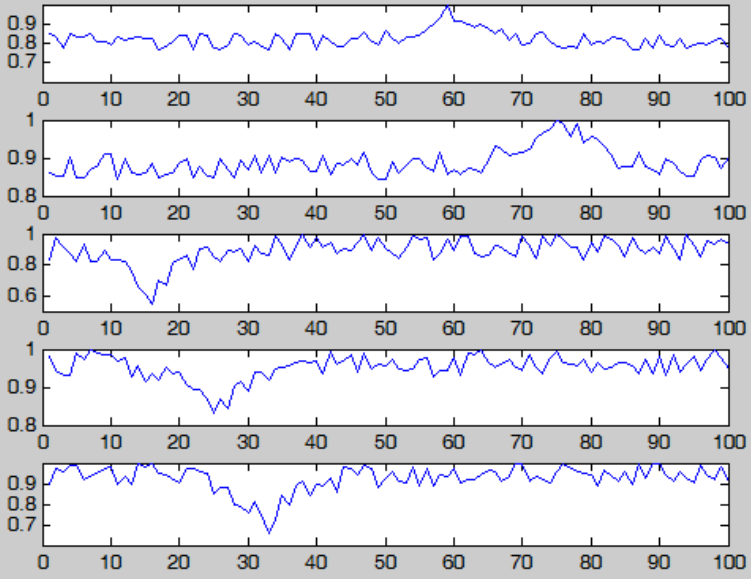
Time series classification



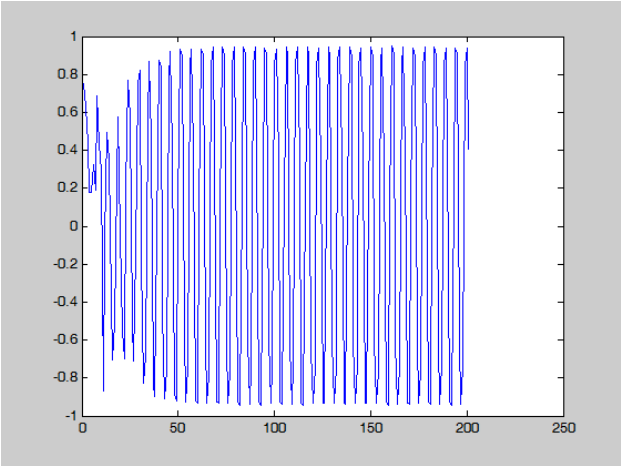
Time series classification

- Nonlinear recursive relations
- Time series prediction
- Hill-Valley classification
- Lorenz series

Nonlinear recursion

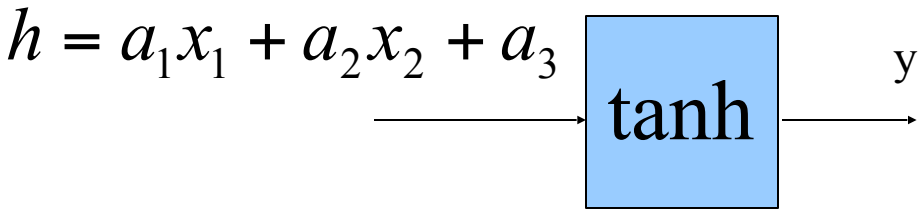
- Post-nonlinear combination of predecessors
- $z[t]=\tanh(a_1 z[t-1]+ a_2 z[t-2]+\ldots+ a_\tau z[t-\tau])+ e[t],$

  $t= \tau,\ldots,N$



Post-nonlinear Projection

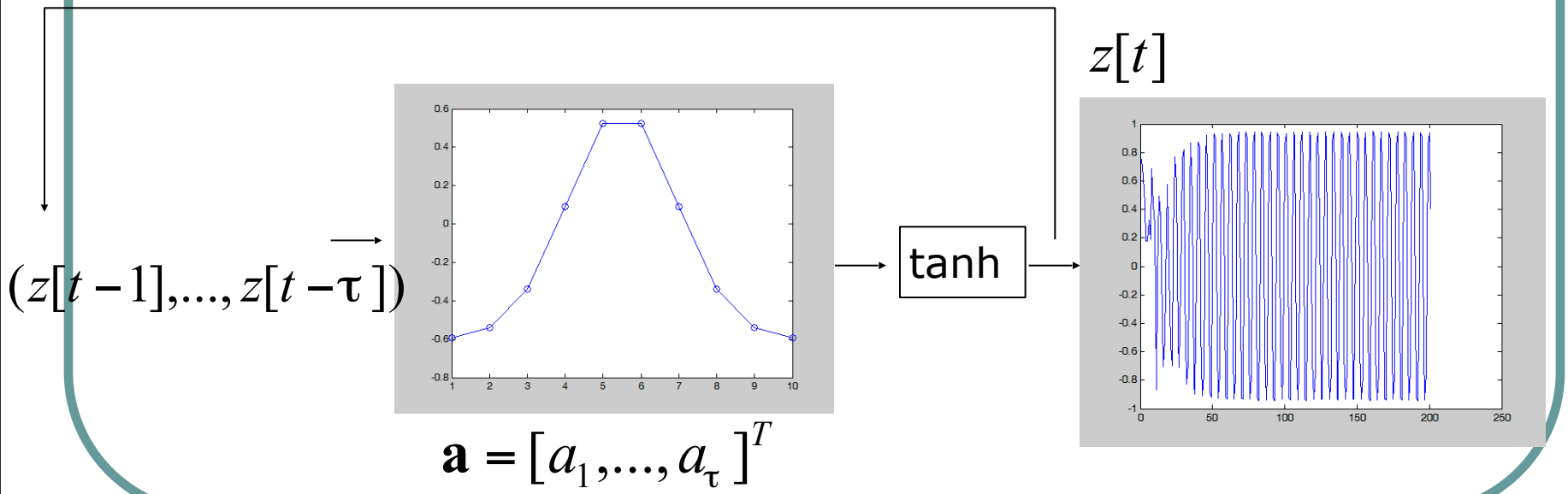
$$y = \tanh(h = a_1 x_1 + a_2 x_2 + a_3)$$



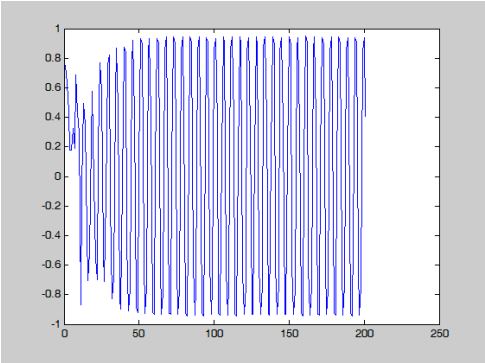
Data generation

- $z[t]=\tanh(a_1 z[t-1]+ a_2 z[t-2]+\ldots+ a_\tau z[t-\tau])+ e[t]$,

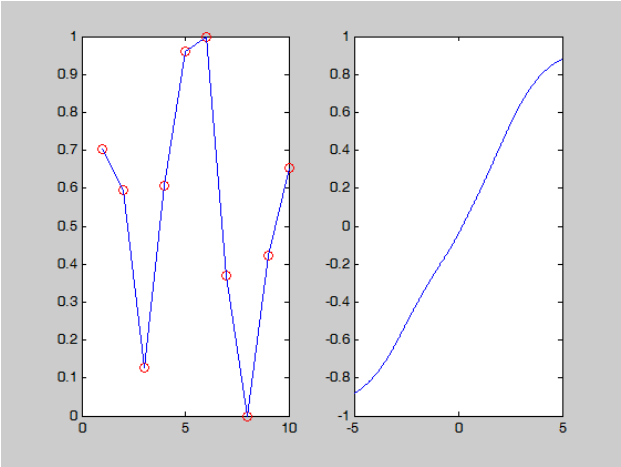
  $t= \tau,\ldots,N$



Nonlinear Recursion

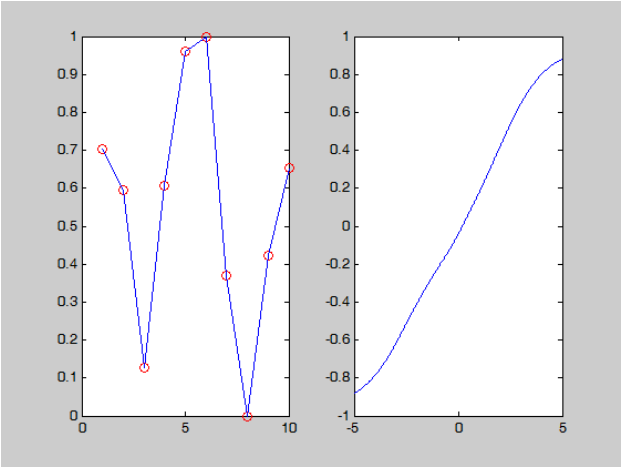


LM learning

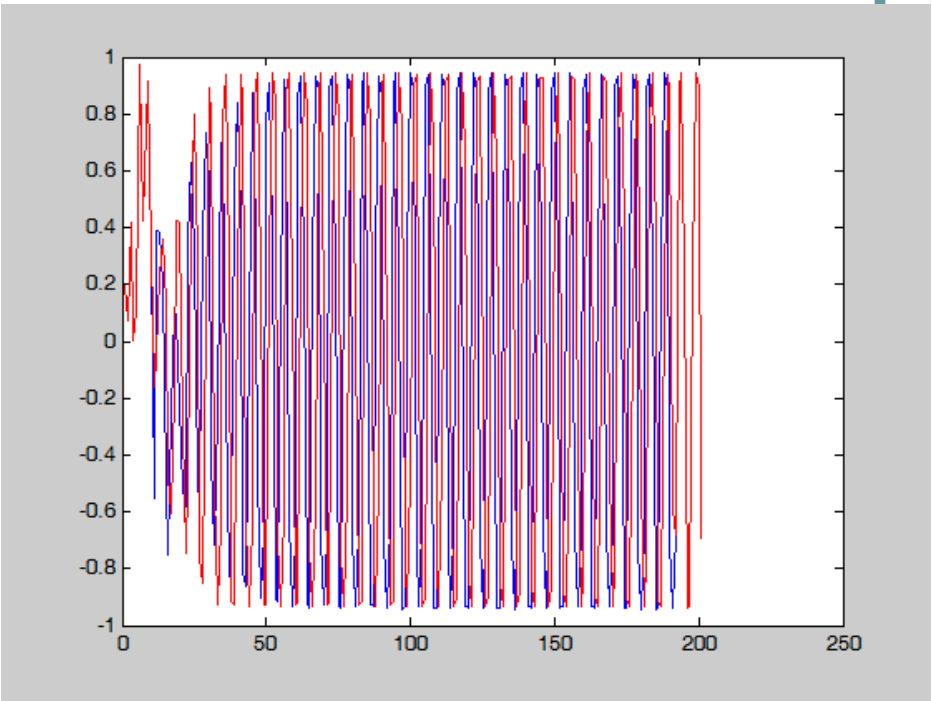


Prediction

- Use initial τ-1 instances to generate the full time series (red) based on estimated post-nonlinear filter



