Discriminant analysis - Hills and Valleys

benchmarks

Machine Learning

- UCI Machine Learning Repository

hill-valley

- Hill-Valley.names
- Hill_Valley_visual_examples.jpg
- Hill_Valley_without_noise_Training.data
- Hill_Valley_without_noise_Testing.data
- Hill_Valley_with_noise_Training.data
- Hill_Valley_with_noise_Testing.data

Read Data

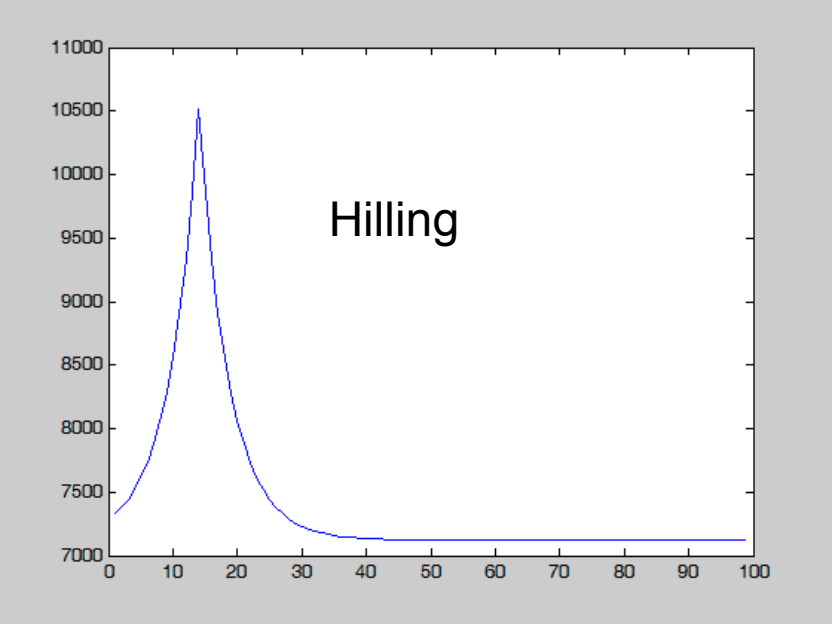
- Remove the first line of each file
- Read data

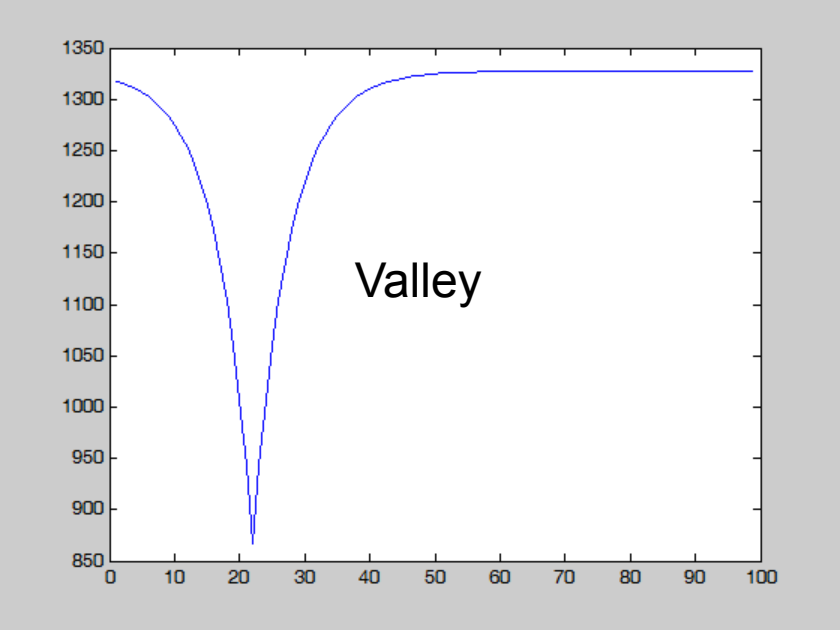
read_data.m

```
function [x,y]=read_data(fstr)
fid=fopen(fstr);
x=[];y=[];
while 1
    tline = fgetl(fid);
    if ~ischar(tline), break, end
    z=sscanf(tline,'%f,');
    x=[x;z(1:length(z)-1)'];
    y=[y z(length(z))];
end
fclose(fid);
```

Hilling and Valley

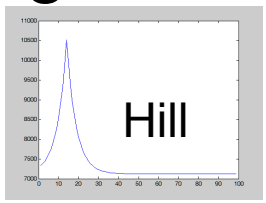
- 606 sequences in one of training data
-

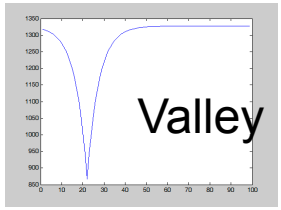




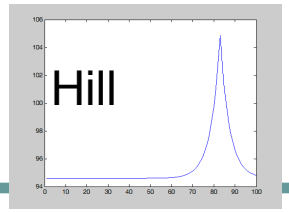
Traning vs Testing

- Training

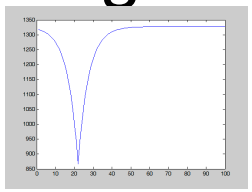
1  Hill

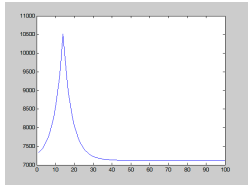
2  Valley

….

