find centers and variances
- Matlab kmeans.m

http://134.208.26.59/MathProgramming2010/Lecture10/Lecture10II.pdf



{x[t]}, {y[t]}
M, λ
u = kmeans(x, M)
b = kvar(x, u)
O = g(x | u, b)
r = inv(OᵀO + λI)Oᵀg
v = [-dE/du -dE/db]
p = v
δnew = vᵀv
k = 0
n = size([u b])
not HC
EXIT
a = linmin(u, b, r, p)
[u, b] = [u, b] + a p
O = g(x | u, b)
r = inv(OᵀO + λI)Oᵀg
v = [-dE/du -dE/db]
δold = δnew
δnew = vᵀv
β = δnew/δold
p = v + βp
k = k + 1
if k = n | vᵀp ≤ 0
p = v
k = 0

Relation between RBF and MLP

- Why Euclidean distance?
- Why exponential function?
- Why hyper-tangent function?

Network parameter

a vector of d elements

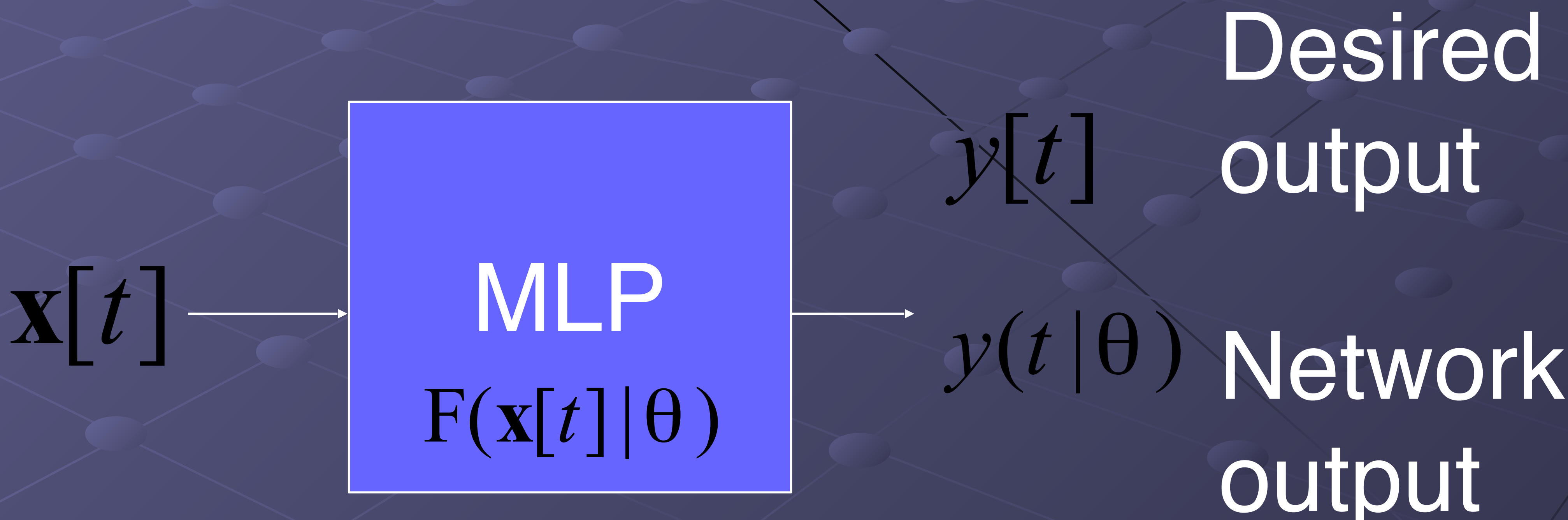
scalar

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

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

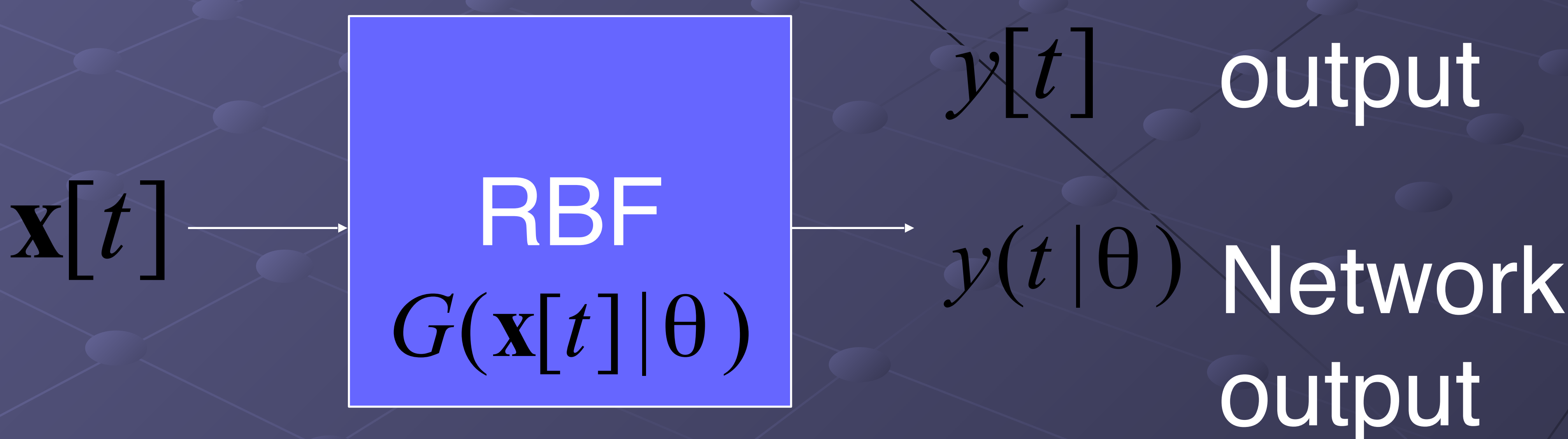
Training set

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



Training set

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



Mean square errors

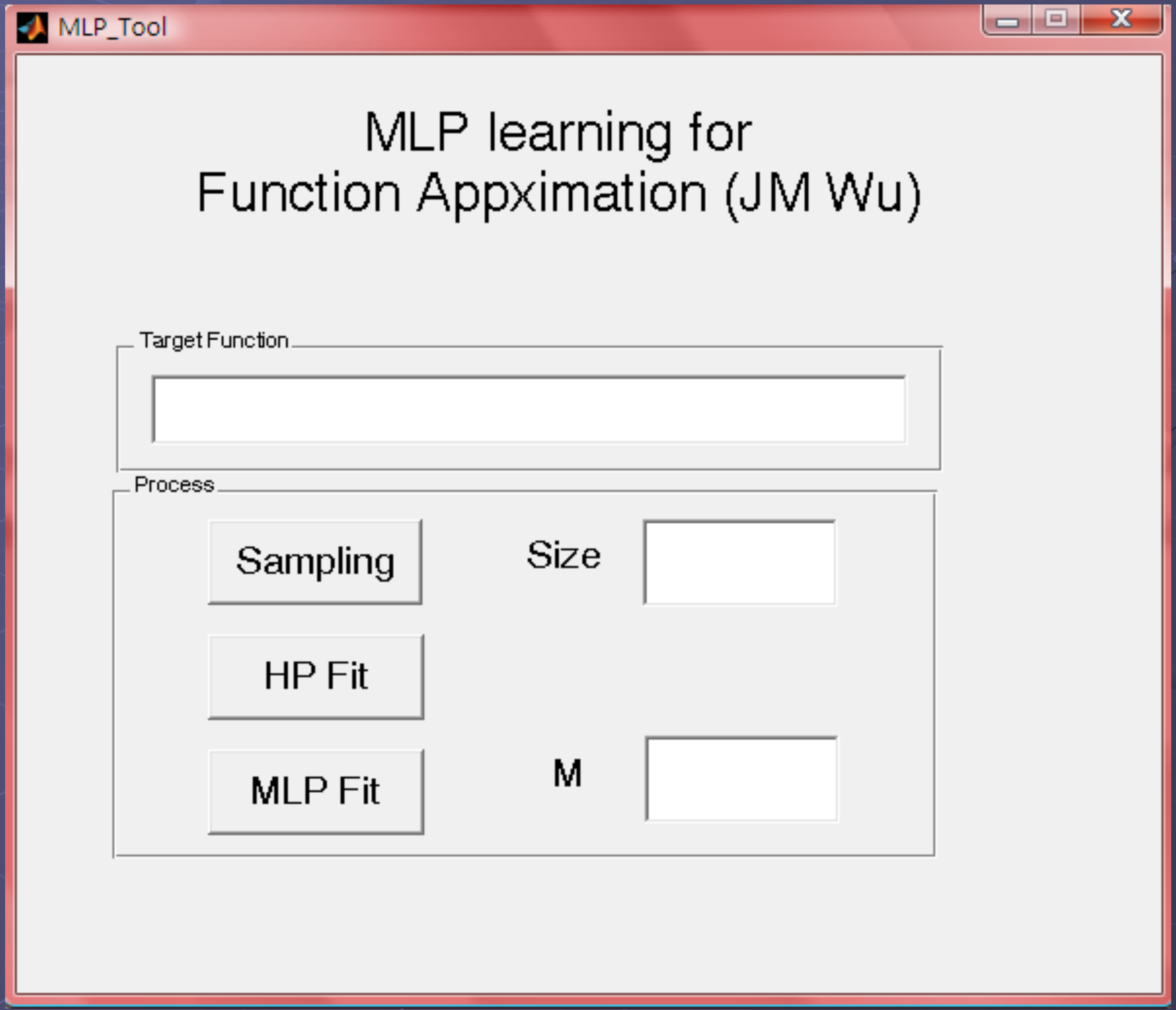
$$E_S(\theta) = \frac{1}{2N}\sum_{t=1}^{N}(y[t] - y(t\,|\,\theta))^2$$

Unconstrained optimization

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

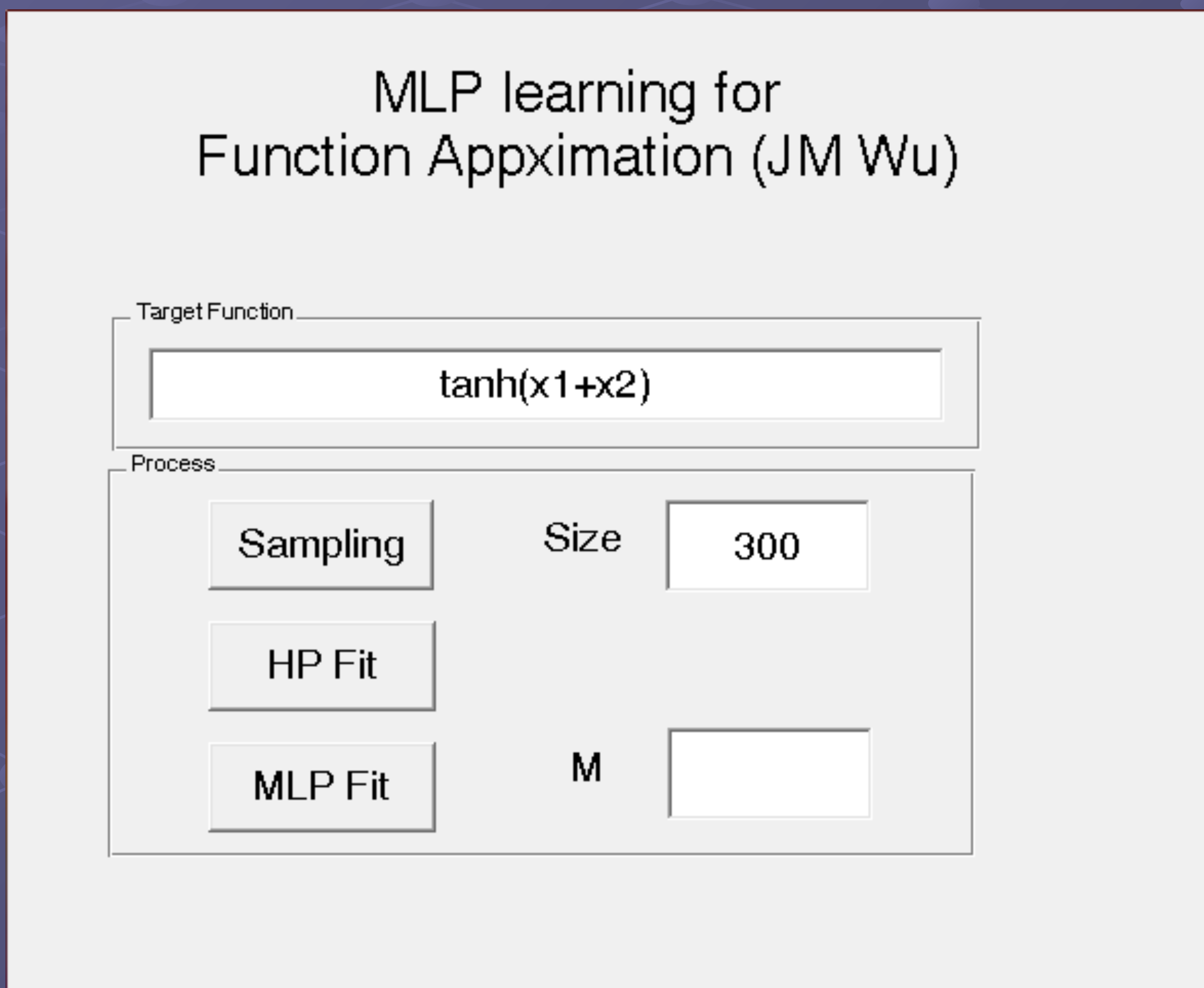
$$Find\ \theta \text{ to minimize } \mathrm{E_S}(\theta)$$