Post-nonlinear filter

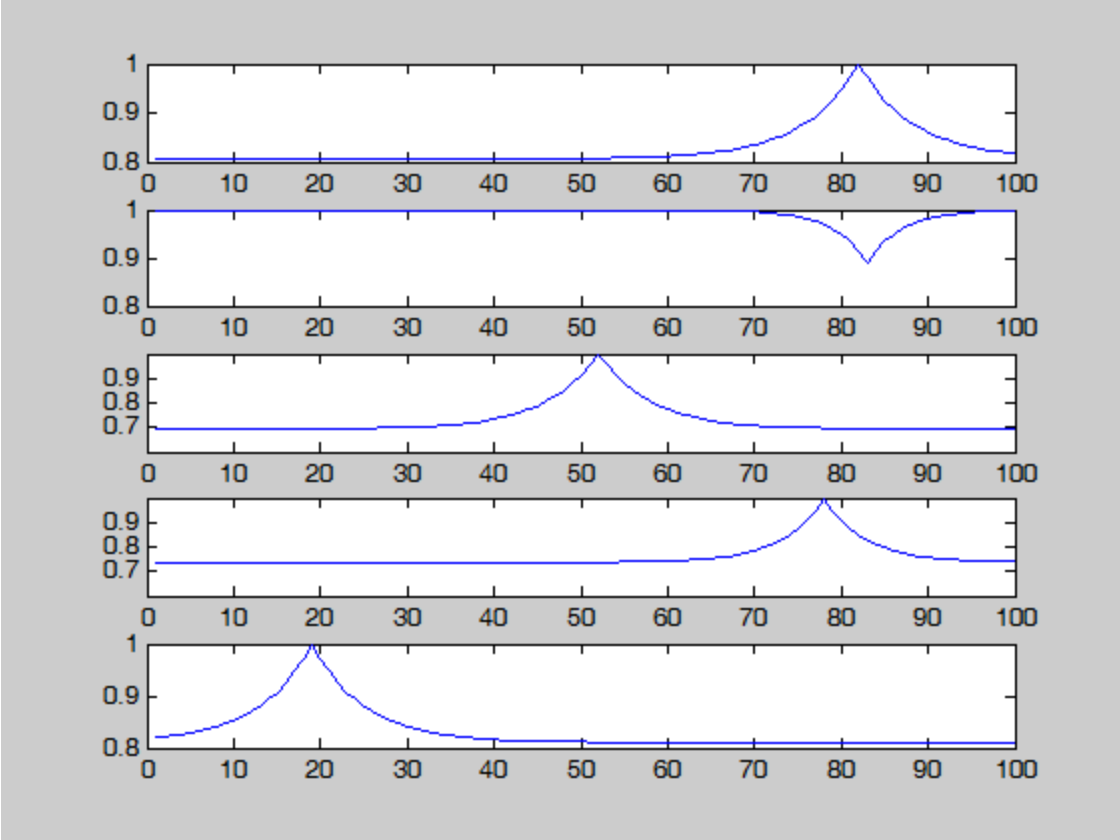


Nonlinear recursions: hill-valley

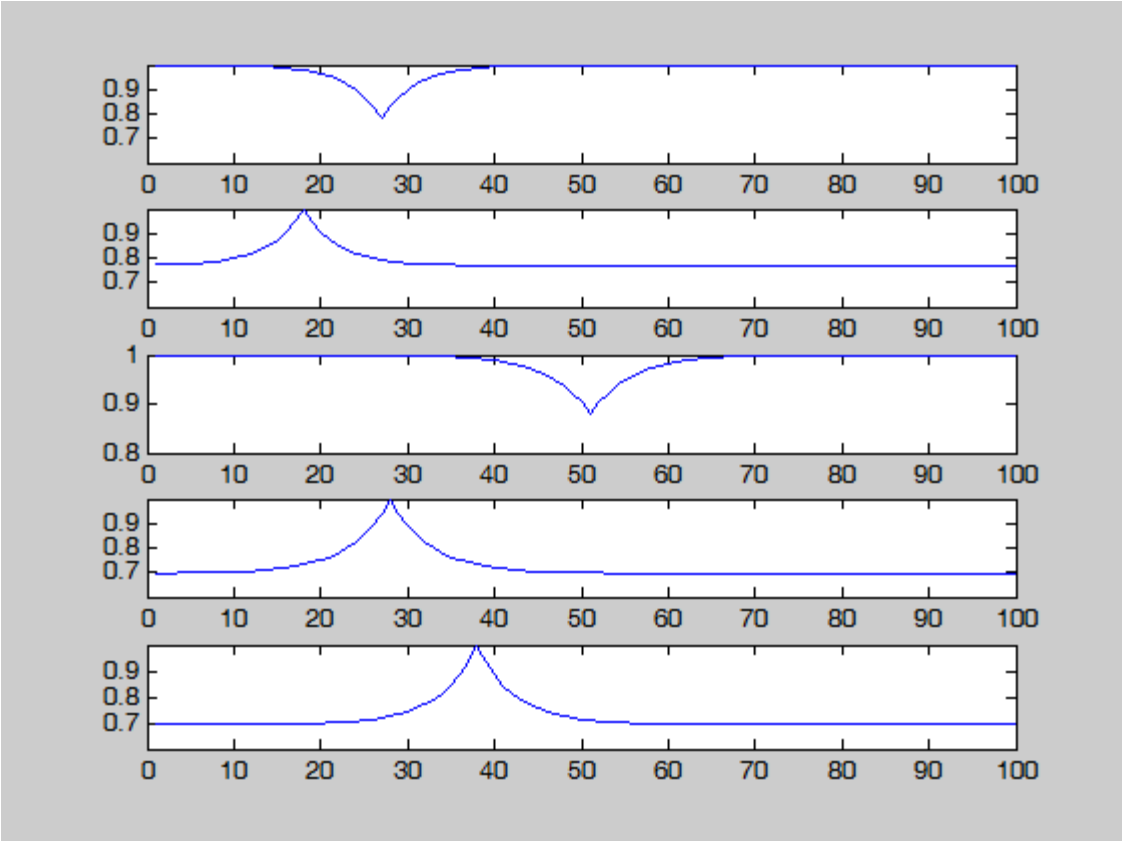
- Classify hill and valley

UCI Machine Learning Repository

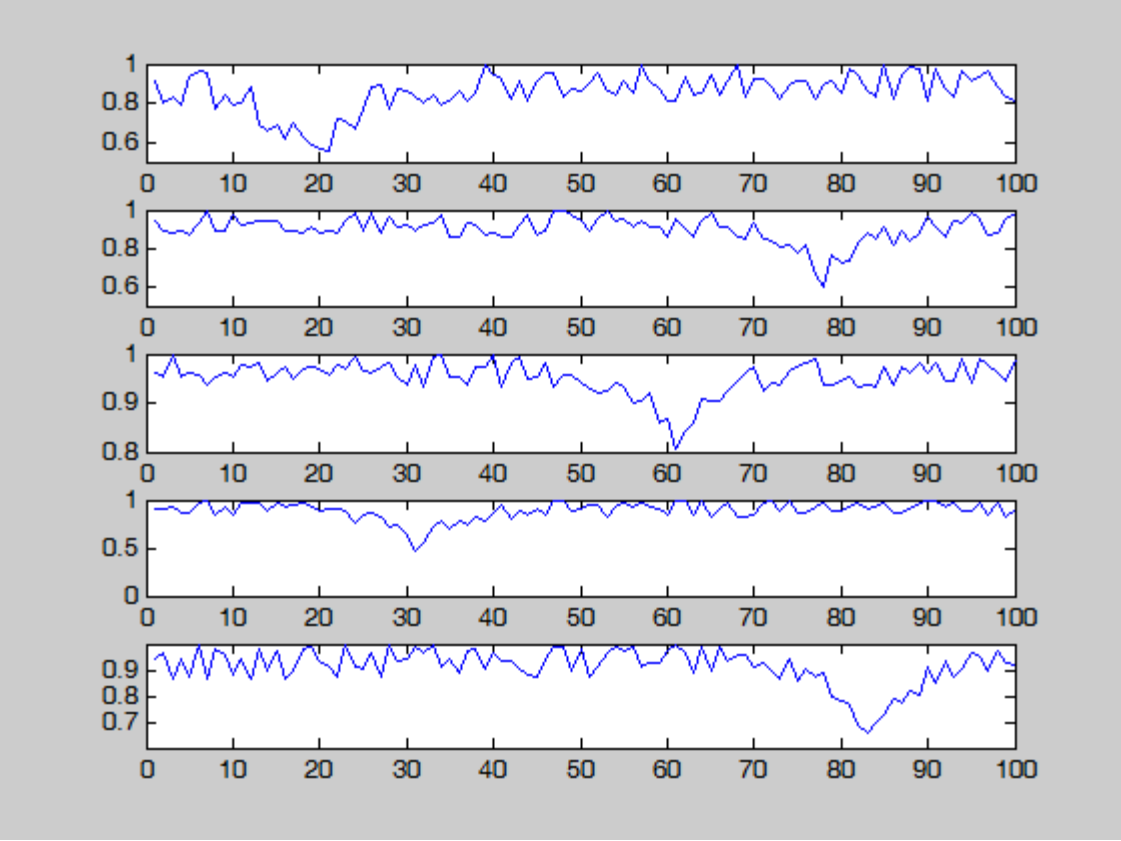
Noise-free Hill Valley



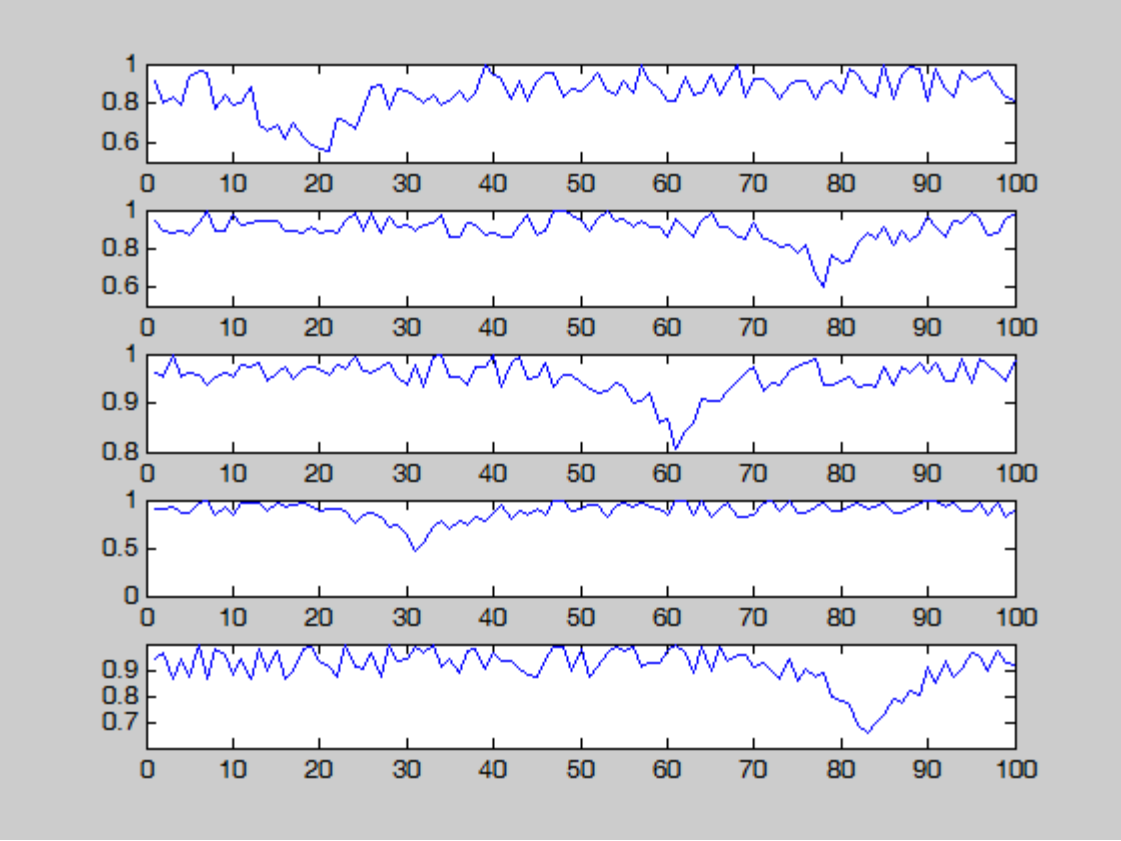
Noise-free Hill Valley



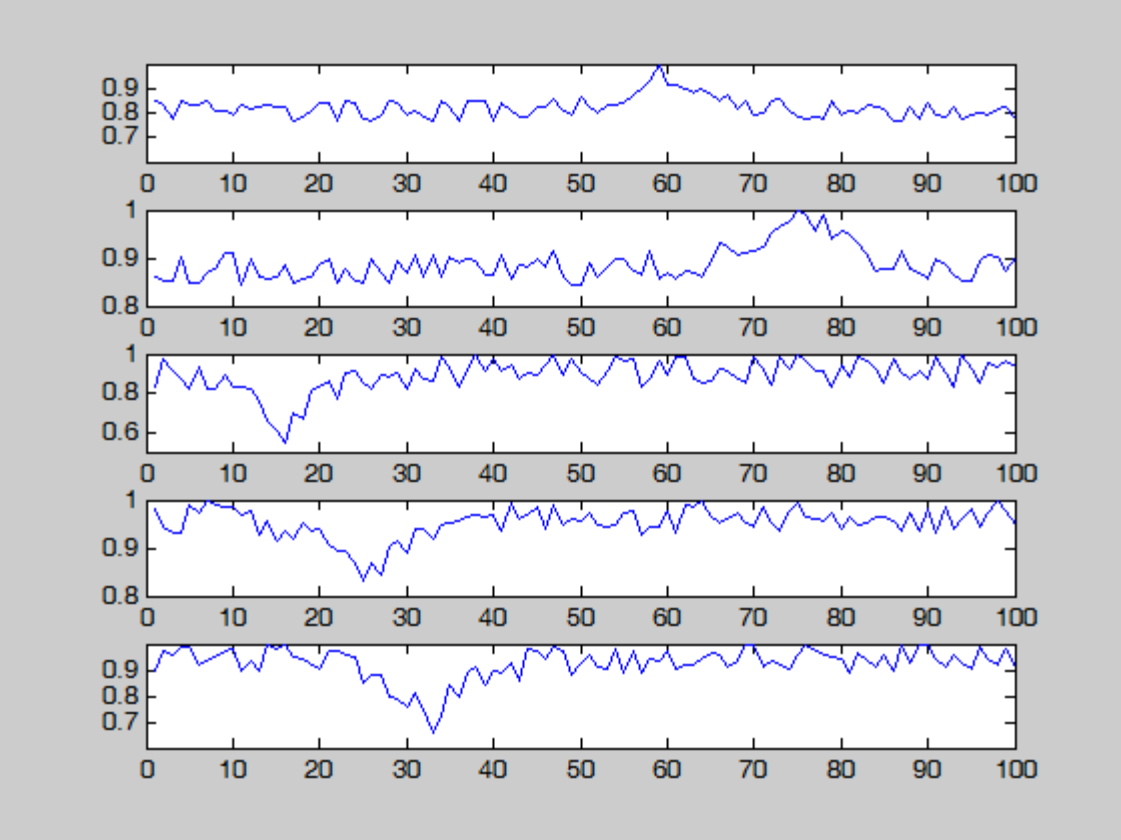
