# Learning Multilayer Potts perceptrons for function approximation

Jiann-Ming Wu

Department of Applied Mathematics

National Dong Hwa University

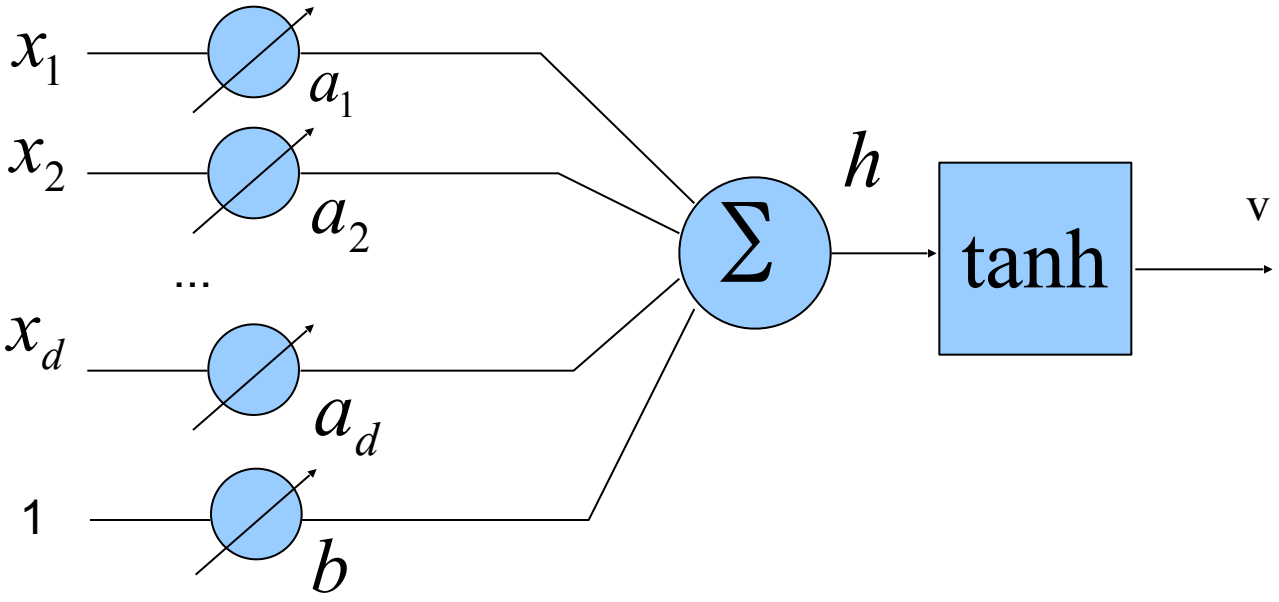
# Outline

- Multilayer Perceptrons (MLP)
  - Perceptrons
  - Post-tanh projections
- Function approximation
- Multilayer Potts perceptrons (MLPotts)
  - Potts perceptrons
  - Post-nonlinear projections
- Levenberg-Marquardt learning of MLPotts networks
- Conclusions

# Perceptrons

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

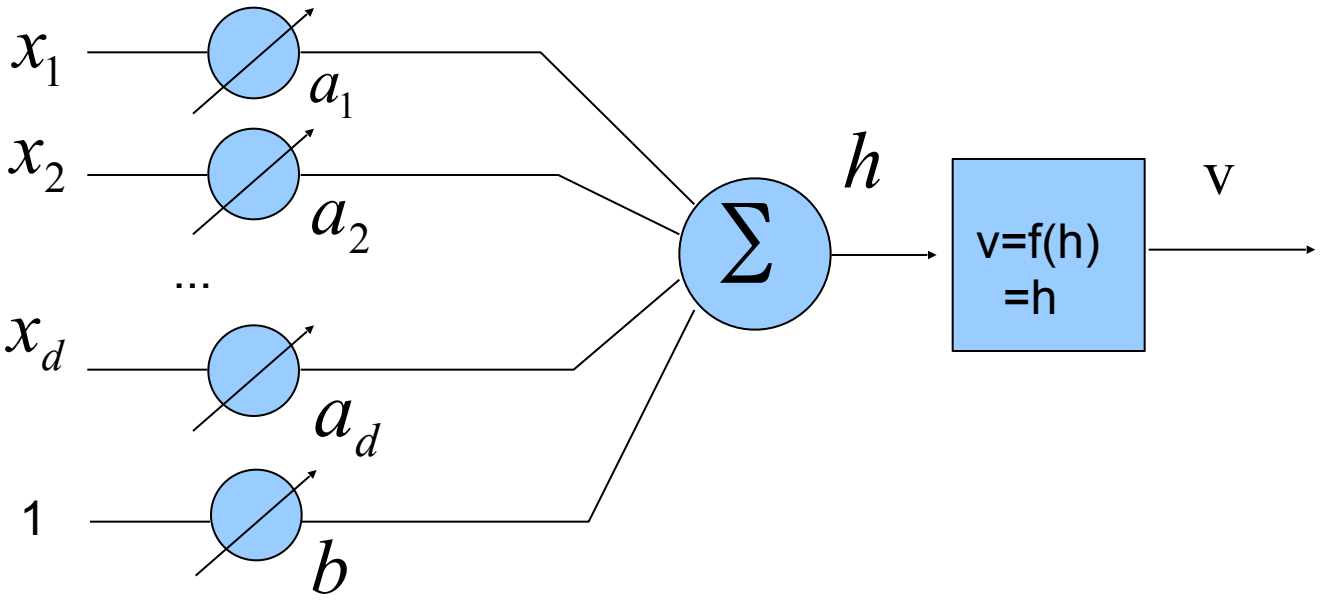
$$v = \tanh(h = a_1 x_1 + a_2 x_2 + \ldots + a_d x_d + b)$$



# Linear projection

- Reduce to linear projection when tanh is removed

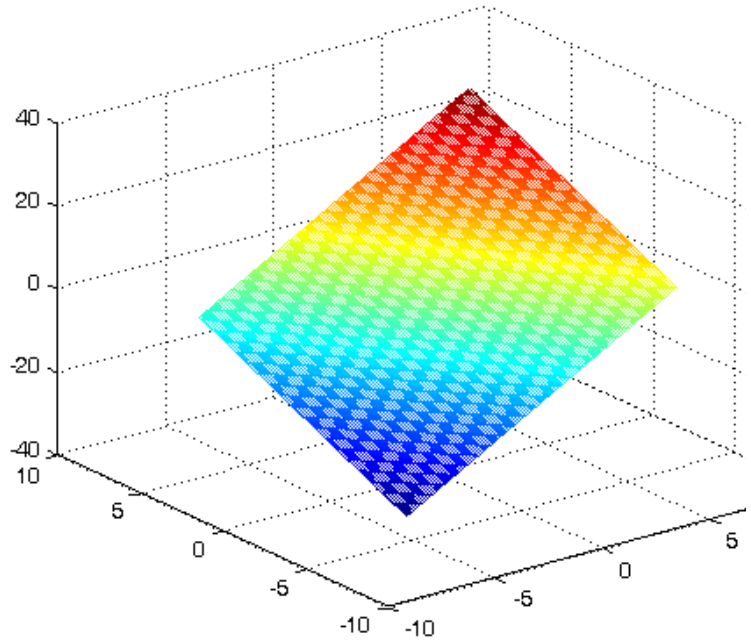
$$v = h = a_1 x_1 + a_2 x_2 + \ldots + a_d x_d + b$$