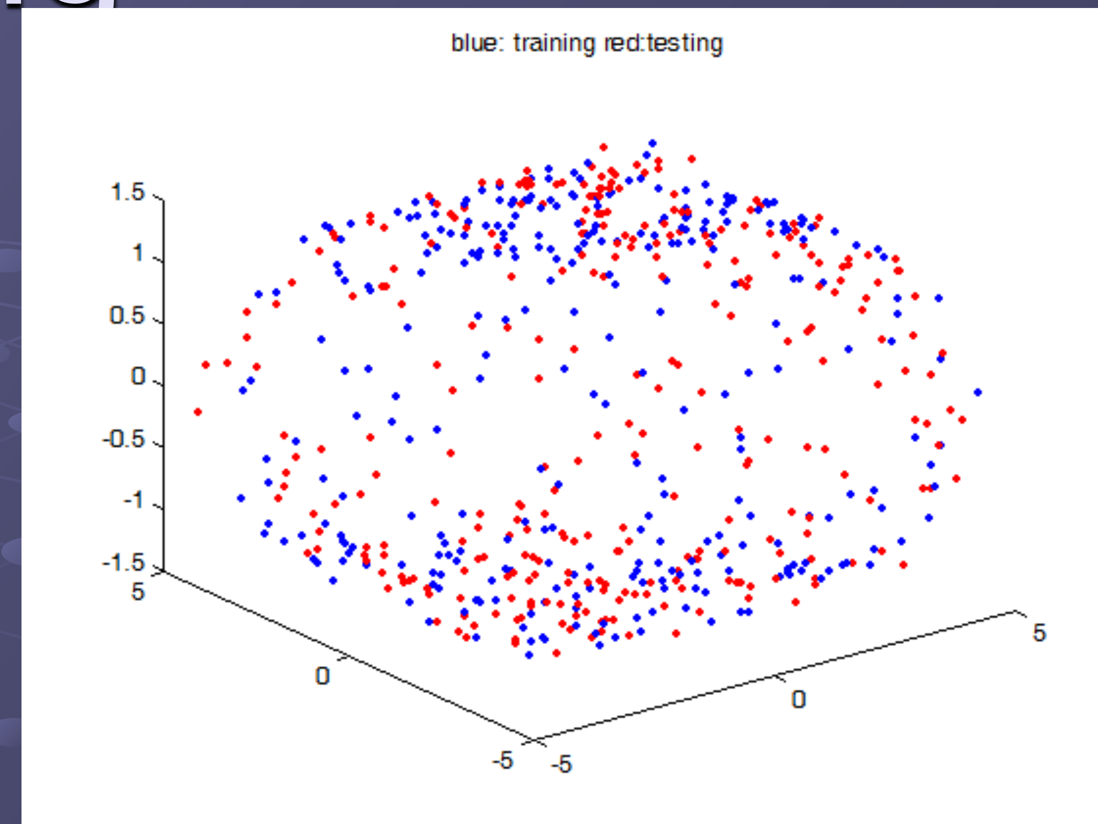
Revisit hyper-plane Fitting

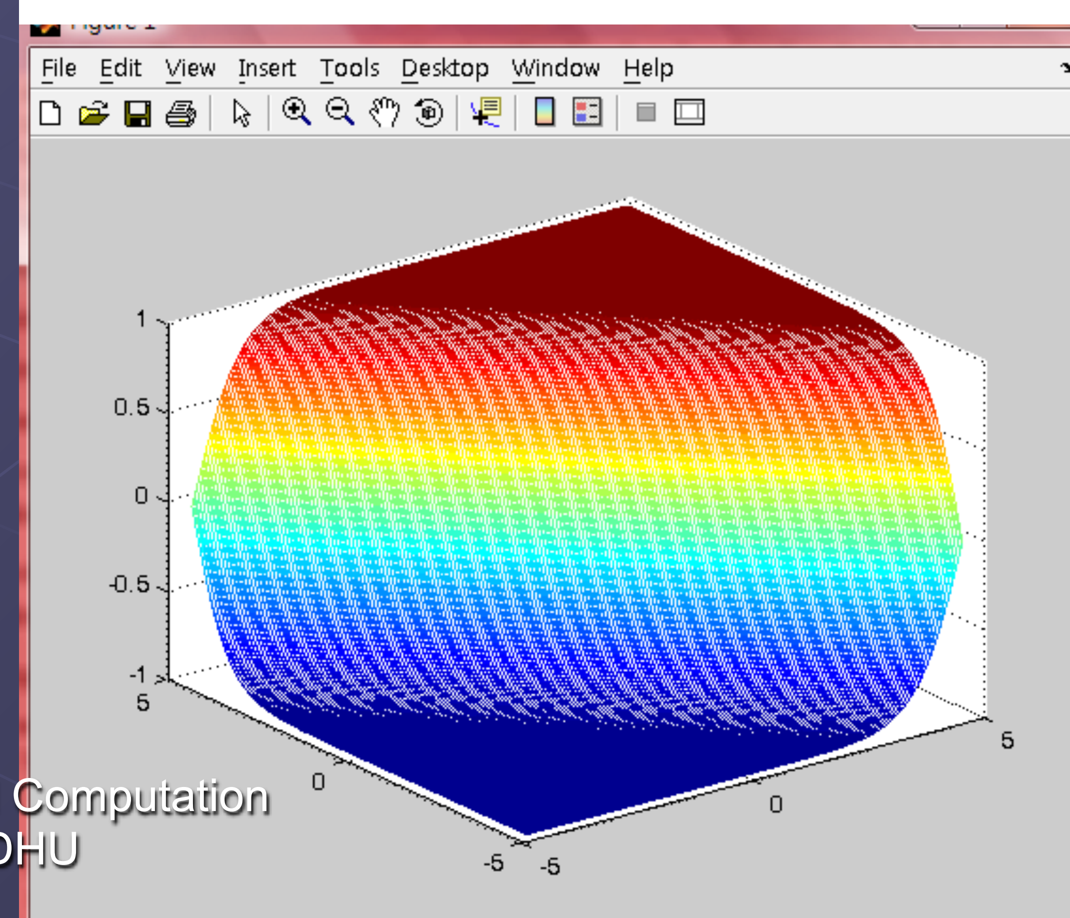


Sampling

Blue : training data
Red : testing data





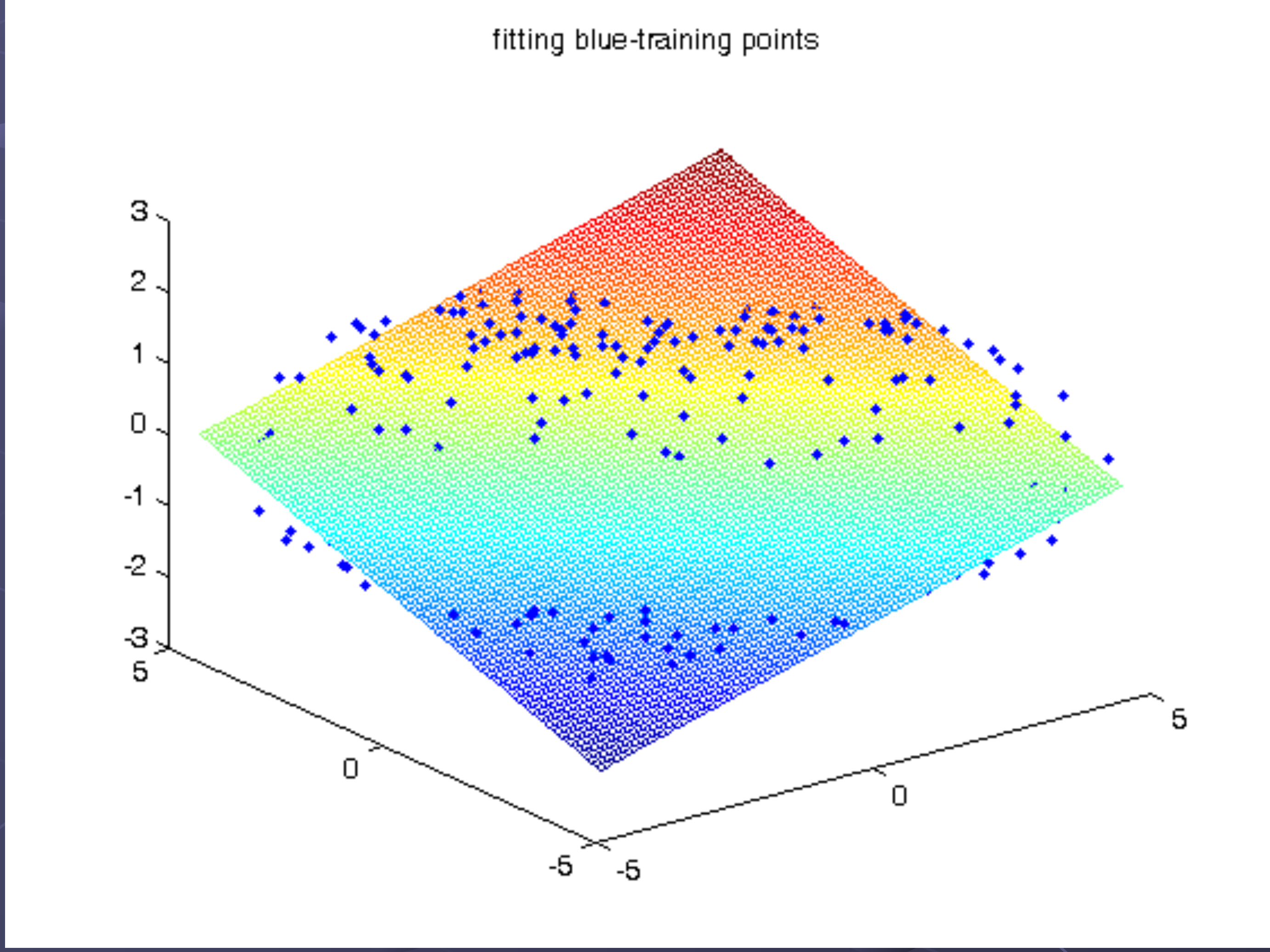


Fitting

1. Fitting blue training data

2. The performance is reflected by the mean square testing error

mean square training error 0.159647
mean square testing error 0.160200



Verification of Generalization

- Training phase
  - Given a training set, $\mathit{find}\ \theta$ to minimize $E_S(\theta)$
- Testing phase
  - Verify $F(x;\theta_{opt})$ by a testing set, which is assumed unknown during training phase