Hill-Valley with noise



Hill-Valley with noise



Hill-Valley with noise

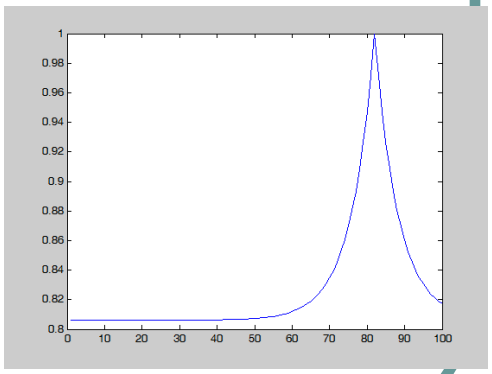


Nonlinear recursion

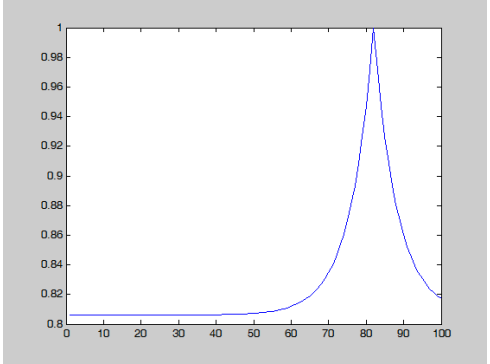
- LM_RBF: $z[t]=F(z[t-1], z[t-2], \ldots, z[t-\tau])+ e[t]$,

  $t= \tau,\ldots,N$

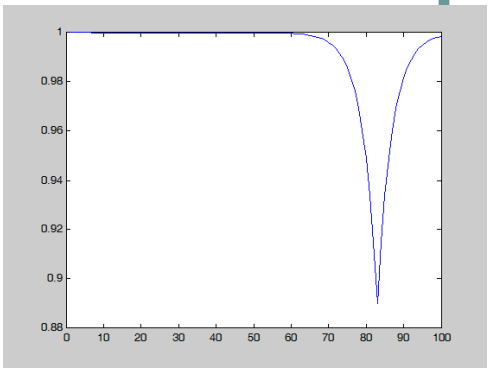
$$(z[t-1],\ldots,z[t-\tau]) \rightarrow \boxed{\text{LM\_RBF 1}} \xrightarrow{z[t]}$$