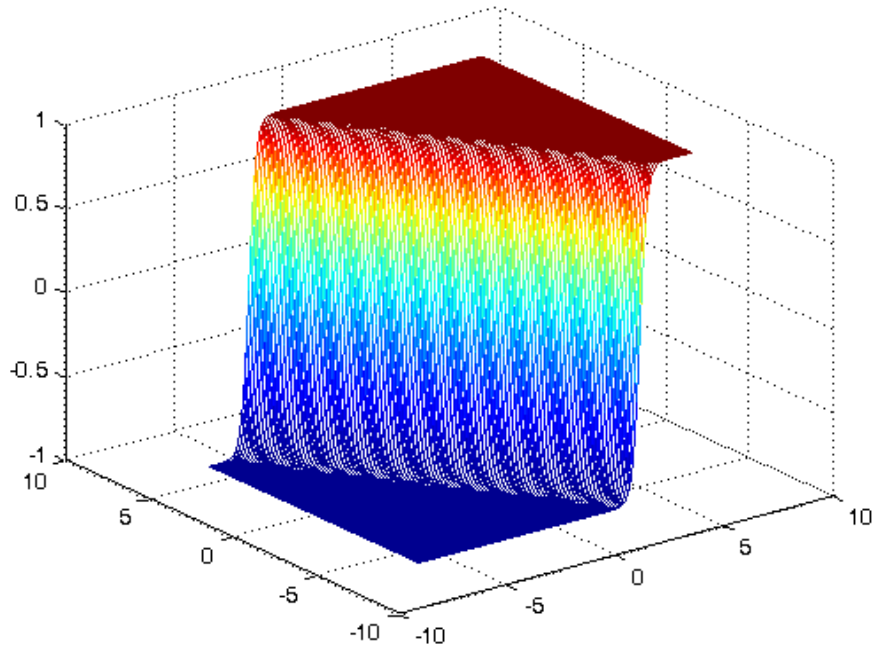


# Linear projection vs post-tanh projection

d=2

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



letters to nature

*Nature* **323**, 533 - 536 (09 October 1986); doi:10.1038/323533a0

# Learning representations by back-propagating errors

DAVID E. RUMELHART*, GEOFFREY E. HINTON† & RONALD J. WILLIAMS*

*Institute for Cognitive Science, C-015, University of California, San Diego, La Jolla, California 92093, USA
†Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Philadelphia 15213, USA
†To whom correspondence should be addressed.

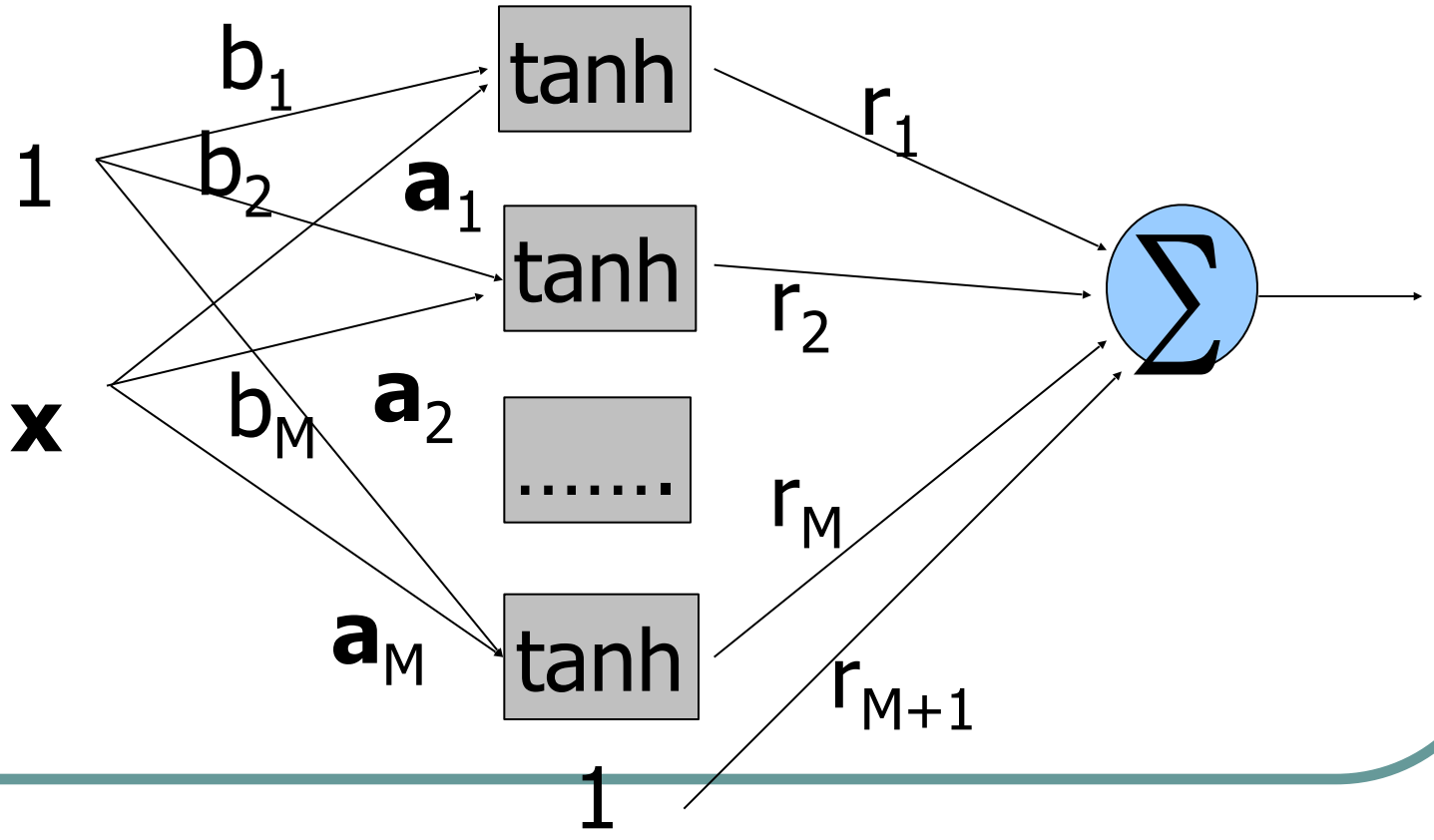
**We describe a new learning procedure, back-propagation, for networks of neurone-like units. The procedure repeatedly adjusts the weights of the connections in the network so as to minimize a measure of the difference between the actual output vector of the net and the desired output vector. As a result of the weight adjustments, internal 'hidden' units which are not part of the input or output come to represent important features of the task domain, and the regularities in the task are captured by the interactions of these units. The ability to create useful new features distinguishes back-propagation from earlier, simpler methods such as the perceptron-convergence procedure[1].**