- Both training and testing sets are assumed oriented from the same generating process

Performance evaluation

- Mean square error of training
  - Substitute each predictor in a training set to a network function
  - Determine the mean square error of approximating the desired target by the network output
- Mean square error of testing indicates generalization capability of a network function

Mean square training error

$$E_S(\theta) = \frac{1}{2N}\sum_{t=1}^{N}(y[t] - \mathrm{F}(\mathbf{x}[t];\theta))^2$$

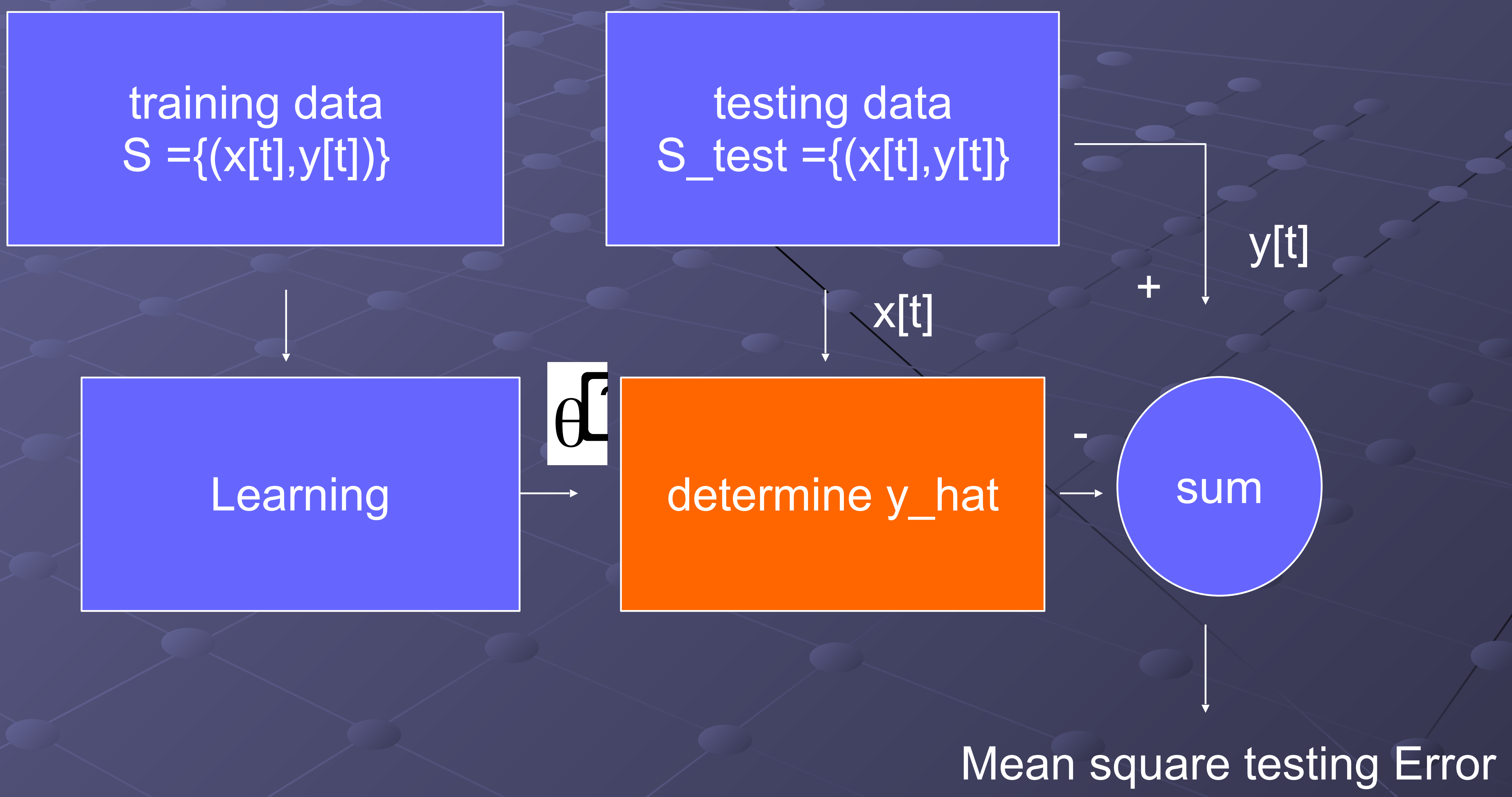
$$\mathrm{S} = \{(\mathbf{x}[\mathrm{t}], \mathrm{y}[\mathrm{t}])\}_{\mathrm{t}} \text{ : training set}$$