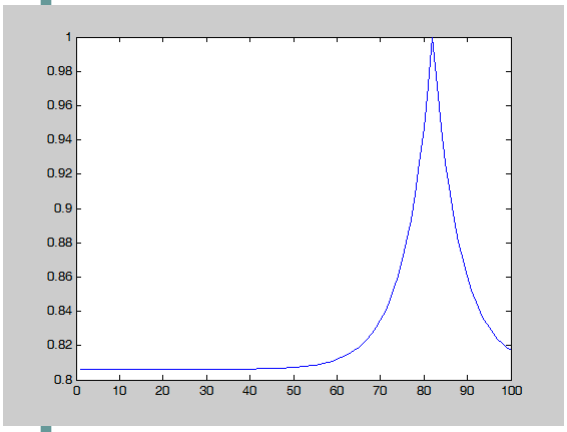
$(z[t-1],\ldots,z[t-\tau]) \rightarrow$ LM_RBF 1 $\xrightarrow{\quad} z[t]$



$(z[t-1],\ldots,z[t-\tau]) \rightarrow$ LM_RBF 2 $\xrightarrow{\quad} z[t]$

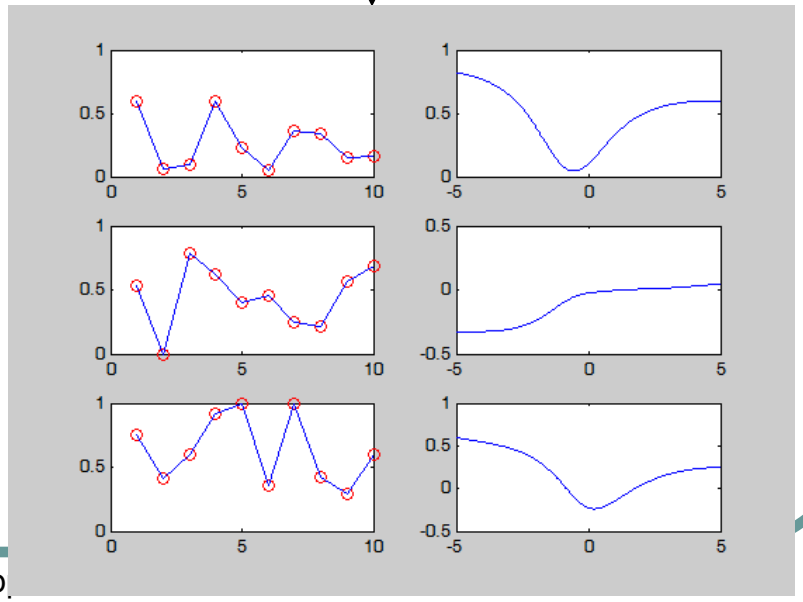


Nonlinear Recursion

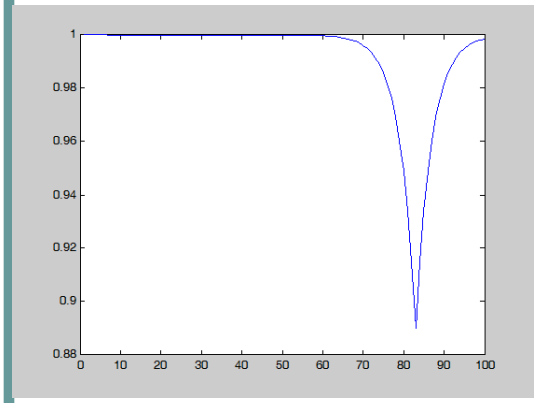


LM learning

MLPotts 1 (H)

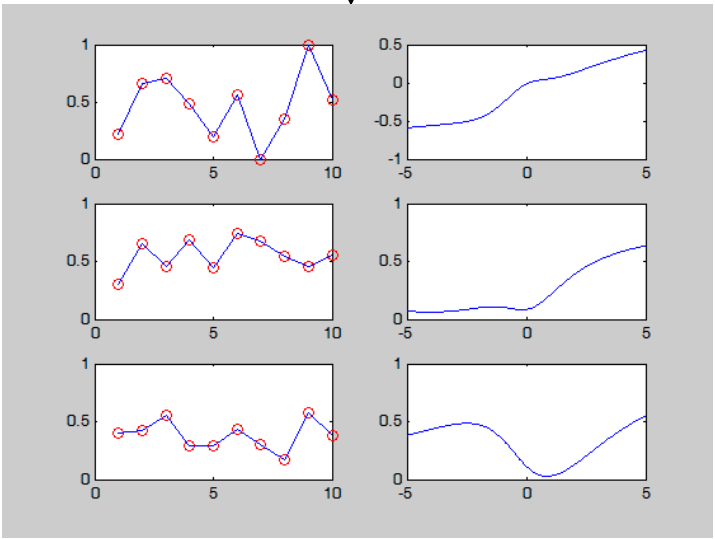


Nonlinear Recursion

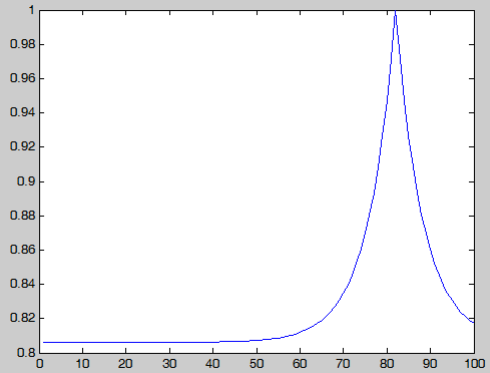


LM learning

MLPotts 2 (V)

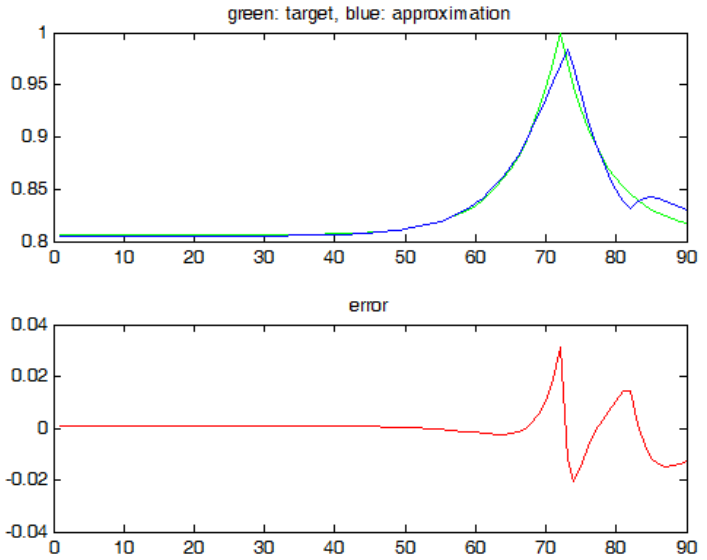


Approximating error

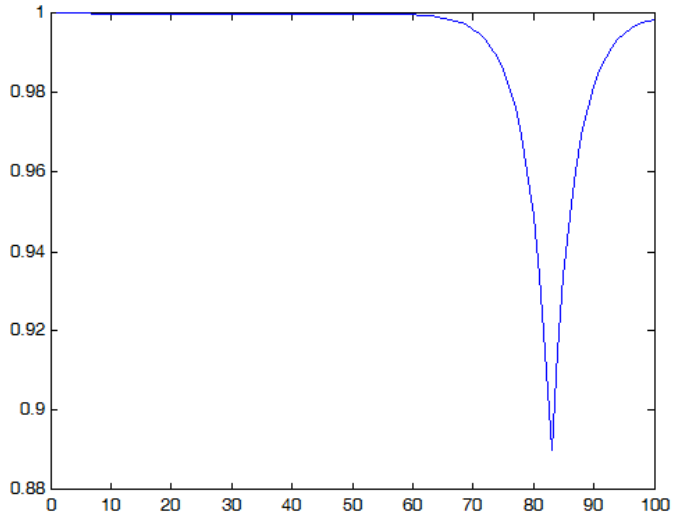


Green: Original Hill
Blue: Approximated Hill
Red: Approximating error

LM_RBF 1 (H)



Approximating error

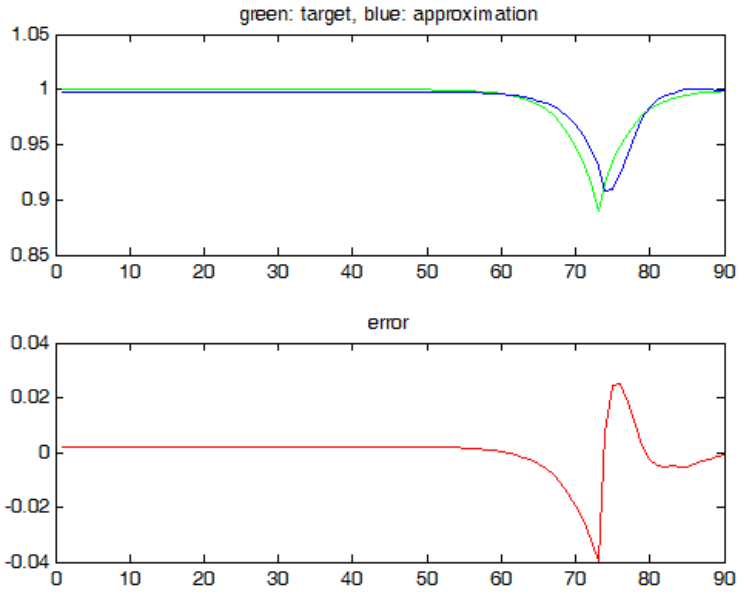


LM_RBF 2 (V)