## References

1. Rosenblatt, F. *Principles of Neurodynamics* (Spartan, Washington, DC, 1961).

# Multiple perceptrons

Multilayer perceptrons (MLP)
(Rumelhart, 1986)



# Network Function

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

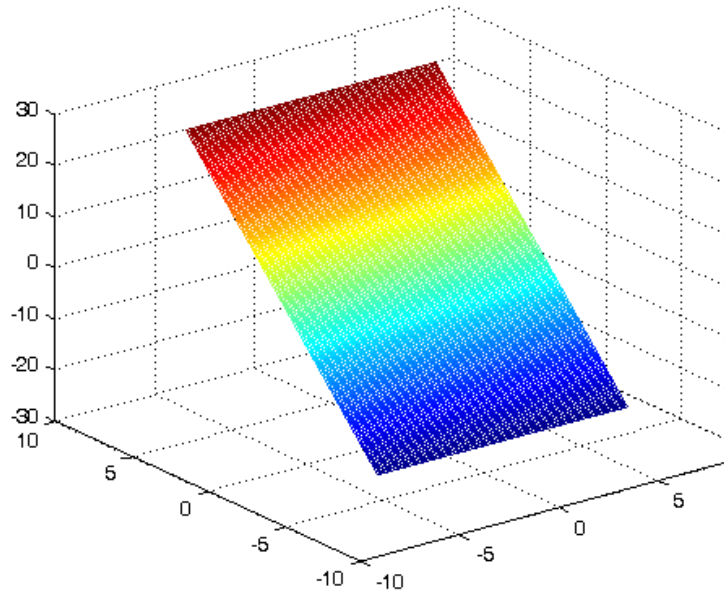
$$\theta = \{\mathbf{a}_m\} \cup \{b_m\} \cup \{r_m\}$$

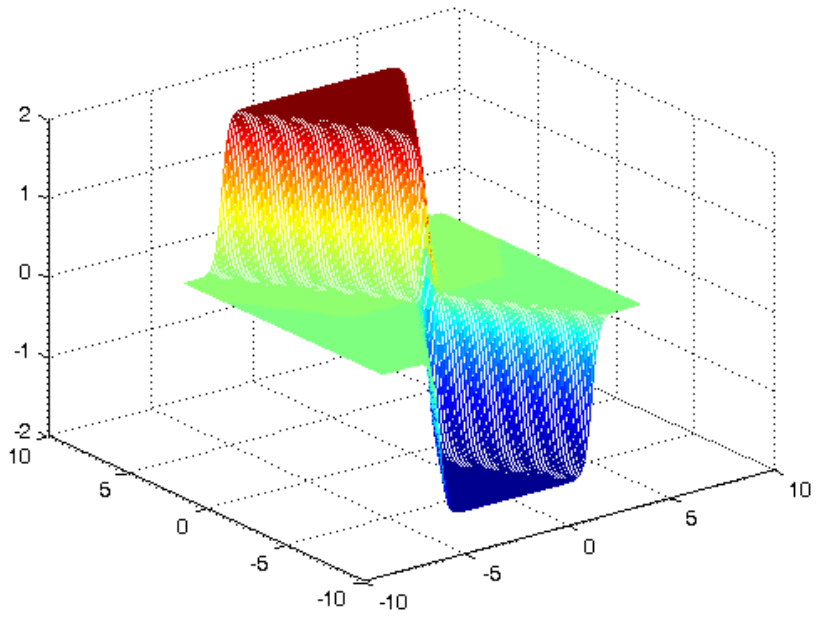
# Multiple post-tanh projections

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)



# 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

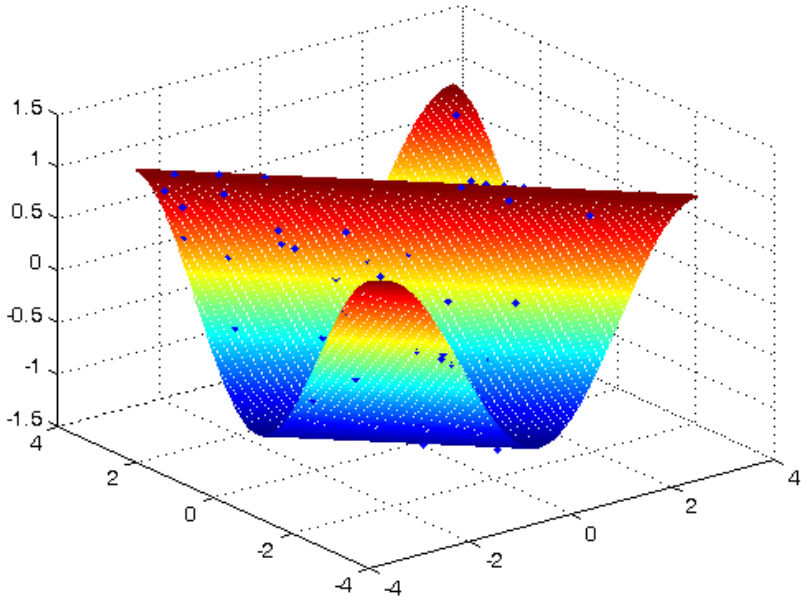
$$y[t] = \mathrm{F}(\mathbf{x}[t]) + noise$$