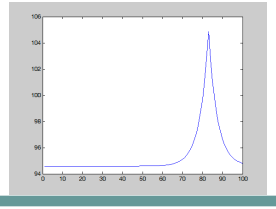
606  Hill

- testing

 Hill or Valley

 Hill or Valley

….

 Hill or Valley

Discriminant analysis

- Nearest classifier ?
- PottsNDA ?
- PCA+Classifier ?
- MLP Learning ?
- RBF Learning ?
- SVM learning ?

Nearest Classification

demo_Nearest_C.m

```
fstr='Hill_Valley_with_noise_Training.data';
[x,y]=read_data(fstr);
rescale=max(max(abs(x)));
A=x/rescale;
fstr='Hill_Valley_with_noise_Testing.data';
[B,testy]=read_data(fstr);
B=B/rescale;
for k=1:1
for i=1:size(B,1)
    x=B(i,:);
    c(i)=Nearest_C(A,y,x);
end
hit(k)=sum(c==testy)
rate(k)=hit(k)/size(testy,2)
end
```

```
hit =

   341


rate =

    0.5627
```

Structure of Pattern

- Time or temporal series
- Spatial pattern

Stochastic signal process

- Conditional Probability

$$p(x_t \mid x_{t-\tau}, \ldots, x_{t-1}) = ?$$

- Prediction based on past instances

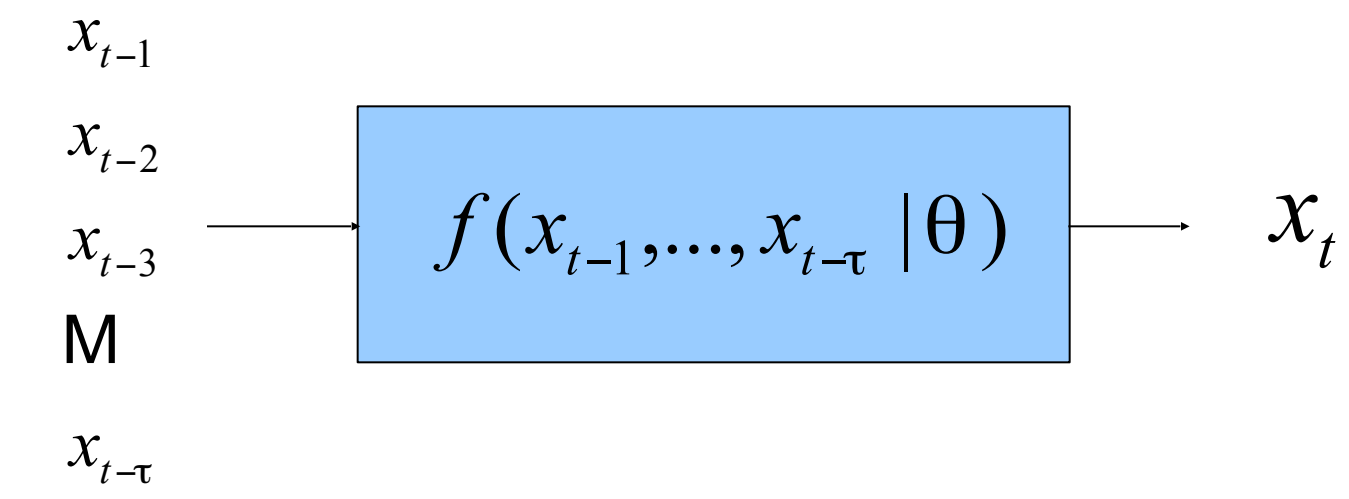
Function approximation

- $x_t = f(x_{t-1}, \ldots, x_{t-\tau} \mid \theta)$

  f : nonlinear function from $R^{\tau\text{-}1}$ to R

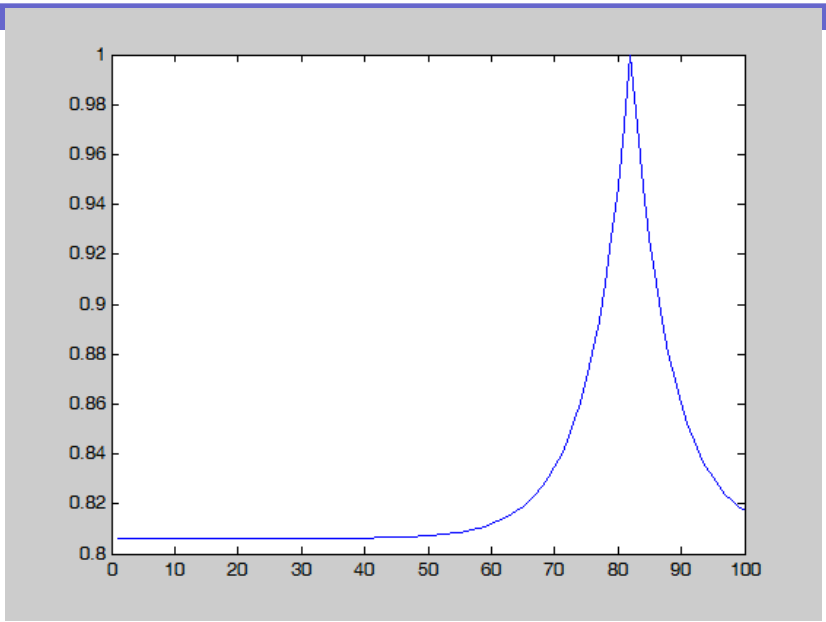
  θ : built - in parameters of f

Paired data for function approximation



$$\{(\mathbf{x}_t = [x_{t-1},\ldots,x_{t-\tau}]^T, y_t = x_t)\}_{t=\tau+1}^{100}$$

Hill model

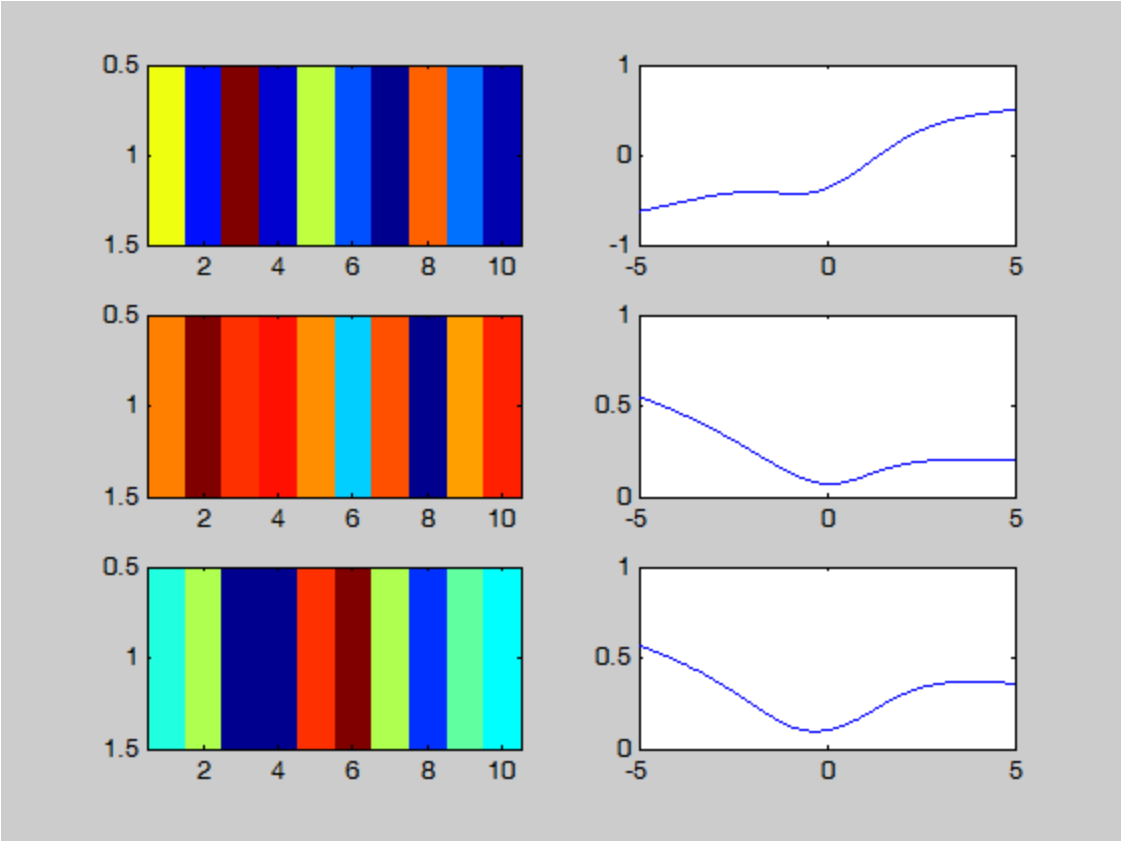


$$\{(\mathbf{x}_t = [x_{t-1}, \ldots, x_{t-\tau}]^T, y_t = x_t)\}_{t=\tau+1}^{100}$$