Green: Original Hill

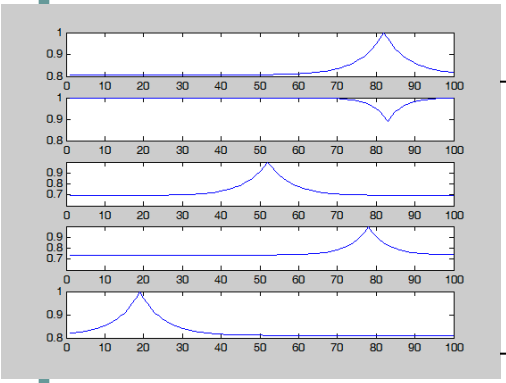
Blue: Approximated Hill

Red: Approximating erro



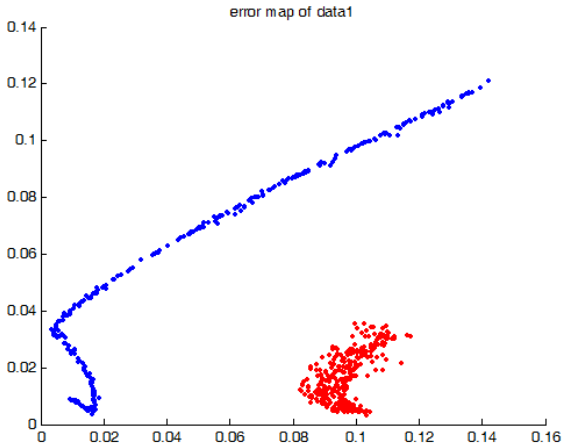
Separable 2D diagram of Hill-Valley

Noise-free data set 1



→ LM_RBF 1 (H) →

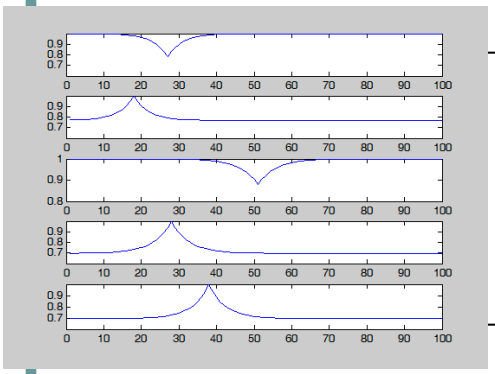
→ LM_RBF 2 (V) →



Mean Approximating Errors o
Two Models

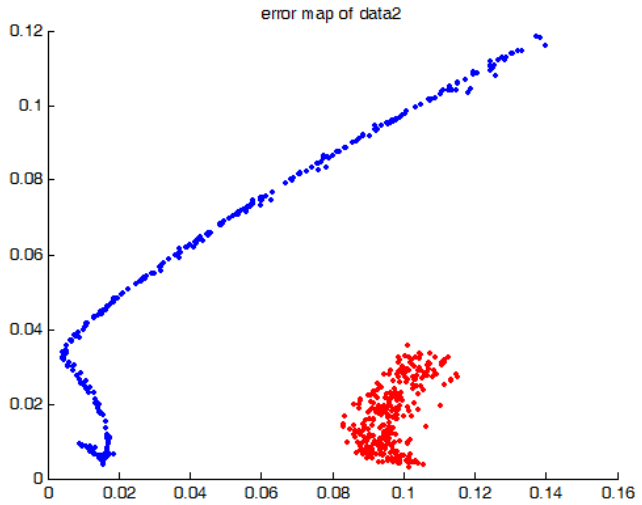
Separable 2D diagram of Hill-Valley

Noise-free data set 2



→ LM_RBF 1 (H) →

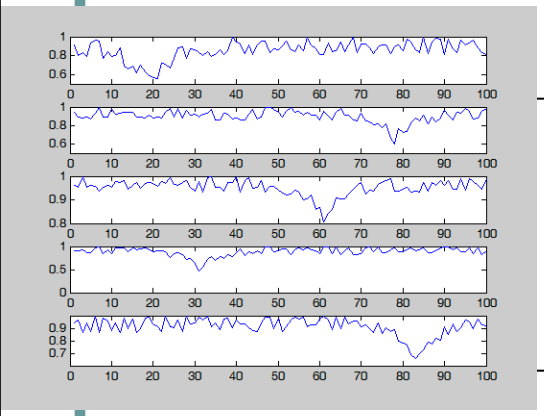
→ LM_RBF 2 (V) →



Mean Approximating Errors o
Two Models

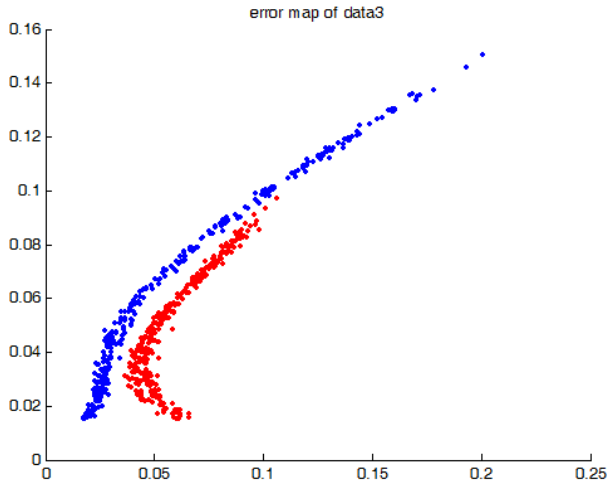
Separable 2D diagram of Hill-Valley

Dataset 1 (Noise)



→ LM_RBF 1 (H) →

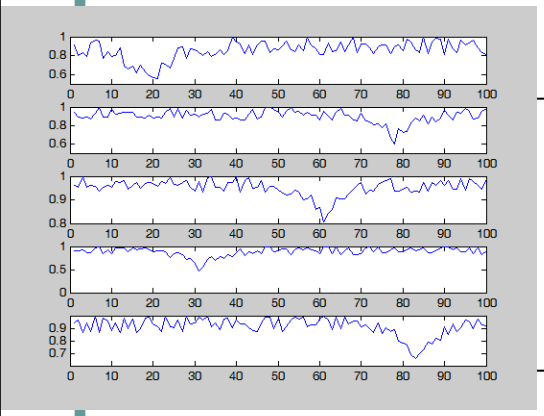
→ LM_RBF 2 (V) →



Mean Approximating Errors o
Two Models

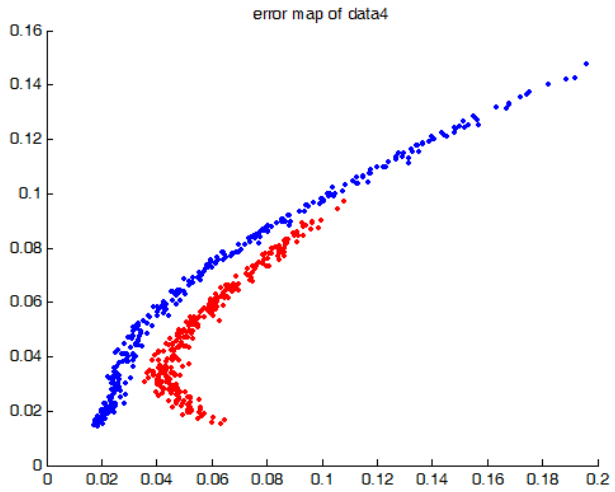
Separable 2D diagram of Hill-Valley

Dataset 2 (Noise)



→ LM_RBF 1 (H) →

→ LM_RBF 2 (V) →



Mean Approximating Errors o
Two Models

Chaos Time Series

- Eric's Home Page

Lorenz Attractor

The equations that govern the Lorenz attractor are:

$$\frac{dx}{dt} = \sigma(y - x)$$

$$\frac{dy}{dt} = x(\rho - z) - y$$

$$\frac{dz}{dt} = xy - \beta z$$

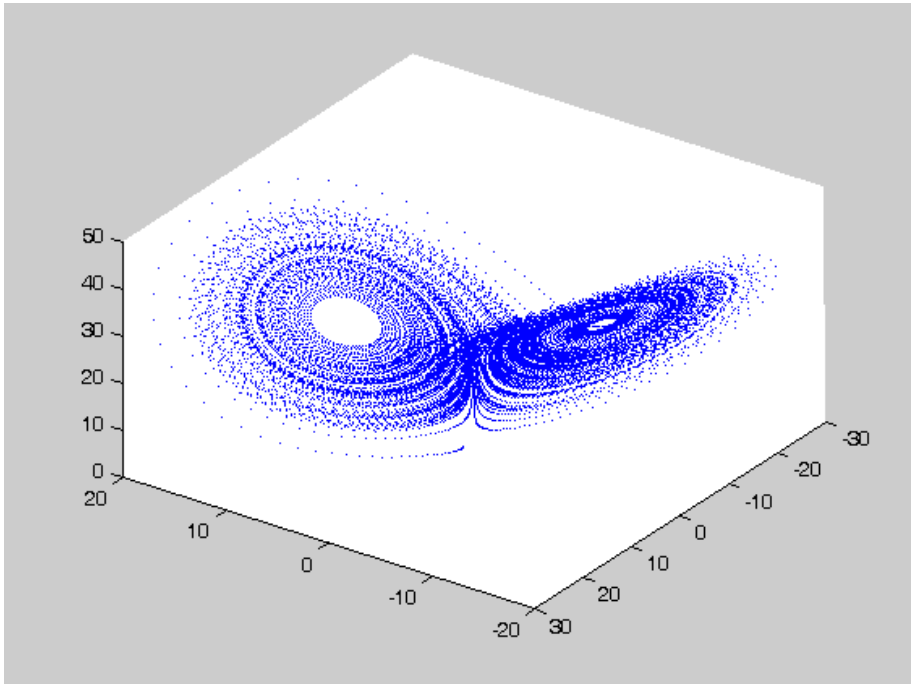
Lorenz

```
>> NN=20000;
>> [x,y,z] = lorenz(NN);
>> plot3(x,y,z,'.');
```