# Paired data

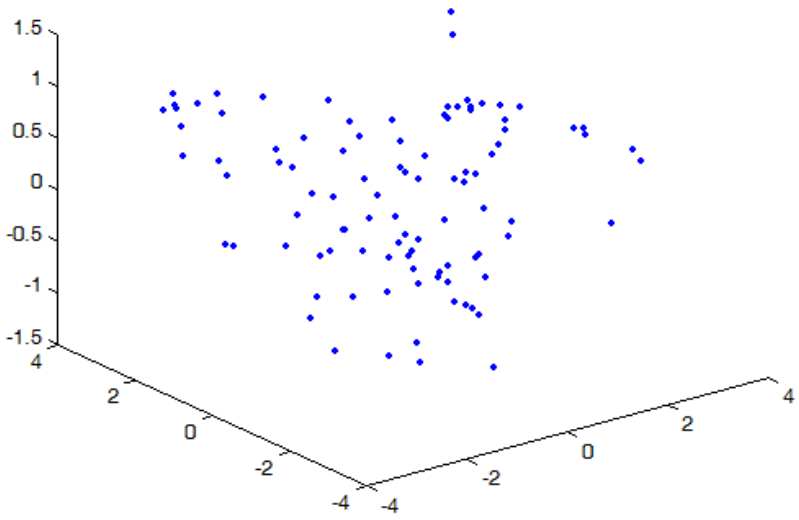
$$d = 2$$

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

Target function




Paired data



# 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

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

# Unconstrained optimization

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

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

# 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)

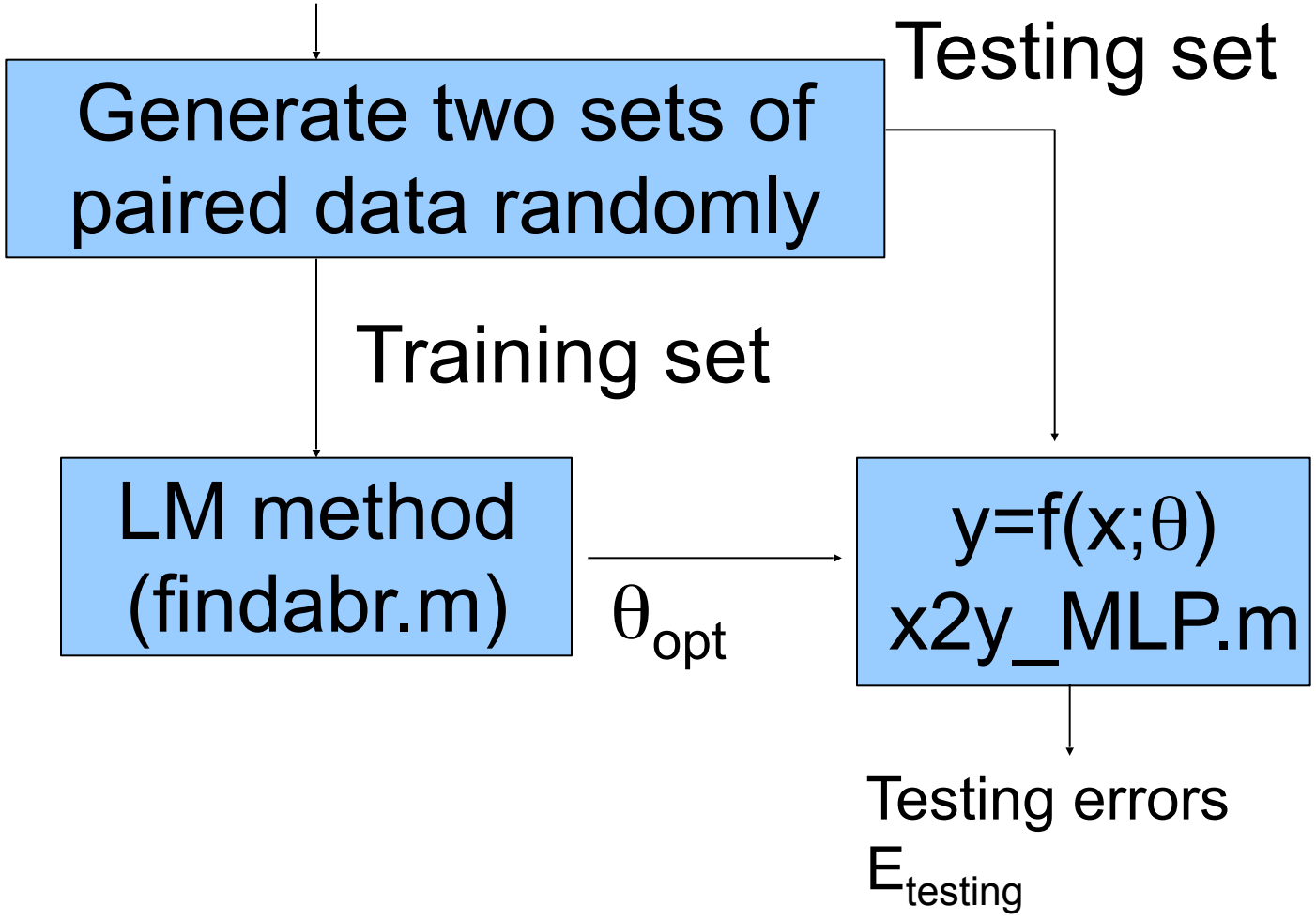
TABLE III
TARGET FUNCTIONS

| 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



# Multilayer Potts perceptrons

- Perceptrons

  Rosenblatt (1962), Widrow (1962)
- Adalines

  Widrow, 1962
- Generalized adalines

  Wu 2001, Wu et al 2006
- Multilayer Potts perceptrons

  Wu 2008

# Why tanh?

- Spin random variable $s \in \{+1, -1\}$
- s depends on h
- Conditional probability of s to h is defined by

$$\Pr(s \mid h) \propto \exp(hs)$$

- Then

$$v = E[s \mid h] = \tanh(h)$$

# Potts random variable

- Potts random variables $\delta$
- $\delta \in \{e_1, e_2, \ldots, e_K\}$
- K states
- $\mathbf{e_k}$ : a unitary vector with the kth bit one and others zeros

# Potts perceptrons

- Conditional probability of $\delta$ to h is defined by

$$\Pr(\boldsymbol{\delta} = \mathbf{e}_k \mid h) \propto q(h; c_k, \sigma^2)$$

$q$ denotes the normal pdf

with mean $c_k$ and variance $\sigma^2$

$$\Pr(\boldsymbol{\delta} = \mathbf{e}_k \mid h) \propto q(h; c_k, \sigma^2)$$

$$\Pr(\boldsymbol{\delta} = \mathbf{e}_k \mid h) \equiv q_k = \frac{q(h; c_k, \sigma^2)}{\sum_{j=1}^{K} q(h; c_j, \sigma^2)}$$