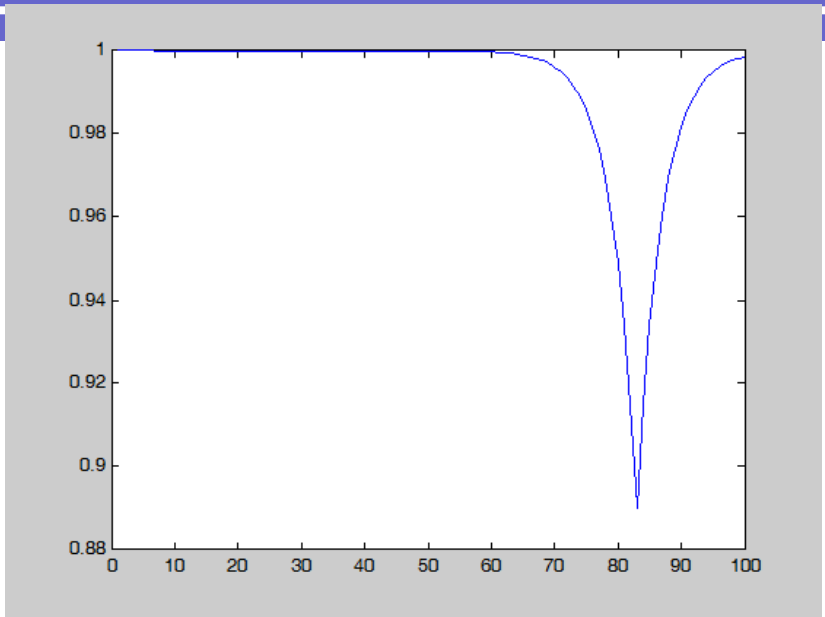
MLPotts learning

MLPotts 1

Approximating function



Valley model

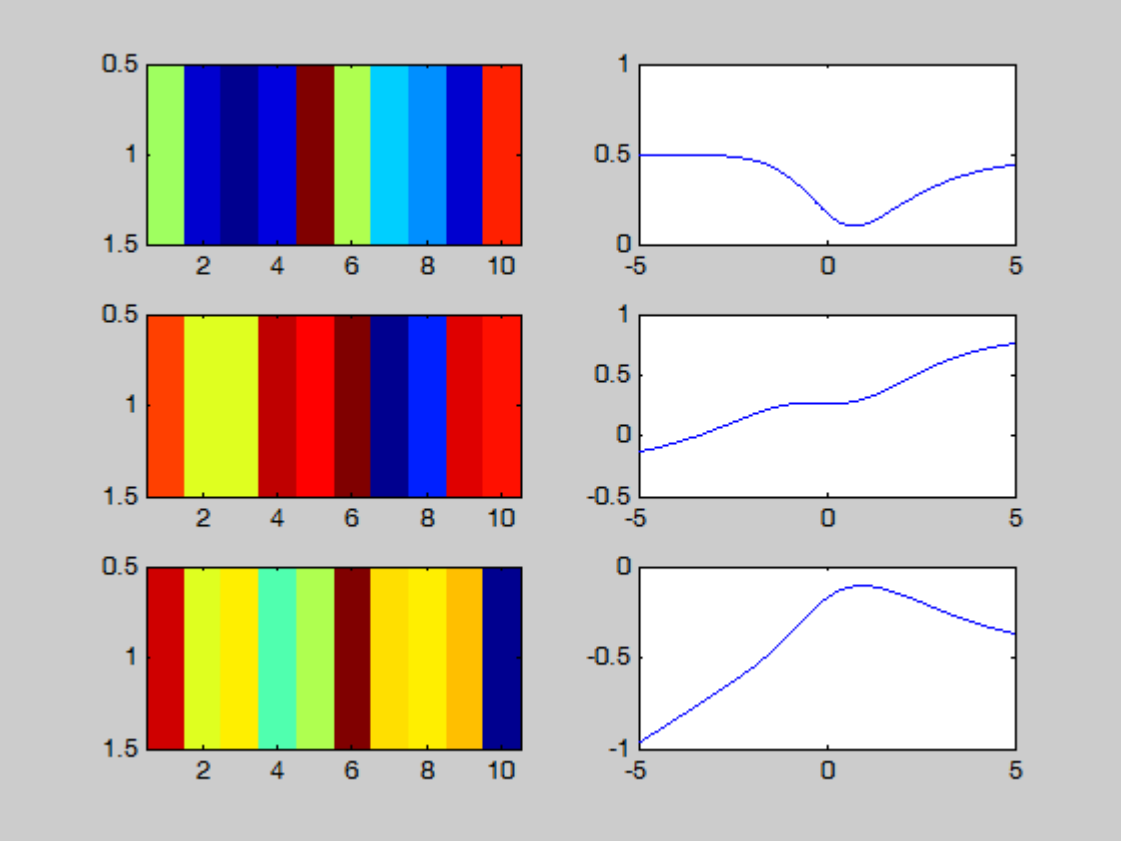


$$\{(\mathbf{x}_t = [x_{t-1}, \ldots, x_{t-\tau}]^T, y_t = x_t)\}_{t=\tau+1}^{100}$$