$$lx : \sigma = 10$$

$$ly : \rho = 28$$

$$lz : \beta = \frac{8}{3}$$

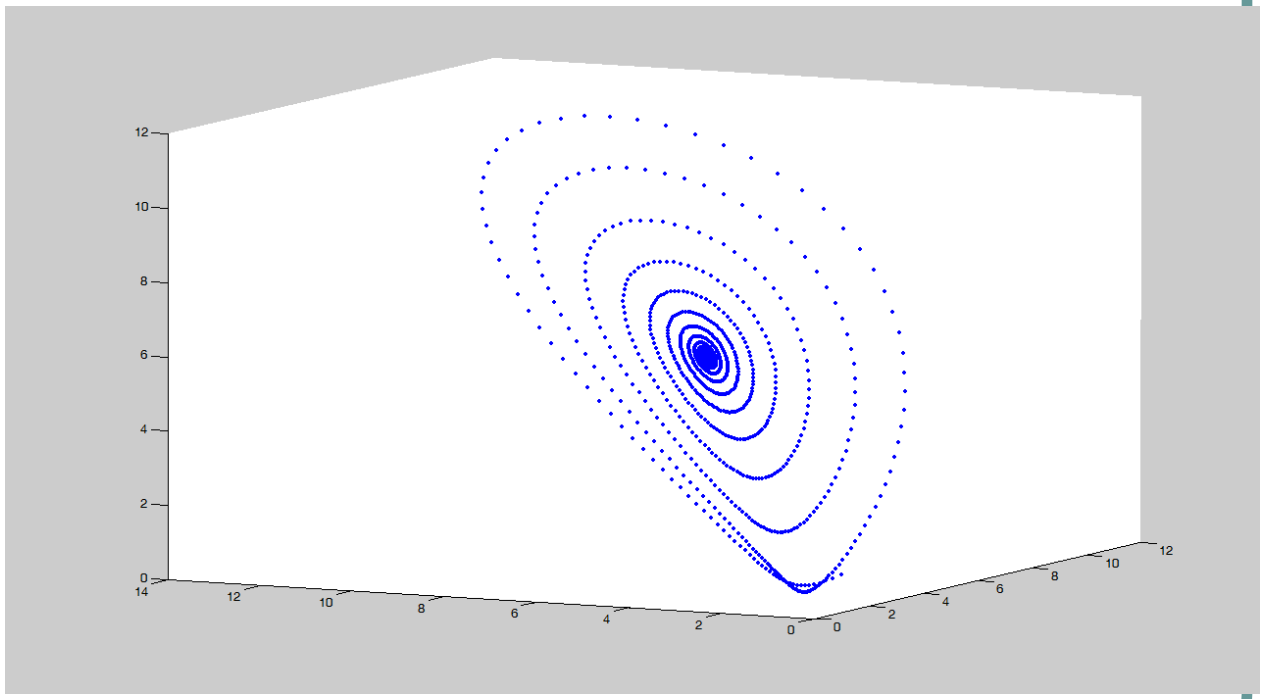


$\rho = 13$

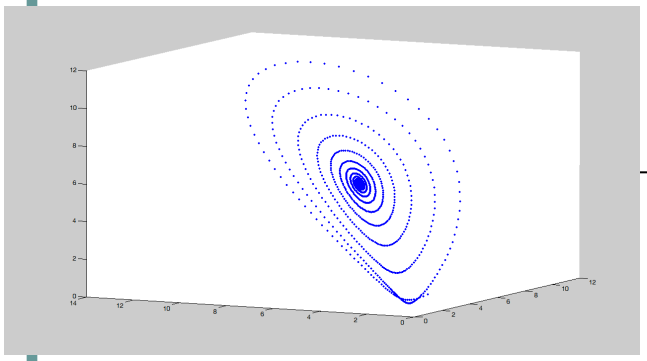
$$lx : \sigma = 10$$

$$ly : \rho = 13$$

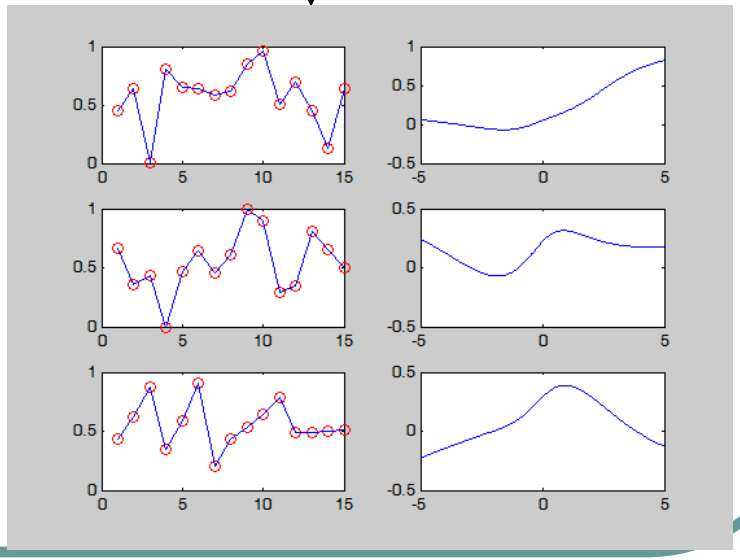
$$lz : \beta = \frac{8}{3}$$



Nonlinear Recursion

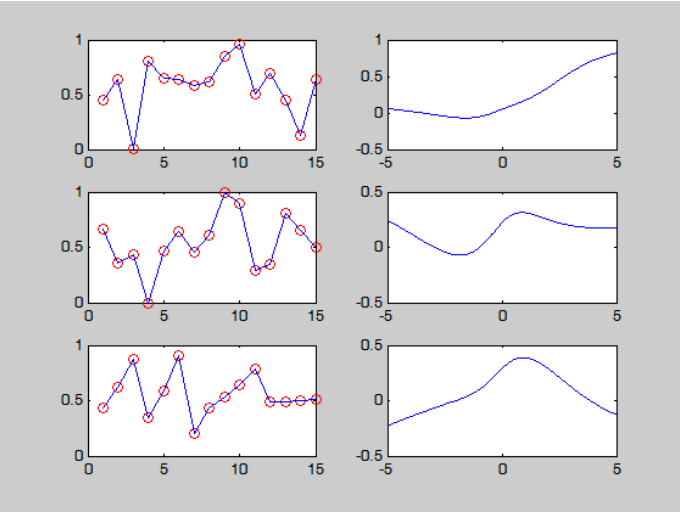


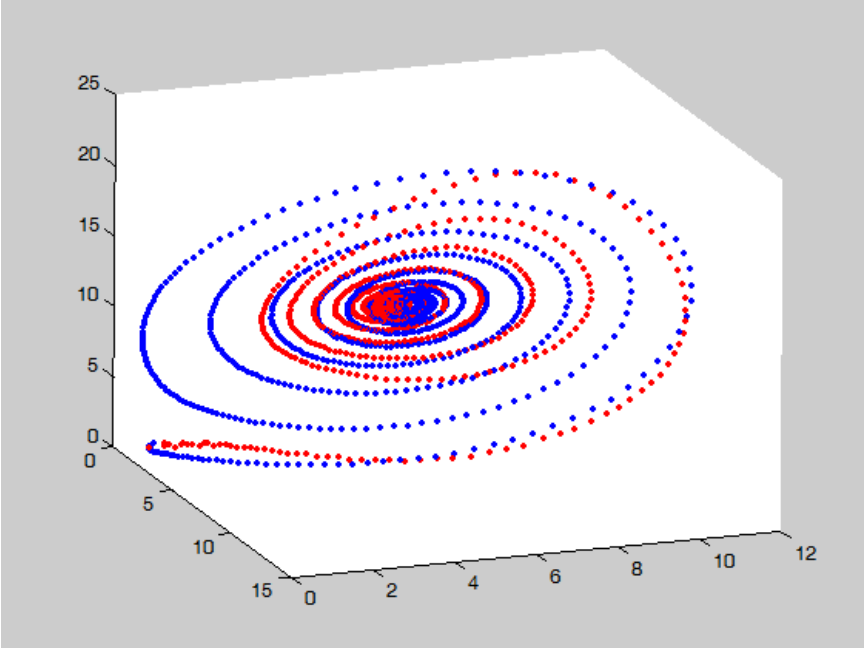
Neural nonlinear recursive relations



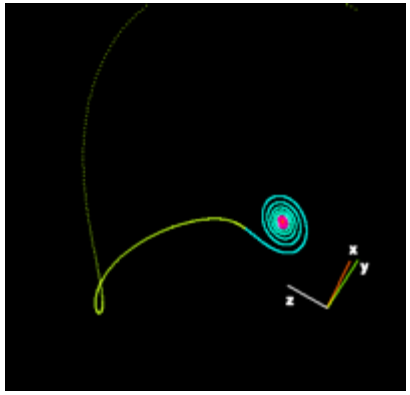
Prediction

- Use initial τ-1 instances to generate the full time series (red) based on derived neural recursions