# Multi-state transfer function

$$\Pr(\boldsymbol{\delta} = \mathbf{e}_k \mid h) = q_k$$

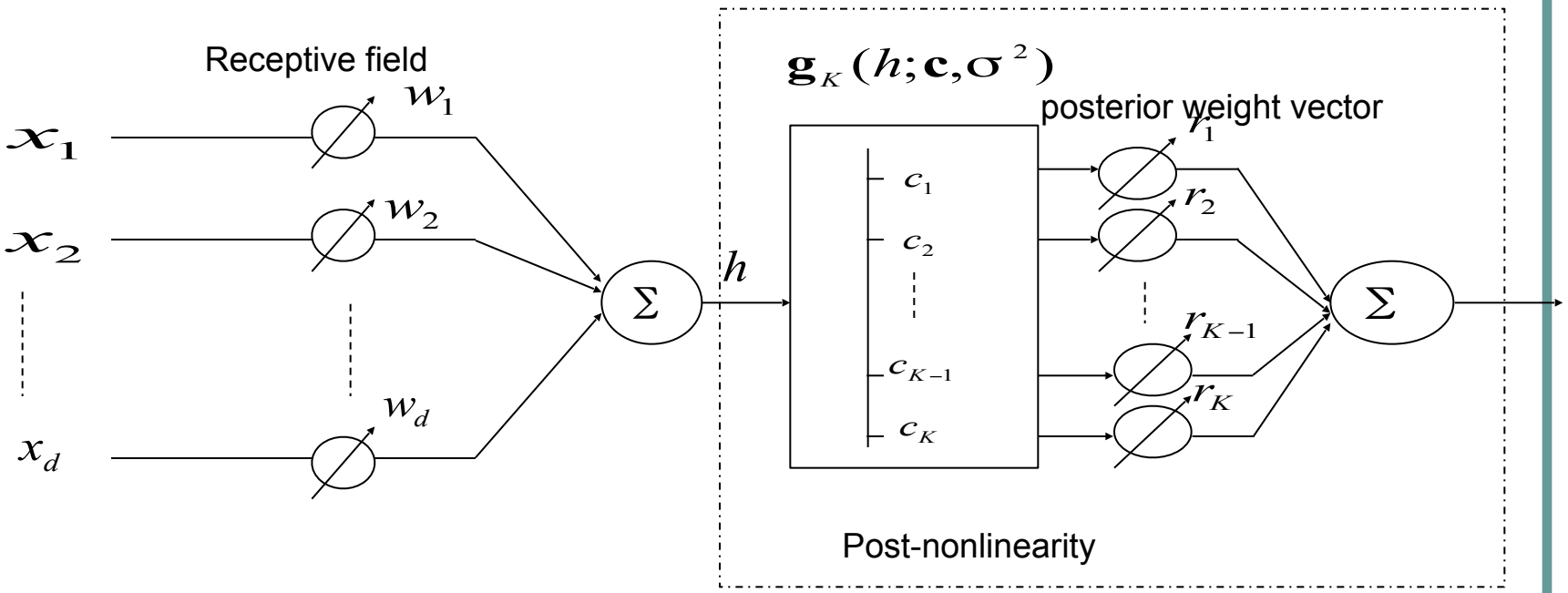
$$\mathbf{g}(h; \mathbf{c}, \sigma^2) \equiv \langle \boldsymbol{\delta} \mid h \rangle = \sum_{k=1}^{K} q_k \mathbf{e}_k$$

$$\Pr(\boldsymbol{\delta} = \mathbf{e}_k \mid h) \equiv q_k = \frac{q(h; c_k, \sigma^2)}{\sum_{j=1}^{K} q(h; c_j, \sigma^2)}$$

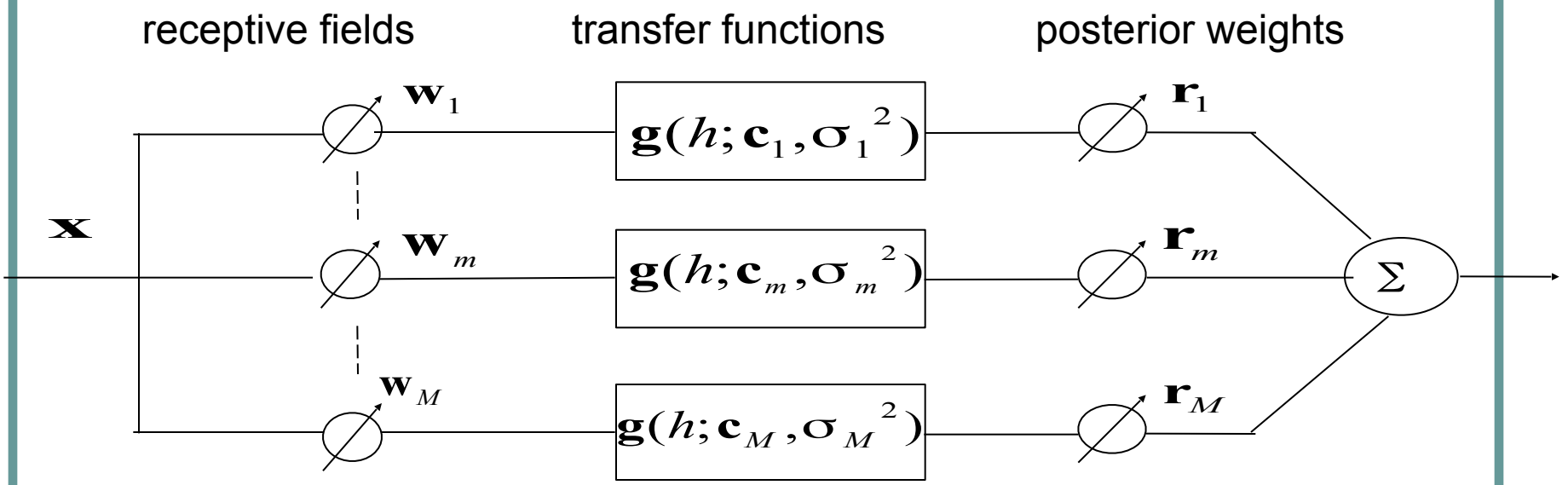
$$\mathbf{g}(h; \mathbf{c}, \sigma^2) \equiv \langle \boldsymbol{\delta} \mid h \rangle = \sum_{k=1}^{K} q_k \mathbf{e}_k$$