MLPotts learning

MLPotts 2

Approximating function



Learning Phase

- Use a hill pattern to derive an MLP network with parameters, a, b, r, collectively denoted by θ_h
- Use a valley pattern to derive an MLP network with parameters, a, b, r, collectively denoted by θ_v

Discriminating phase

- Given a pattern **x**
- Form paired data

$$\{(\mathbf{x}_t = [x_{t-1}, \ldots, x_{t-\tau}]^T, y_t = x_t)\}_{t=\tau+1}^{100}$$

- Mean square error

$$\mathrm{E}(\mathbf{x};\boldsymbol{\theta}) = \sum_t (y_t - F(\mathbf{x}_t;\boldsymbol{\theta}))^2$$

Discriminating phase

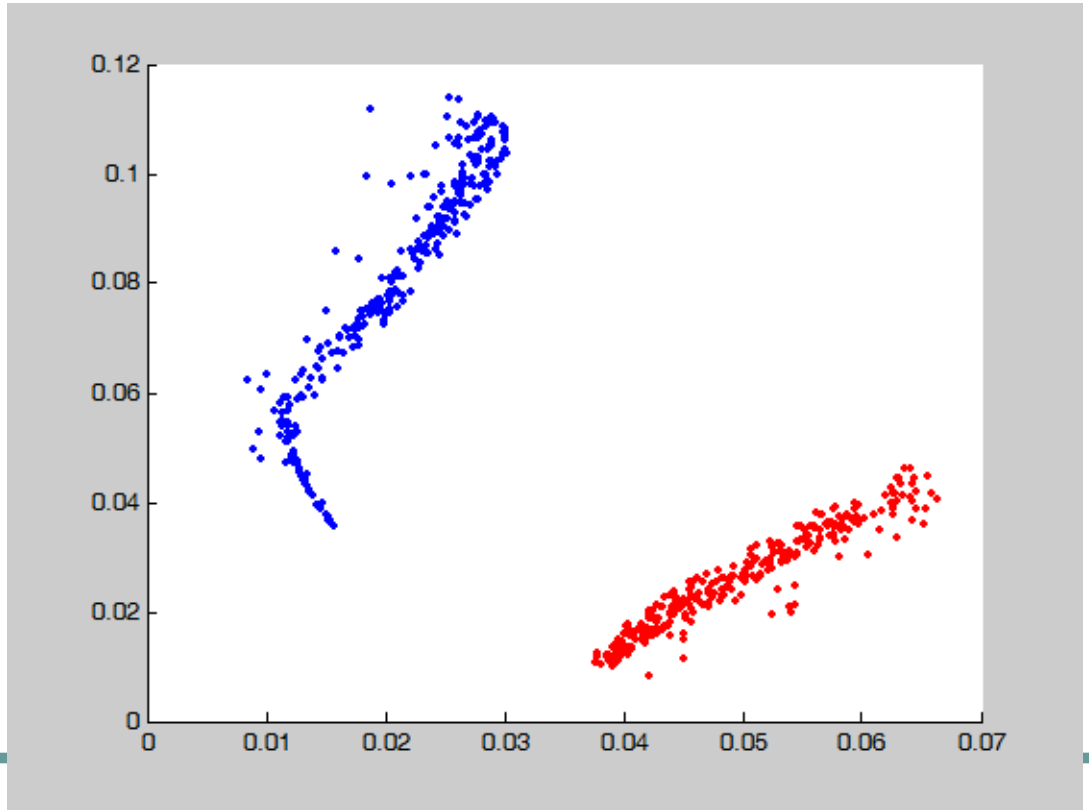
- **x** is classified as a hill if

$$\mathrm{E}(\mathbf{x};\boldsymbol{\theta}_h) \leq \mathrm{E}(\mathbf{x};\boldsymbol{\theta}_v)$$