Mean square testing error

$$E_{S\_\mathrm{Test}}(\theta) = \frac{1}{2N}\sum_{t=1}^{N}\left(y[t] - \mathrm{F}(\mathbf{x}[t];\theta)\right)^2$$

$$\mathrm{S\_Test} = \{(\mathbf{x}[\mathrm{t}], \mathrm{y}[\mathrm{t}])\}_{\mathrm{t}} : \text{testing set}$$

Prediction


training data
S ={(x[t],y[t])}
testing data
S_test ={(x[t],y[t]}
y[t]
+
x[t]
Learning
θ
determine y_hat
-
sum
Mean square testing Error

Over-fitting

- Extremely low training error but high testing error
- Over-fitting implies false generalization

Table lookup

- Over-fitting
- S: training set

$$f(x) = y[t^*]$$

$$t^* = \arg\min_t \| \mathbf{x} - \mathbf{x}[t] \|^2$$

Iterative approaches

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

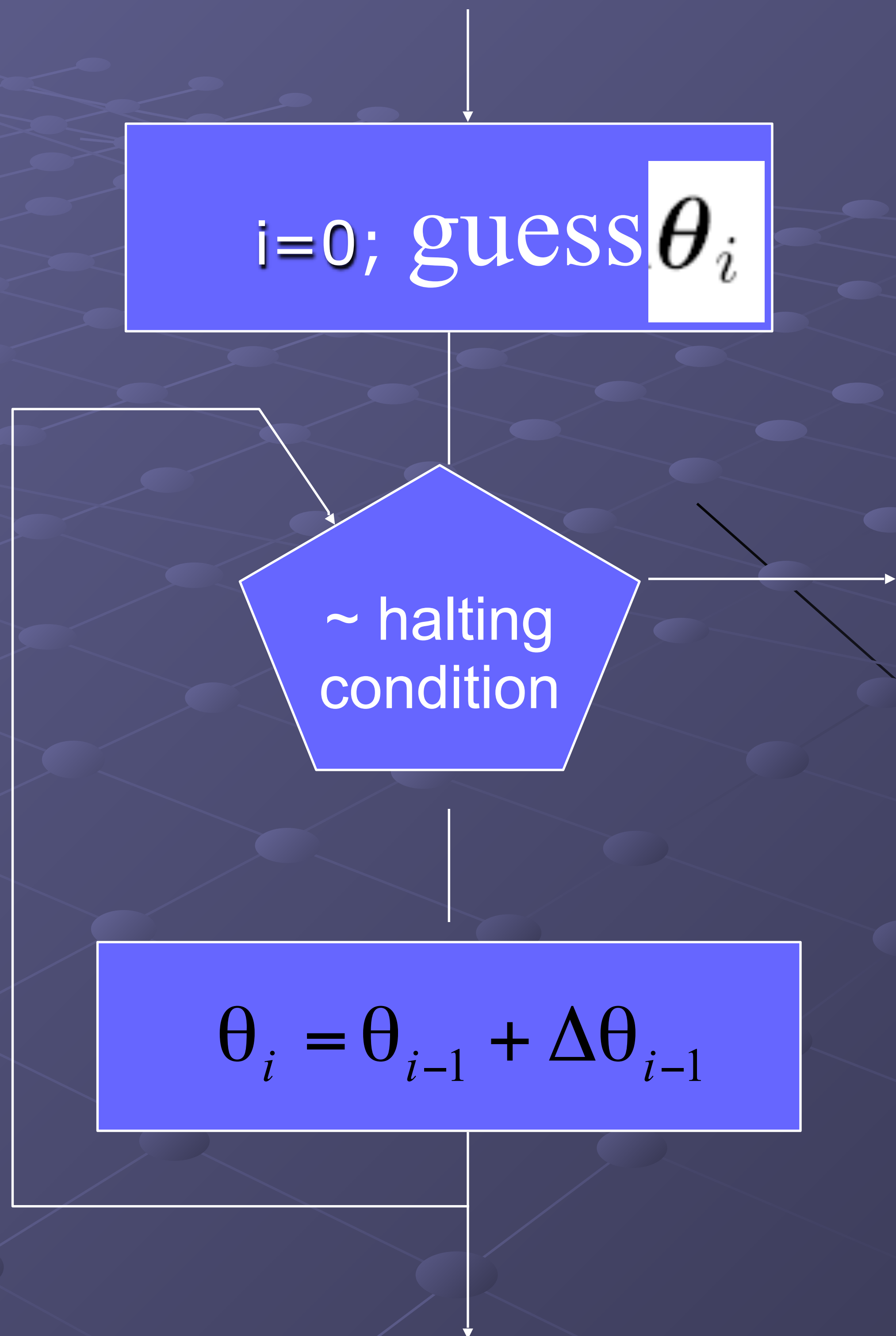
Iterative approach

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

Flow Chart


i=0; guess θ_i
~ halting condition
$\theta_i = \theta_{i-1} + \Delta\theta_{i-1}$

Objective function

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

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

Error of
the ith paired data

Desired
output

Network
output

Gradient method

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

- Derive the derivative of E_S with respect to θ
- Substitute θ to $\frac{dE_S(\theta)}{d\theta}$