$$= \left[ q_1, \square, q_K \right]^T$$

# A weighted Potts perceptron

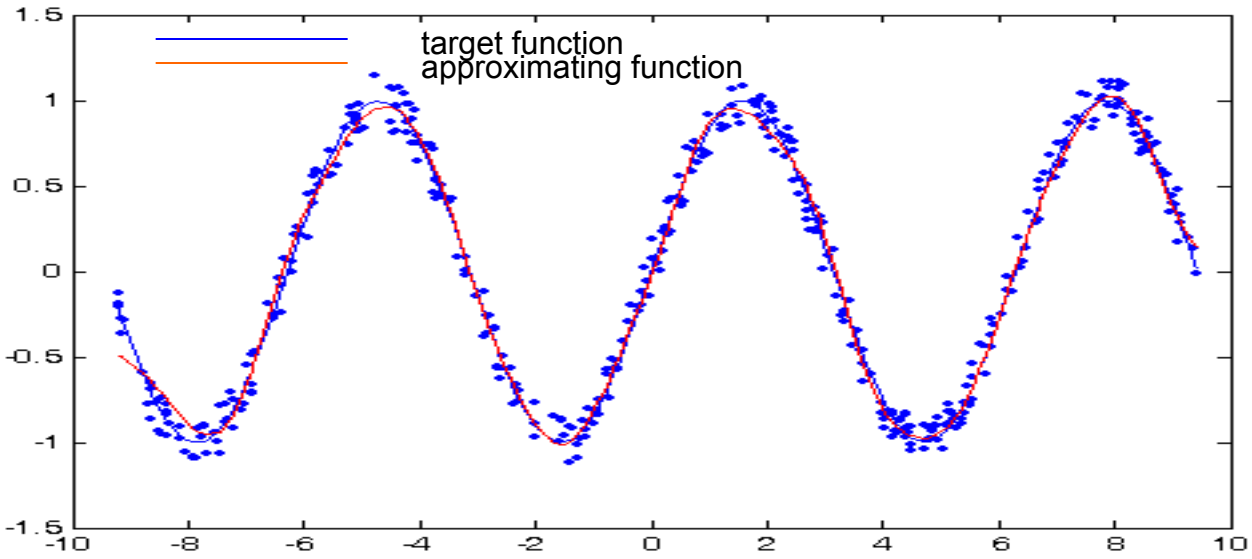


# Multiple Potts perceptrons



# Levenberg-Marquardt (LM)

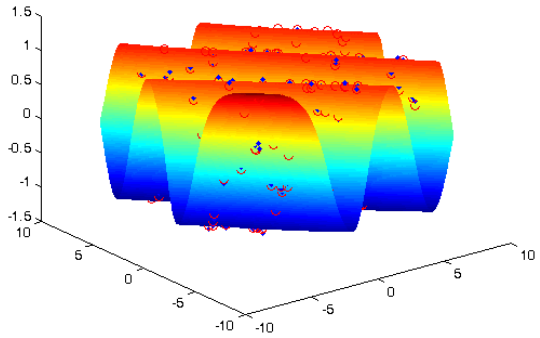
- Numerical methods for unconstrained optimization
  - Gradient descent (GD) method
  - Newton-Gauss (NG) method
  - LM method: a hybrid of GD and NG methods

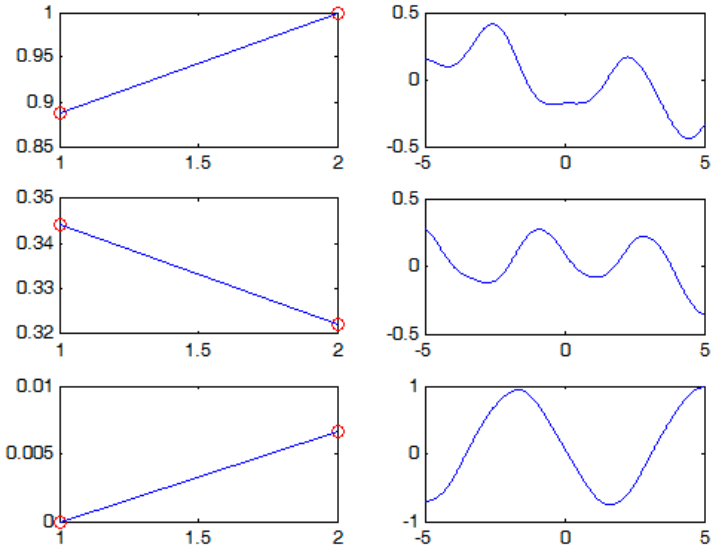

target function
approximating function
1.5
1
0.5
0
-0.5
-1
-1.5
-10
-8
-6
-4
-2
0
2
4
6
8
10

# MLPotts learning

$$y_2(\mathbf{x}) = \sin(\mathbf{a}_1^T \mathbf{x})$$

$$\mathbf{x} \in [-2\pi, 2\pi]^2$$



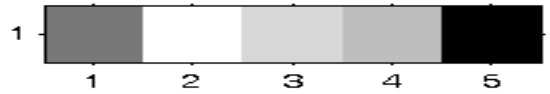


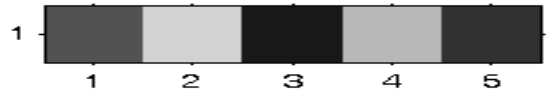
Target function: $y_2$
MLPotts(M=3,K=41)

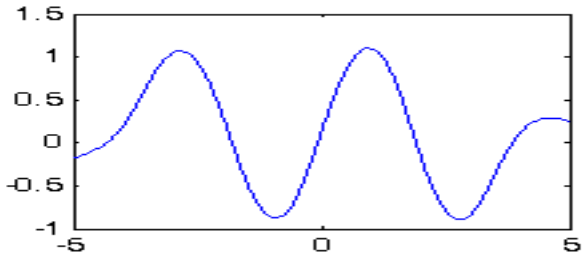
Mean square error(Testing cost): 0.003803

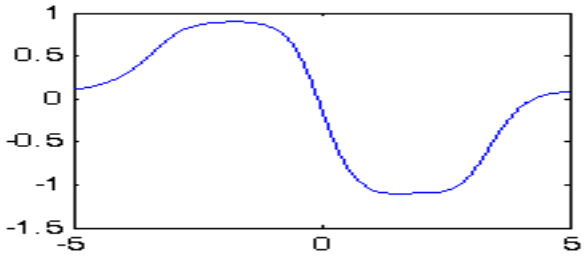
$$y_5(\mathbf{x}) = \sin(\mathbf{a}_1^T \mathbf{x}) + \tanh(\mathbf{a}_2^T \mathbf{x})$$