a valley otherwise

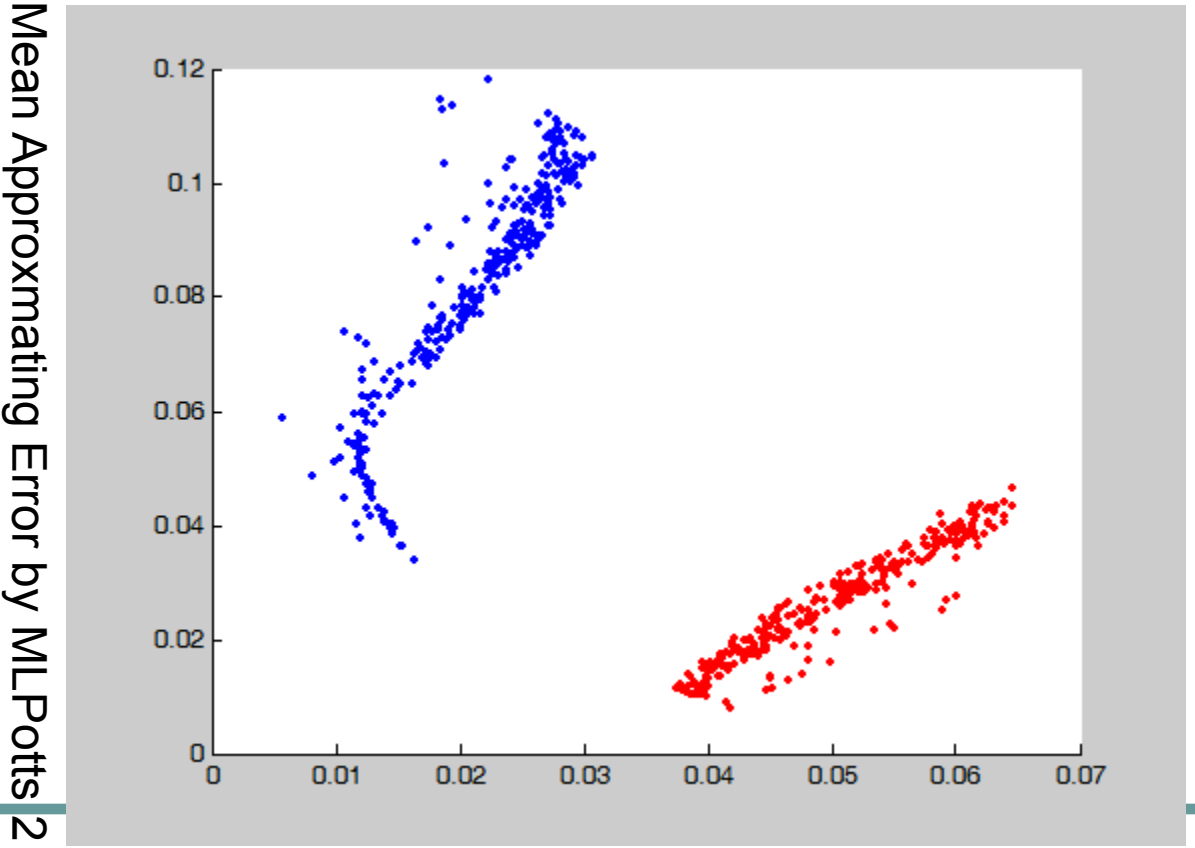
Linear or nonlinear Separable

- Without_noise_training



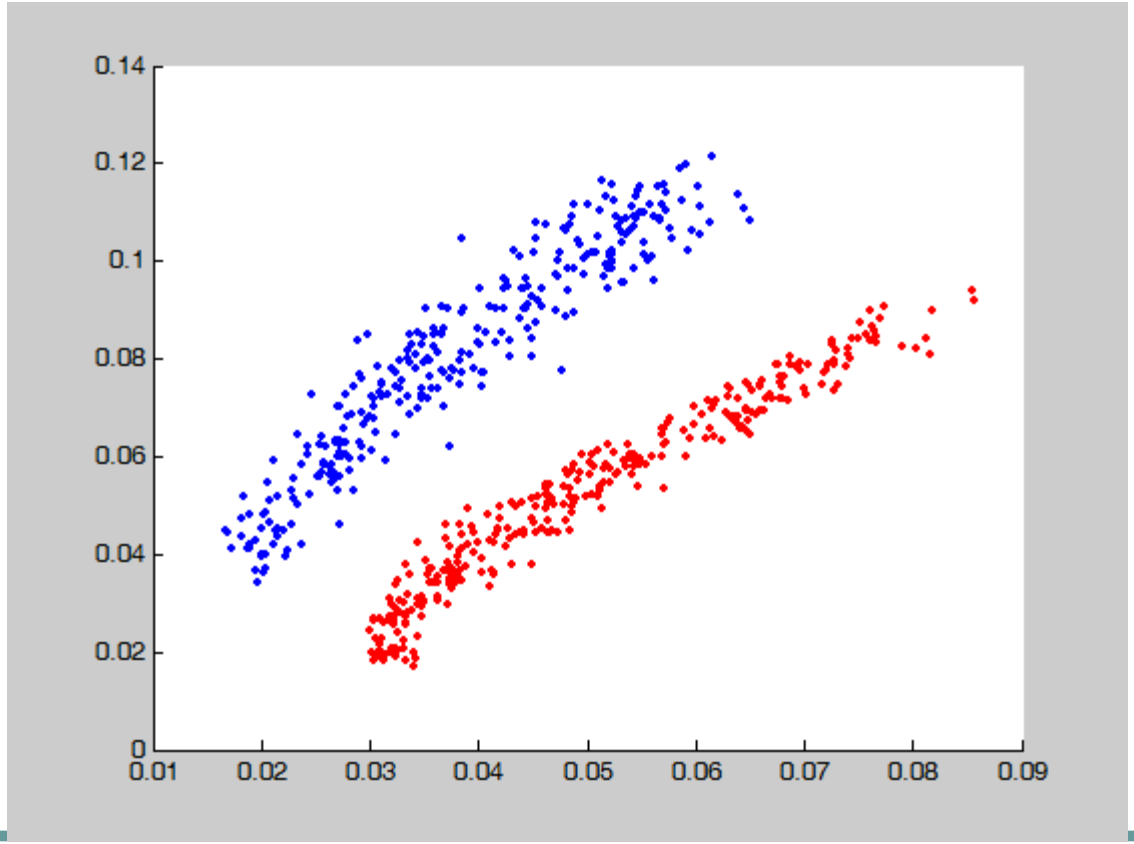
Linear or nonlinear separable

- Without_noise_testing



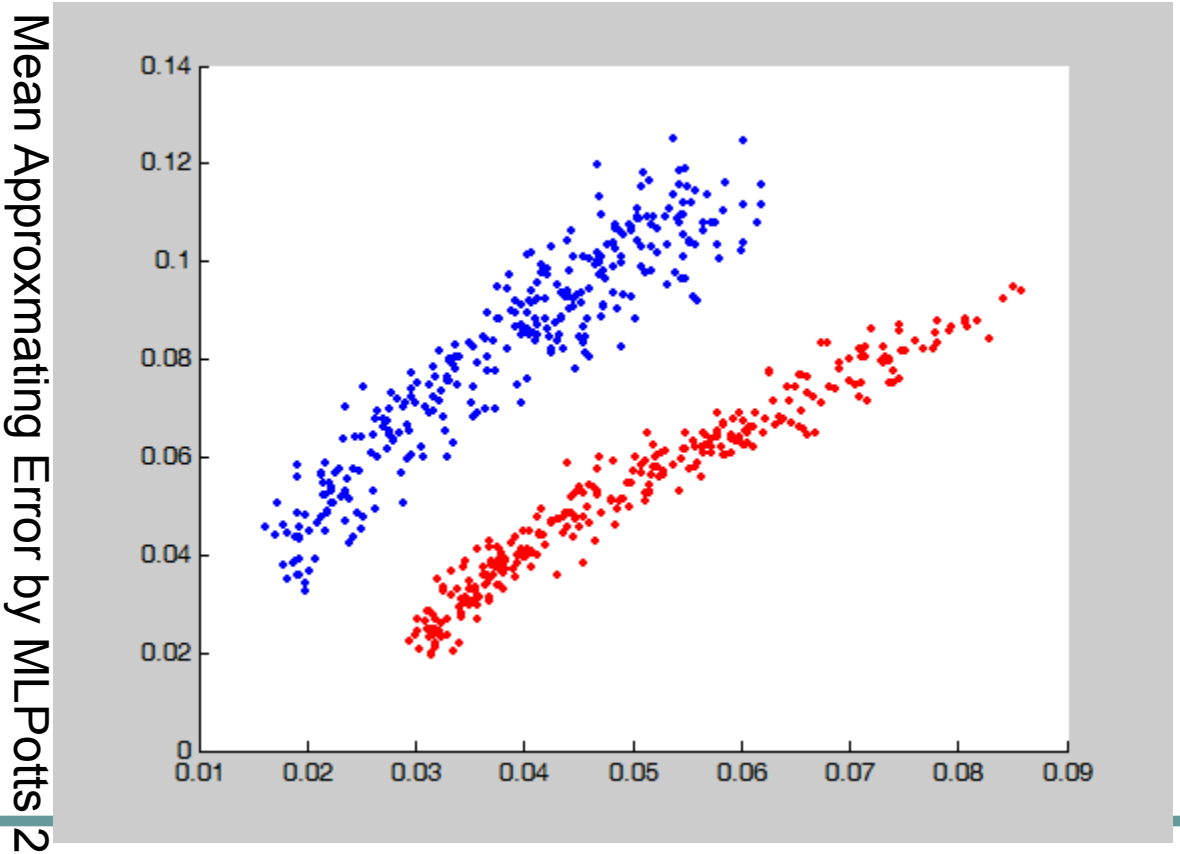
Linear or nonlinear separable

- With_noise_training



Linear or nonlinear separable

- With_noise_testing



Exercise

- Apply learning MLP networks by the Levenberg-Marquardt method to classification of hill-valley patterns

A. Employ MATLAB function read_data.m to input Hill_Valley_without_noise_Training.data
B. X denotes a matrix composed of 606 patterns.
   - Each pattern X(i,:) consists of 100 elements.
C. Y denotes a vector of binary elements
   - X(i,:) is a hill pattern if Y(i)==1
   - X(i,:) is a valley pattern otherwise