Gradient

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

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

Derivation

$$y(t \mid \theta)$$

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

$$\theta = [\mathbf{a}_1^T \ \mathbf{a}_2^T \cdots \mathbf{a}_M^T \ b_1 \ b_2 \cdots b_M \ \mathrm{r}_0 \ \mathrm{r}_1 \ \mathrm{r}_2 \cdots \mathrm{r}_M]^T$$

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

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

$$= \left(\frac{dy(t\,|\,\boldsymbol{\theta})}{d\theta_1} \quad \frac{dy(t\,|\,\boldsymbol{\theta})}{d\theta_2} \quad \cdots \quad \cdots \quad \frac{dy(t\,|\,\boldsymbol{\theta})}{d\theta_{Md+2M+1}} \right)^T$$

$$\boldsymbol{\theta} = [\mathbf{a}_1^T \ \mathbf{a}_2^T \ \cdots \mathbf{a}_M^T \ b_1 \ b_2 \cdots b_M \ \mathbf{r}_0 \ \mathbf{r}_1 \ \mathbf{r}_2 \cdots \mathbf{r}_M]^T$$

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

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

$$= \left(\frac{dy(t\,|\,\boldsymbol{\theta})}{d\mathbf{a}_1^T} \quad \frac{dy(t\,|\,\boldsymbol{\theta})}{d\mathbf{a}_2^T} \quad \cdots \frac{dy(t\,|\,\boldsymbol{\theta})}{db_1^T} \quad \cdots \frac{dy(t\,|\,\boldsymbol{\theta})}{dr_1^T} \cdots \quad \frac{dy(t\,|\,\boldsymbol{\theta})}{dr_M^T} \right)^T$$

$$\boldsymbol{\theta} = [\mathbf{a}_1^T \ \mathbf{a}_2^T \ \cdots \mathbf{a}_M^T \ b_1 \ b_2 \ \cdots b_{\mathrm{M}} \ \mathrm{r}_0 \ \mathrm{r}_1 \ \mathrm{r}_2 \ \cdots \mathrm{r}_{\mathrm{M}}]^T$$

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

$$y(t \mid \theta)$$

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

$$\frac{dy(t \mid \boldsymbol{\theta})}{d\mathbf{a}_k}$$

$$= \frac{d}{d\mathbf{a}_k} \sum_{m=1}^{M} r_m \tanh(\mathbf{a}_m^T \mathbf{x}[t] + b_m) + r_0$$

$$= r_k (1 - \tanh(\mathbf{a}_k^T \mathbf{x}[t] + b_k)^2) \mathbf{x}[t]$$

$$\frac{dy(t\mid\boldsymbol{\theta})}{db_k}$$

$$= \frac{d}{db_k}\sum_{m=1}^{M} r_m \tanh(\mathbf{a}_m^T \mathbf{x}[t] + b_m) + r_0$$

$$= r_k(1 - \tanh(\mathbf{a}_k^T \mathbf{x}[t] + b_k)^2)$$

$$\frac{dy(t\,|\,\boldsymbol{\theta})}{dr_k}$$

$$= \frac{d}{dr_k}\sum_{m=1}^{M} r_m \tanh(\mathbf{a}_m^T\mathbf{x}[t] + b_m) + r_0$$

$$= \tanh(\mathbf{a}_k^T\mathbf{x}[t] + b_k)$$

Error

derivative

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

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

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

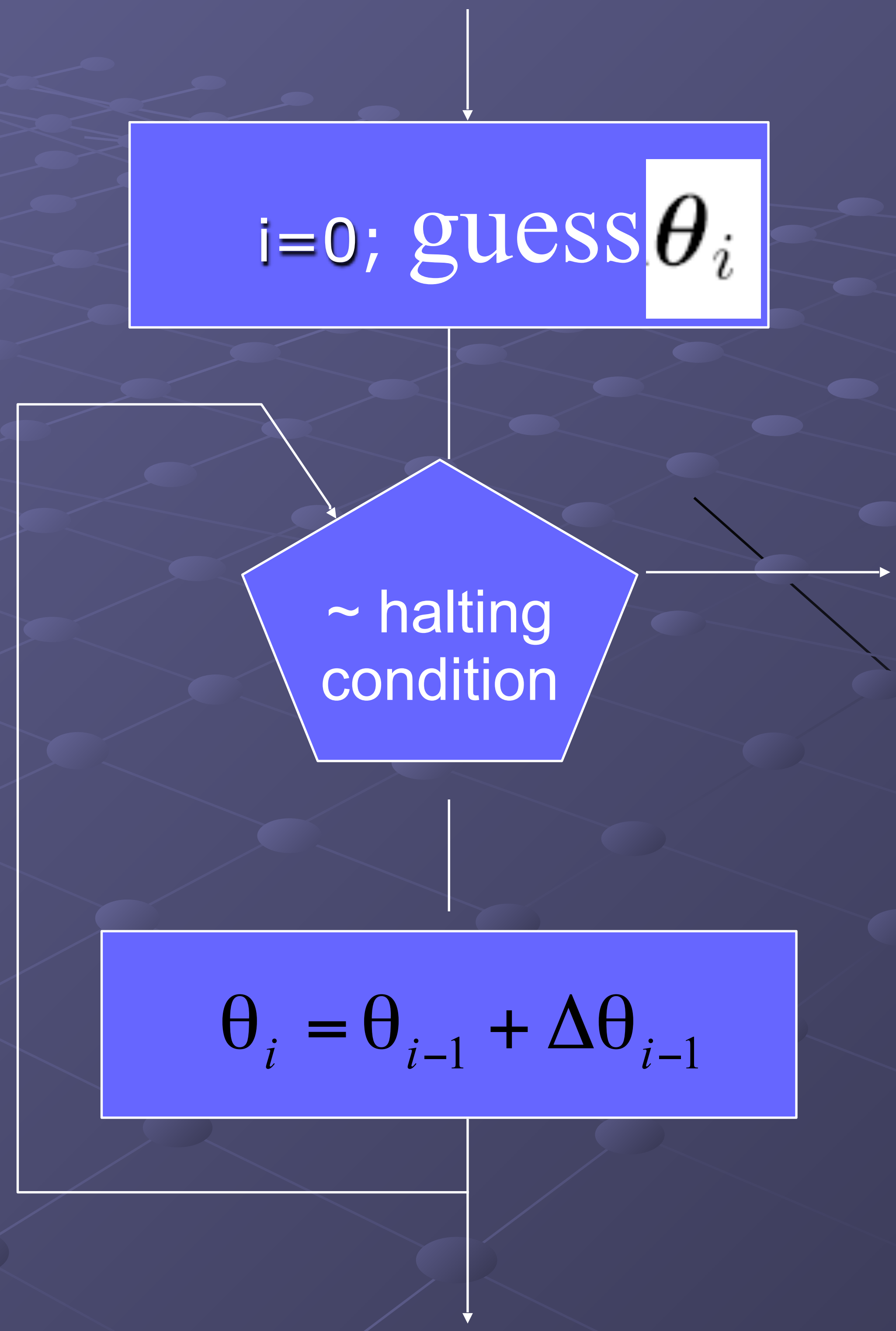
$$\boldsymbol{\Delta\theta}_i \propto -\frac{dE_S(\boldsymbol{\theta})}{d\boldsymbol{\theta}}\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_i}$$