$$\mathbf{x} \in [-2\pi, 2\pi]^5$$



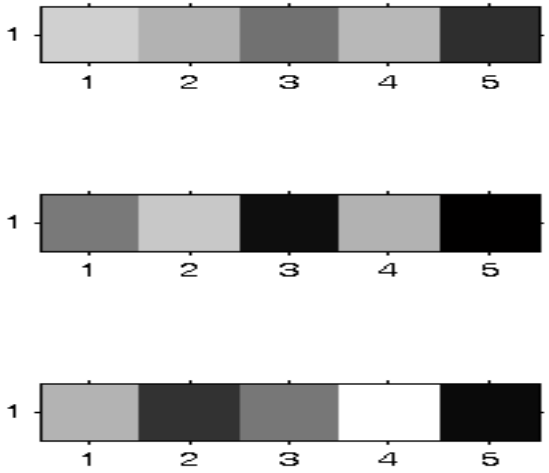



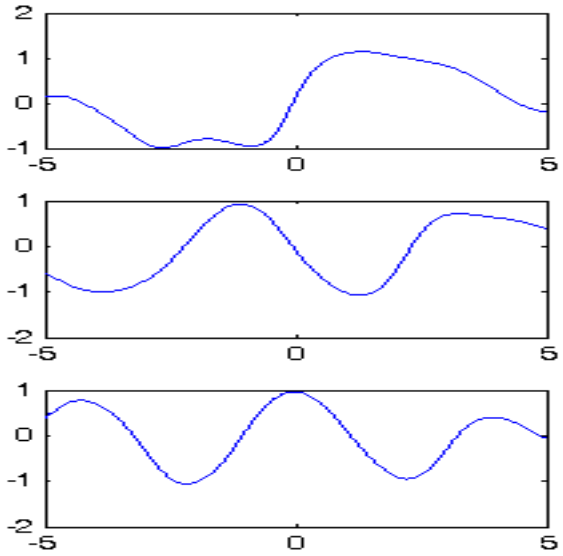




Target function: y5
MLPotts(M=2,K=31)

$$y_5(\mathbf{x}) = \sin(\mathbf{a}_1^T \mathbf{x}) + \tanh(\mathbf{a}_2^T \mathbf{x}) + \cos(\mathbf{a}_3^T \mathbf{x})$$

$$\mathbf{x} \in [-2\pi, 2\pi]^5$$

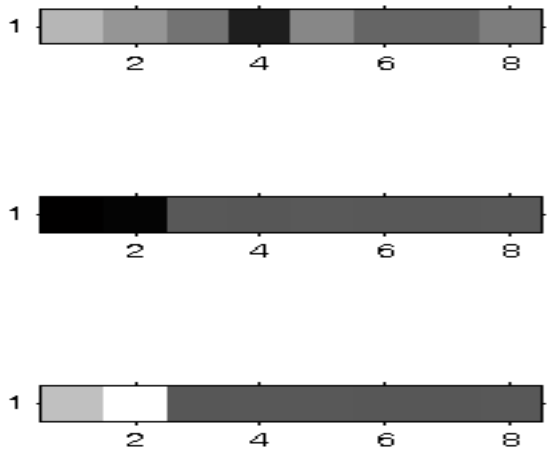


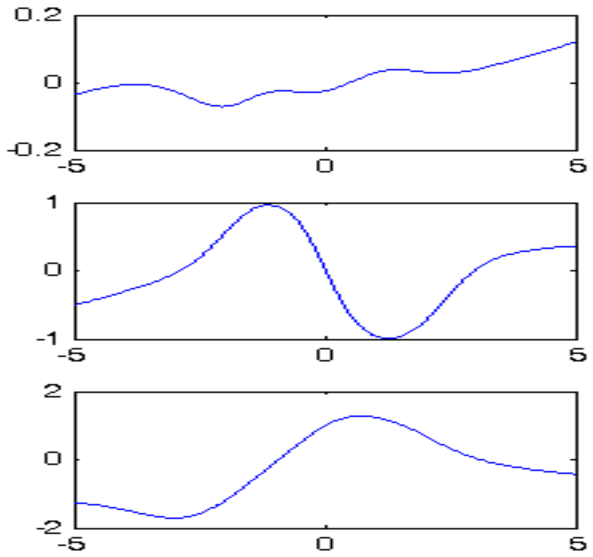




Target function: y6
MLPotts(M=3,K=31)

$$y_8(\mathbf{x}) = \sin(\mathbf{a}_1^T \mathbf{x}) + \tanh(\mathbf{a}_2^T \mathbf{x}) + \cos(\mathbf{a}_3^T \mathbf{x})$$

$$\mathbf{x} \in [-\pi, \pi]^8$$




Target function: y8
MLPotts(M=3,K=31)

Figure 7

# Numerical Simulations

(Wu 2008)

| | target function | domain | sample size $N$ | noise level |
|---|---|---|---|---|
| $y_1$ | $\sin(x)$ | $[-3\pi, 3\pi]$ | 400 | $[-0.15, 0.15]$ |
| $y_2$ | $\sin(a^T x)$ | $[-2\pi, 2\pi]^2$ | 300 | $[-0.15, 0.15]$ |
| $y_3$ | $\sin(a^T x)$ | $[-2\pi, 2\pi]^5$ | 300 | $[-0.10, 0.10]$ |
| $y_4$ | $\sin(a^T x)$ | $[-2\pi, 2\pi]^7$ | 400 | $[-0.05, 0.05]$ |
| $y_5$ | $\sin(a_1^T x) + \tanh(a_2^T x)$ | $[-2\pi, 2\pi]^5$ | 400 | $[-0.10, 0.10]$ |
| $y_6$ | $\sin(a_1^T z) + \tanh(a_2^T z) + \cos(a_3^T z)$ | $[-2\pi, 2\pi]^5$ | 400 | $[-0.10, 0.10]$ |
| $y_7$ | $\sin(a_1 z) + \tanh(a_2^T z) + \cos(a_3^T z)$ | $[-1.5\pi, 1.5\pi]^7$ | 400 | $[-0.10, 0.10]$ |
| $y_8$ | $\sin(a_1^T z) + \tanh(a_2^T z) + \cos(a_3^T z)$ | $[-\pi, \pi]^8$ | 400 | $[-0.10, 0.10]$ |