D. Let X(i,:) denote a hill pattern.

E. Set x=X(i,:). $\mathbf{x}=[x_1,\ldots,x_{100}]$

F. Form paired data

$$\{(\mathbf{z}_t, y_t)\}_{t=\tau+1}^{100}$$

$$\mathbf{z}_t = [x_{t-1},\ldots,x_{t-\tau}]^T$$

$$y_t = x_t$$

Training an MLP network

G. Paired data

$$S = \{(\mathbf{z}_t, y_t)\}_{t=\tau+1}^{100}$$

H. Apply MATLAB function findabr.m to approximate paired data in S

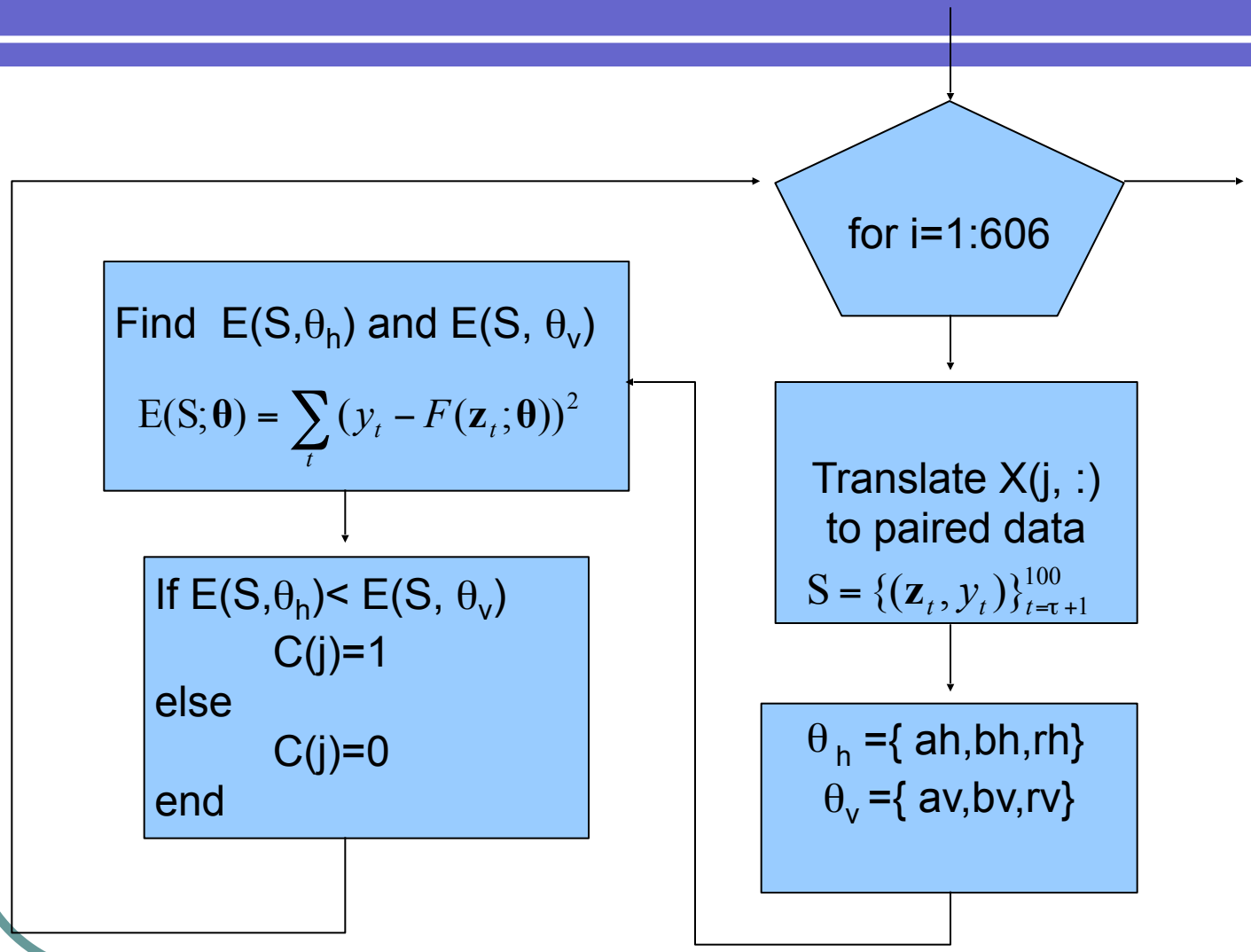
I. Store a,b,r to ah, bh, rh that denote a set of network parameters for hill modeling

Training the other MLP network

J. Let X(i, :) denote a valley pattern. Repeat steps D-H to create the other set of network parameters for valley modeling

K. Store a,b,r to av,bv,rv

Classification



- Apply MATLAB function x2y_MLP.m to translate a predictor to a target

$$y = F(\mathbf{x};\boldsymbol{\theta})$$