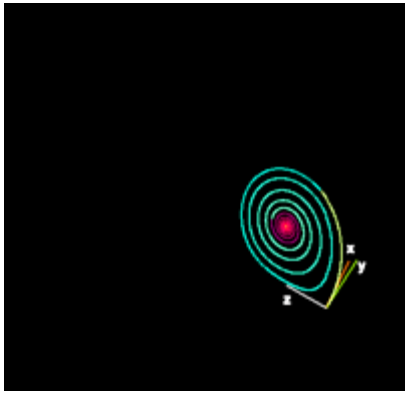




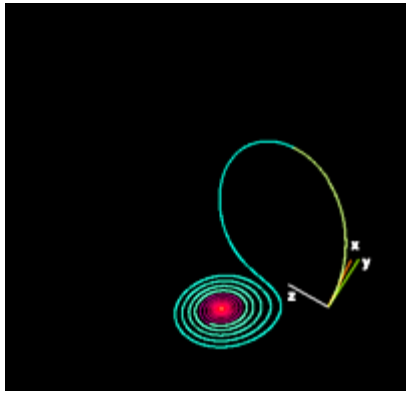
Rayleigh number



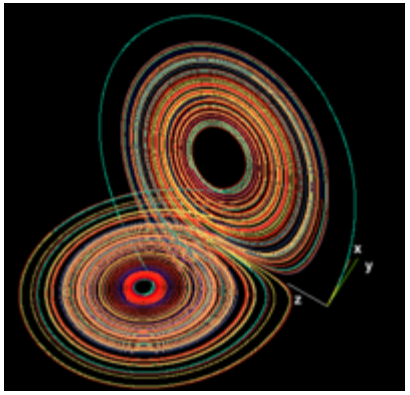
ρ=14, σ=10, β=8/3



ρ=13, σ=10, β=8/3



ρ=15, σ=10, β=8/3

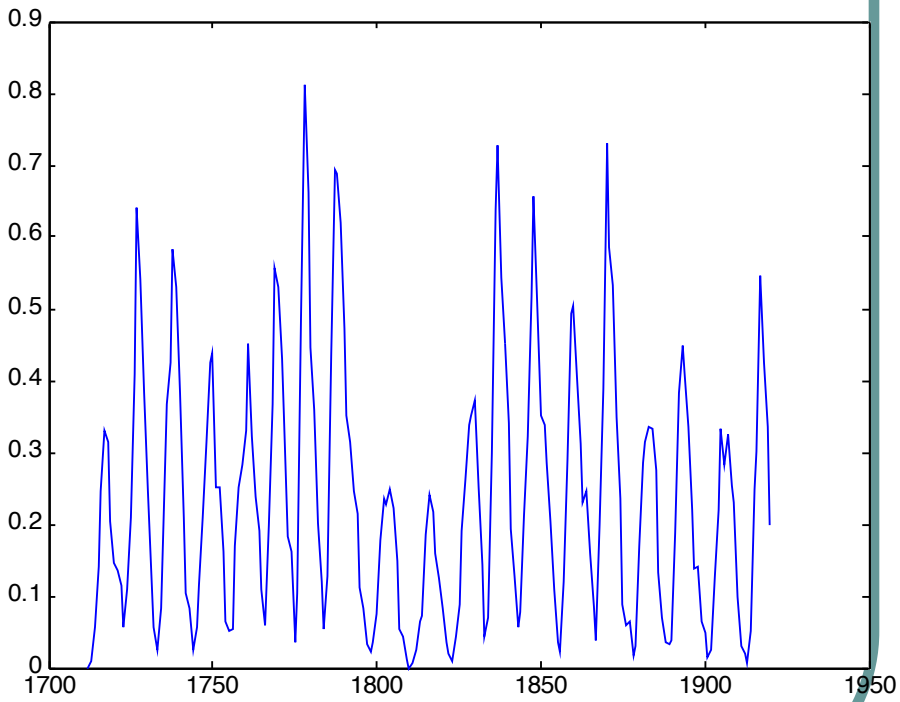


ρ=28, σ=10, β=8/3

Sunspot time series

sunpot data

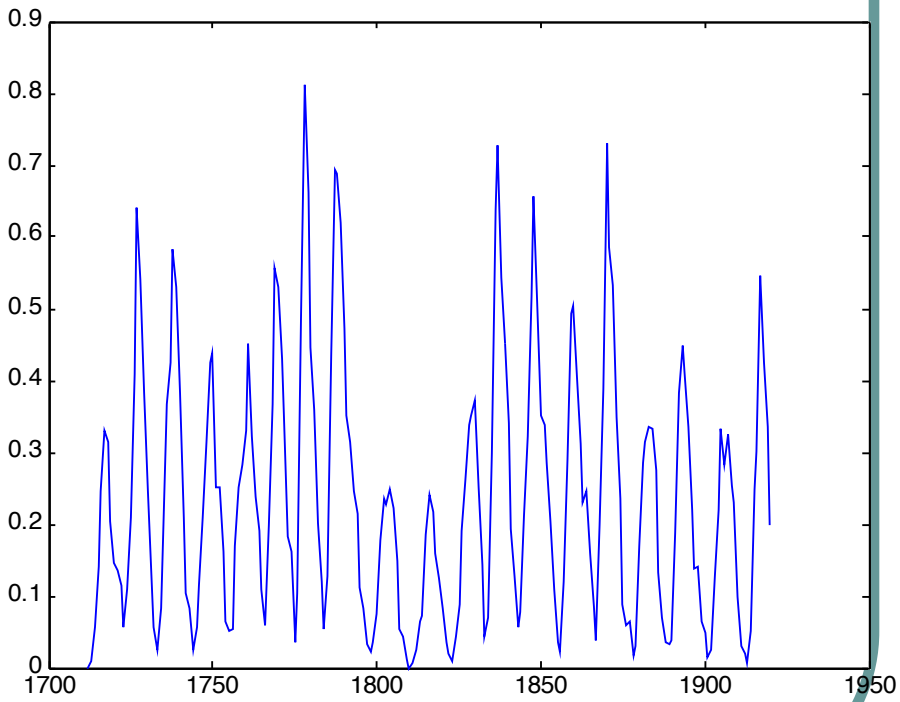
```
load sunspot_train.mat;
n=length(Y);
plot(1712:1712+n-1,Y);
```



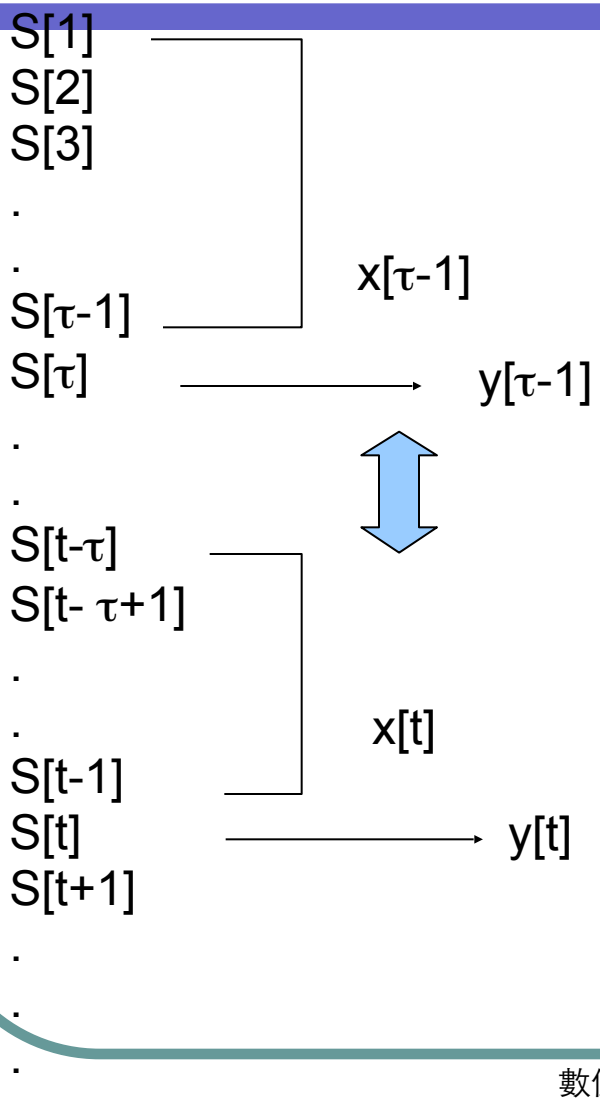
Sunspot time series

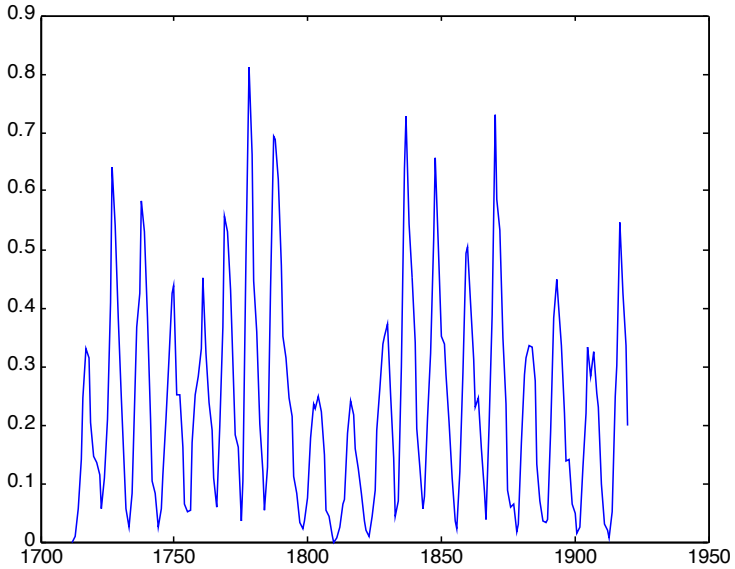
sunpot data (testing)

```
load sunspot_test_1.mat;
n=length(Y);
plot(1952:1952+n-1,Y);
```