# Numerical simulations

| | network | | | | | mse(training) | | mse(testing) | | cpu time |
|---|---|---|---|---|---|---|---|---|---|---|
| Data set | type | M | $\overline{\text{M}}$ | K | d | mean | std | mean | std | mean |
| $S_1, T_1$ | MLP | 21 | - | - | 1 | 1.32e-2 | 5.09e-3 | 1.40e-2 | 5.47e-3 | 5.4e1 |
| | MLP | 31 | - | - | 1 | 1.04e-2 | 2.58e-3 | 1.10e-2 | 2.95e-3 | 1.4e2 |
| | MLPotts | 1 | 1.3 | 31 | 1 | 4.28e-3 | 6.61e-4 | 5.41e-3 | 9.79e-4 | 2.3e1 |
| | MLPotts | 3 | 3 | 31 | 1 | 4.14e-3 | 4.68e-4 | 4.65e-3 | 5.72e-4 | 2.6e1 |
| $S_2, T_2$ | MLP | 21 | - | - | 2 | 6.60e-3 | 1.47e-3 | 2.22e-2 | 8.29e-3 | 6.4e1 |
| | MLPotts | 1 | 1.9 | 21 | 2 | 4.06e-3 | 4.71e-4 | 6.48e-3 | 1.96e-4 | 1.9e1 |
| | MLPotts | 1 | 1.7 | 31 | 2 | 4.74e-3 | 6.93e-4 | 6.62e-3 | 1.66e-3 | 3.3e1 |
| $S_3, T_3$ | MLP | 21 | - | - | 5 | 6.72e-4 | 4.93e-4 | 6.83e-2 | 9.48e-2 | 4.3e2 |
| | MLPotts | 1 | 1.6 | 41 | 5 | 1.66e-3 | 1.76e-4 | 3.49e-3 | 1.99e-3 | 1.7e1 |
| $S_4, T_4$ | MLP | 21 | - | - | 7 | 7.43e-4 | 6.20e-4 | 1.48e-1 | 2.13e-1 | 1.1e3 |
| | MLPotts | 4 | 4.2 | 31 | 7 | 8.47e-4 | 2.33e-4 | 2.81e-3 | 2.14e-3 | 5.9e1 |

# Ratio of parameters for post-nonlinear

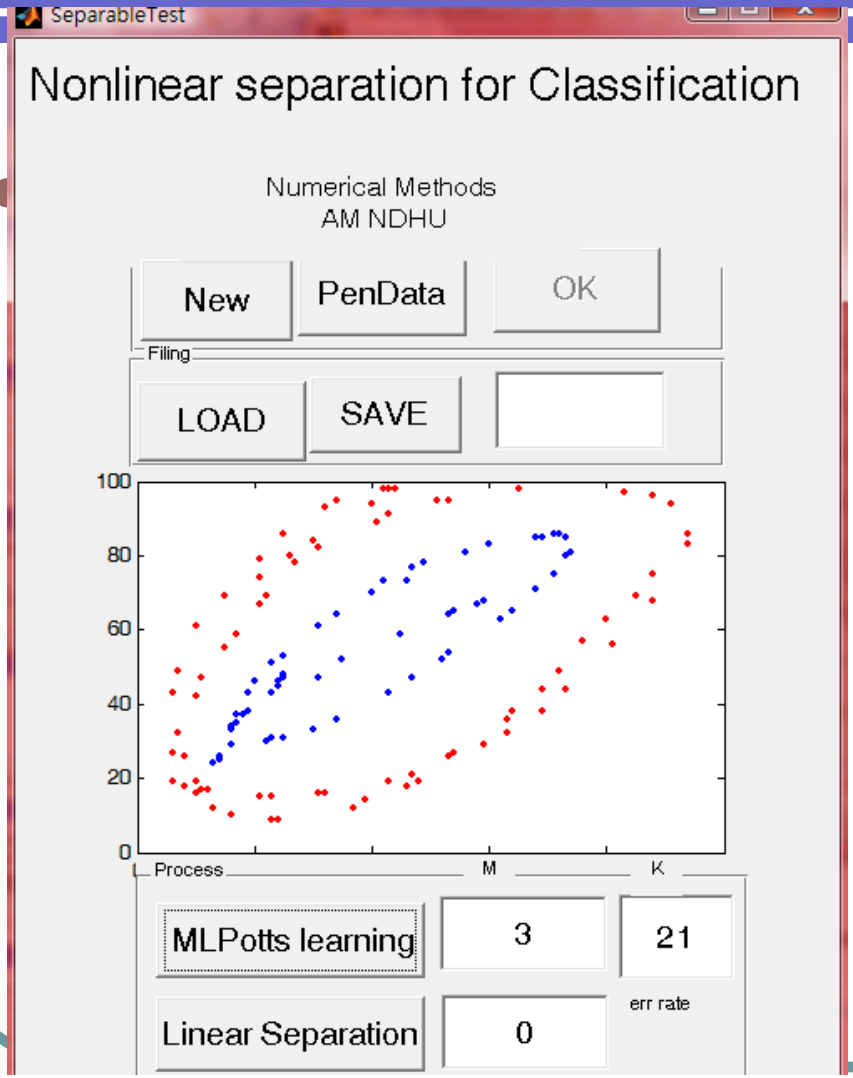
- d: data dimension
- Ratio of parameters utilized for post-nonlinear representations
  - MLP $$\frac{1}{1+d}$$
  - MLPotts $$\frac{2K+1}{1+2K+d}$$

Ex. d=8, K=31

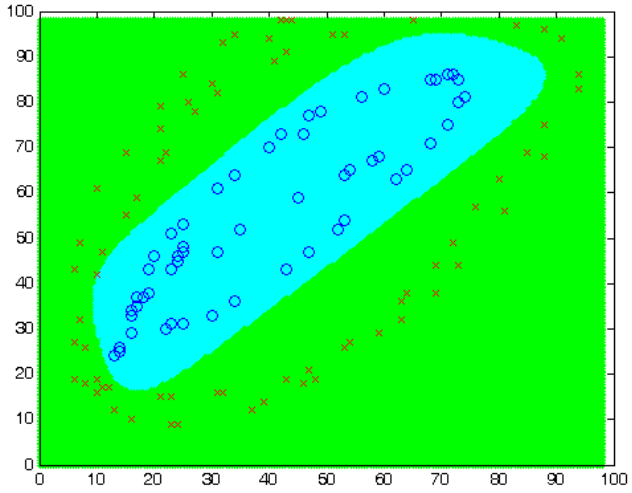
$\text{Ratio}_{\text{MLP}}$ : 12.50 %

$\text{Ratio}_{\text{MLPotts}}$ : 80%

# Classification




MATLAB
code2006\Apps\FunAppr\Separation



# Conclusions

- Advantages of MLPotts
  - Increasing the state number of Potts perceptrons is shown able to boost utilization of parameters for post-nonlinear representations
  - Perceptrons are special cases of Potts perceptrons
  - MLP networks are special cases of MLPotts networks
- LM method is effective for MLPotts learning
- MLPotts learning outperforms MLP learning for function approximation as well as time series prediction (2008 Wu)

# References


- Jiann-Ming Wu, S. J. Chiu, Independent component analysis using Potts models, IEEE Trans. On Neural Networks, Vol. 12, No. 2, March (2001,SCI)

- Jiann-Ming Wu, Z.H. Lin and Pei-Hsun Hsu, Function approximation using generalized adalines, IEEE Trans. On Neural Networks,Vol.17 No.3, 541-558, MAY 2006. (SCI)

- Jiann-Ming Wu, Multilayer Potts perceptrons with Levenberg-Marquardt learning, IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 19, NO. 12, DECEMBER 2008(SCI)