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

Exercise

- Gradient descent method for learning MLP networks
- Gradient descent method for learning RBF networks

Flow Chart


i=0; guess θ_i
~ halting condition
$\theta_i = \theta_{i-1} + \Delta\theta_{i-1}$

Calculate the change

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

$$= \frac{-1}{N}\sum_{t=1}^{N}\varepsilon(t,\boldsymbol{\theta}_i)\psi(t,\boldsymbol{\theta}_i)$$

- Prepare a, b, r from given theta
- Substitute a,b,r to $\varphi(t,\boldsymbol{\theta})=\frac{dy(t|\boldsymbol{\theta})}{d\boldsymbol{\theta}}$
- Substitute a,b,r to $\varepsilon(t,\boldsymbol{\theta}) = y[t] - y(t|\boldsymbol{\theta})$

for all t

Substitute a,b,r to $\varphi(t,\boldsymbol{\theta}) = \dfrac{dy(t|\boldsymbol{\theta})}{d\boldsymbol{\theta}}$

- Substitute current a,b,r to

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

$$= \left(\frac{dy(t\,|\,\boldsymbol{\theta})}{d\mathbf{a}_1^T} \quad \frac{dy(t\,|\,\boldsymbol{\theta})}{d\mathbf{a}_2^T} \quad \cdots \frac{dy(t\,|\,\boldsymbol{\theta})}{db_1^T} \quad \cdots \frac{dy(t\,|\,\boldsymbol{\theta})}{dr_1^T} \cdots \quad \frac{dy(t\,|\,\boldsymbol{\theta})}{dr_M^T} \right)^T$$

Substitute a,b,r to

$$\frac{dy(t|\boldsymbol{\theta})}{d\mathbf{a}_k} = r_k(1 - \tanh(\mathbf{a}_k^T\mathbf{x}[t] + b_k)^2)\mathbf{x}[t]$$

$$\frac{dy(t|\boldsymbol{\theta})}{db_k} = r_k(1 - \tanh(\mathbf{a}_k^T\mathbf{x}[t] + b_k)^2)$$

$$\frac{dy(t|\boldsymbol{\theta})}{dr_k} = \tanh(\mathbf{a}_k^T\mathbf{x}[t] + b_k)$$

for all k

Newton-Gauss Method

Second order expansion

Linear expansion of errors

Error : nonlinear function of θ

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

Linear expansion at $\theta = \theta_i$

$$\begin{aligned}\widetilde{\varepsilon}_i(t,\boldsymbol{\theta}) &= \varepsilon(t,\boldsymbol{\theta}_i) + (\boldsymbol{\theta} - \boldsymbol{\theta}_i)^T \frac{d\varepsilon(t,\boldsymbol{\theta})}{d\boldsymbol{\theta}}\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_i} \\ &= \varepsilon(t,\boldsymbol{\theta}_i) - (\boldsymbol{\theta} - \boldsymbol{\theta}_i)^T \frac{dy(t|\boldsymbol{\theta})}{d\boldsymbol{\theta}}\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_i}\end{aligned}$$

Linear expansion of error

Let

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

$$\widetilde{\varepsilon}_i(t, \boldsymbol{\theta}) = \varepsilon(t, \boldsymbol{\theta}_i) - (\boldsymbol{\theta} - \boldsymbol{\theta}_i)^T \psi(t, \boldsymbol{\theta}_i)$$

is a linear function of θ

Criteria for Newton-Gauss

Mean square error

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

Quadratic form

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

Quadratic form

Expansion at $\theta = \theta_i$

$$L_i(\boldsymbol{\theta}) = L_i(\boldsymbol{\theta}_i) + \nabla(\boldsymbol{\theta}_i)^T(\boldsymbol{\theta} - \boldsymbol{\theta}_i) + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_i)^T R(\boldsymbol{\theta}_i)(\boldsymbol{\theta} - \boldsymbol{\theta}_i),$$

gradient $\nabla(\boldsymbol{\theta}_i) = \frac{-1}{N}\sum_{t=1}^{N} \varepsilon(t, \boldsymbol{\theta}_i)\psi(t, \boldsymbol{\theta}_i)$

Gauss-Newton Hessian

$$R(\boldsymbol{\theta}_i) = \frac{1}{N}\sum_{t=1}^{N} \psi(t, \boldsymbol{\theta}_i)\psi^T(t, \boldsymbol{\theta}_i)$$

Newton-Gauss method

$$\frac{dL_i(\boldsymbol{\theta})}{d\boldsymbol{\theta}} = \nabla(\boldsymbol{\theta}_i) + R(\boldsymbol{\theta}_i)(\boldsymbol{\theta} - \boldsymbol{\theta}_i) = 0$$

$$\Delta\theta_i = \theta - \theta_i$$

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

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

$$\Rightarrow \quad \Delta\theta_i = -(R(\theta_i))^{-1}\nabla(\theta_i)$$

Levenberg-Marquardt method

$\Delta\theta_i$ is determined by

$$(R(\boldsymbol{\theta}_i) + \lambda \mathbf{I})\boldsymbol{\Delta\theta}_i = -\boldsymbol{\nabla}(\boldsymbol{\theta}_i)$$

$$\lambda = 0$$

$$(R(\boldsymbol{\theta}_i) + \lambda \mathbf{I})\boldsymbol{\Delta\theta}_i = -\boldsymbol{\nabla}(\boldsymbol{\theta}_i)$$

↓

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

Newton-Gauss method

Sufficiently large λ

$$(R(\boldsymbol{\theta}_i) + \lambda \mathbf{I})\boldsymbol{\Delta}\boldsymbol{\theta}_i = -\boldsymbol{\nabla}(\boldsymbol{\theta}_i)$$

$$\Downarrow$$

$$\boldsymbol{\Delta}\boldsymbol{\theta}_i \propto -\boldsymbol{\nabla}(\boldsymbol{\theta}_i)$$

Gradient method

Control λ

Current parameter

Next parameter

$$\alpha_i = \frac{E_S(\boldsymbol{\theta}_i) - E_S(\boldsymbol{\theta}_i + \boldsymbol{\Delta\theta}_i)}{E_S(\boldsymbol{\theta}_i) - L_i(\boldsymbol{\theta}_i + \boldsymbol{\Delta\theta}_i)}$$

$$\frac{\text{Actual cost reduction}}{\text{Predicted cost reduction}}$$

High α_i

Reduce λ

improve efficiency

Force to Newton-Gauss method

Low α_i

Increase λ

Improve reliability

Force to gradient method

Heuristic adaption

(a) If $\alpha_i > 0.75$, $\lambda \leftarrow 0.5\lambda$.

(b) If $\alpha_i < 0.25$, $\lambda \leftarrow 2\lambda$.

LM method

1. Initialize θ_i, i=0 and set λ
2. Calculate $\nabla(\boldsymbol{\theta}_i)$ and $R(\boldsymbol{\theta}_i)$ and $\Delta\theta_i$
3. Update network parameters

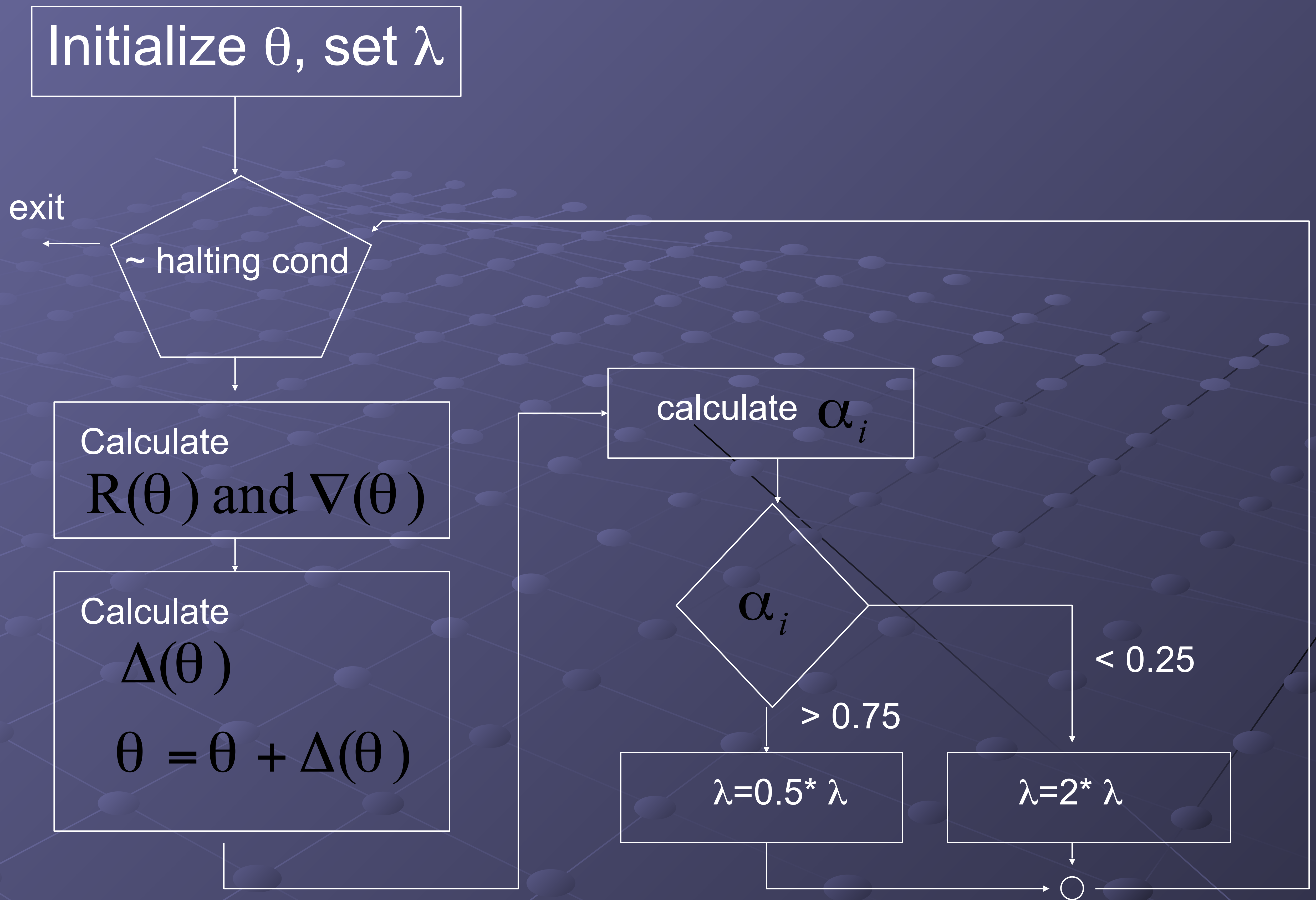
$$\theta_{i+1} = \theta_i + \Delta\theta_i$$

4. Calculate α_i
5. Update λ

   (a) If $\alpha_i > 0.75$, $\lambda \leftarrow 0.5\lambda$.

   (b) If $\alpha_i < 0.25$, $\lambda \leftarrow 2\lambda$.

6. If halting condition hold, exit otherwise go to step 2


Initialize θ, set λ
exit
~ halting cond
Calculate
R(θ) and ∇(θ)
Calculate
Δ(θ)
θ = θ + Δ(θ)
calculate α_i
α_i
< 0.25
> 0.75
λ=0.5* λ
λ=2* λ

Project

Learning RBF by the LM method

1. (30 pt) Derivation

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

2. (140 pt) Matlab codes
3. (50 pt) Testing

Derivative

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

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

$$\frac{dy(t|\theta)}{d\theta} = [\frac{dy(t|\theta)}{d\theta_1}, \cdots, \frac{dy(t|\theta)}{d\theta_{M*d+2M+1}}]^T$$

Derivative

$$\frac{dy(t|\theta)}{d\mathbf{u}_m} = ?$$

$$\frac{dy(t|\theta)}{d\sigma_m} = ?$$

$$\frac{dy(t|\theta)}{dw_m} = ?$$

Matlab coding

- (30 pt)Main program
- Matlab Functions
  a. (10 pt)Calculate $E_S(\boldsymbol{\theta}_i)$

  b. (10 pt)Calculate $\mathrm{L}_i(\boldsymbol{\theta}_i)$

Matlab coding

- Matlab functions
  - (10 pt) Calculate $\frac{dy(t|\theta)}{d\mathbf{u}_m}\Big|_{\theta=\theta_i}$
  - (10 pt) Calculate $\frac{dy(t|\theta)}{d\sigma_m}\Big|_{\theta=\theta_i}$
  - (10 pt) Calculate $\frac{dy(t|\theta)}{dw_m}\Big|_{\theta=\theta_i}$

Matlab coding

- Matlab functions
  - (20 pt) Calculate $\nabla(\theta_i)$
  - (20 pt) Calculate $R(\theta_i)$
  - (20 pt) Calculate $G(\mathbf{x} \mid \theta)$

Test

- Give two examples to test your matlab codes for learning RBF networks by Levenberg-Marquardt method