Time-series 2 paired data

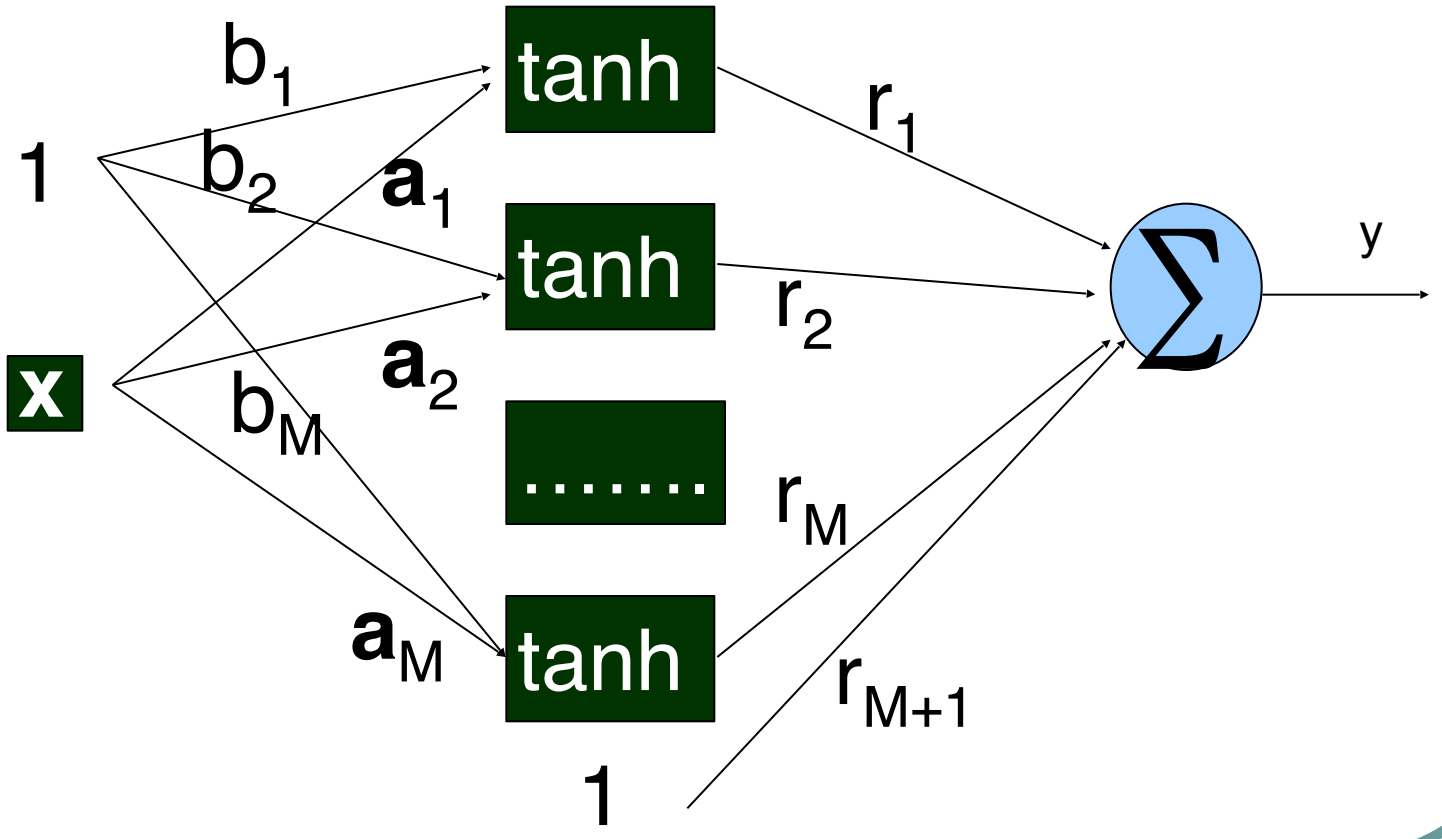




Network

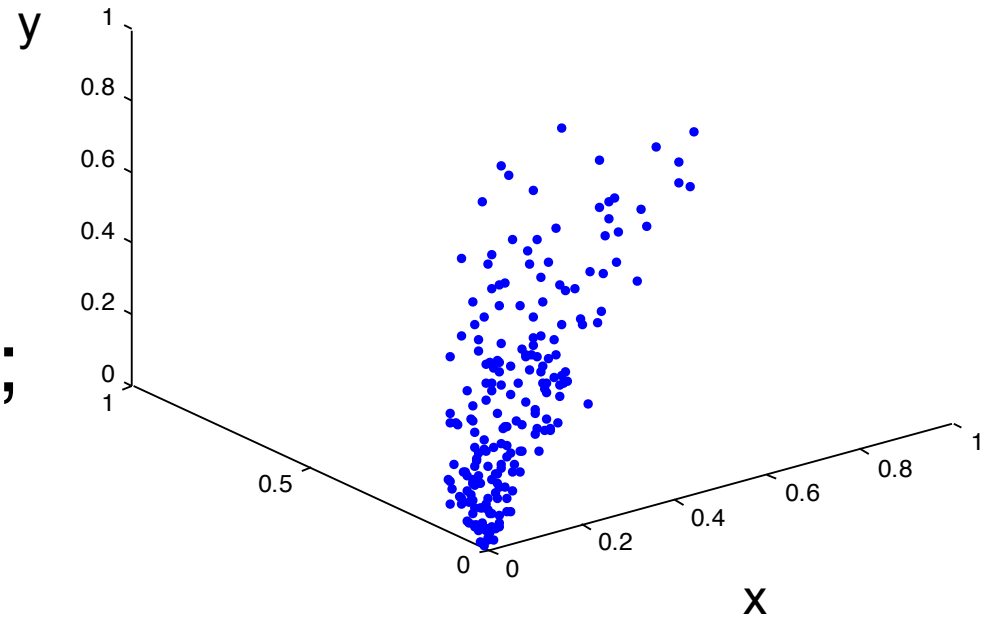
$$f(x_i;\theta) = \sum_{m=1}^{M} r_m \tanh(\mathbf{a}_m^T \mathbf{x} + b_m) + r_0$$

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



Paired data : tau=2

```
x=[];
len=3;
for i=1:n-len
    p=Y(i:i+len-1)';
    x=[x p];
    y(i)=Y(i+len);
end
figure;
plot3(x(2,:),x(1,:),y,'.');hold on;
```

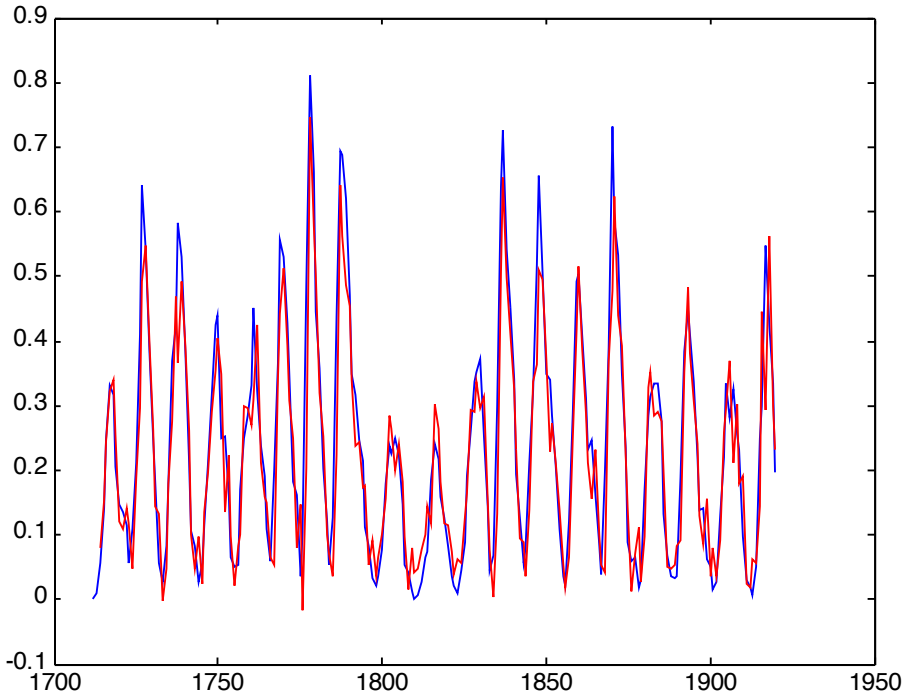


Learning and prediction

```
M=input('keyin the number of hidden units:');
[a,b,r]=learn_MLP(x',y',M);
y=eval_MLP2(x,r,a,b,M);
hold on;
plot(1712+2:1712+2+length(y)-1,y,'r')
```

MSE for training data 0.005029
ME for training data 0.052191K

Prediction



Sunspot

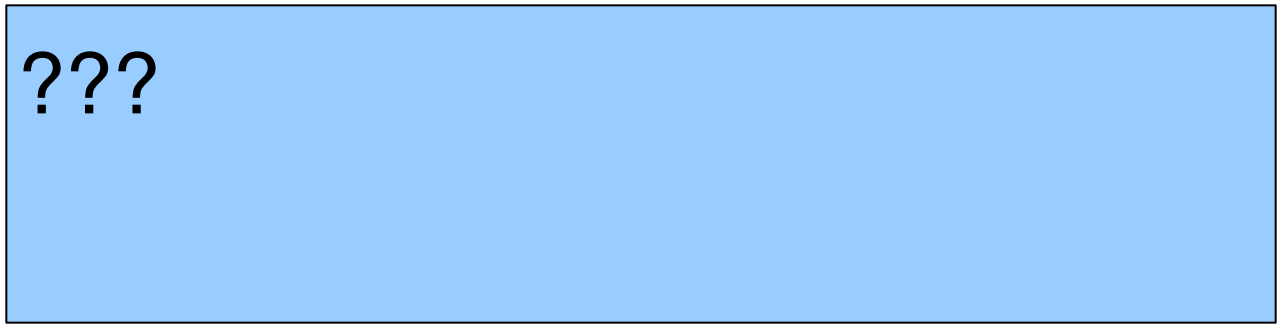
Source codes

Paired data: tau=5

```
x=[];
len=5;
for i=1:n-len
    p=Y(i:i+len-1)';
    x=[x p];
    y(i)=Y(i+len);
end
```

Learning and prediction

```
M=input('keyin the number of hidden units:');
[a,b,r]=learn_MLP(x',y',M);
y=eval_MLP2(x,r,a,b,M);
hold on;
plot(1712+2:1712+2+length(y)-1,y,'r')
```



Example

- Repeat numerical experiments in slide #34 with

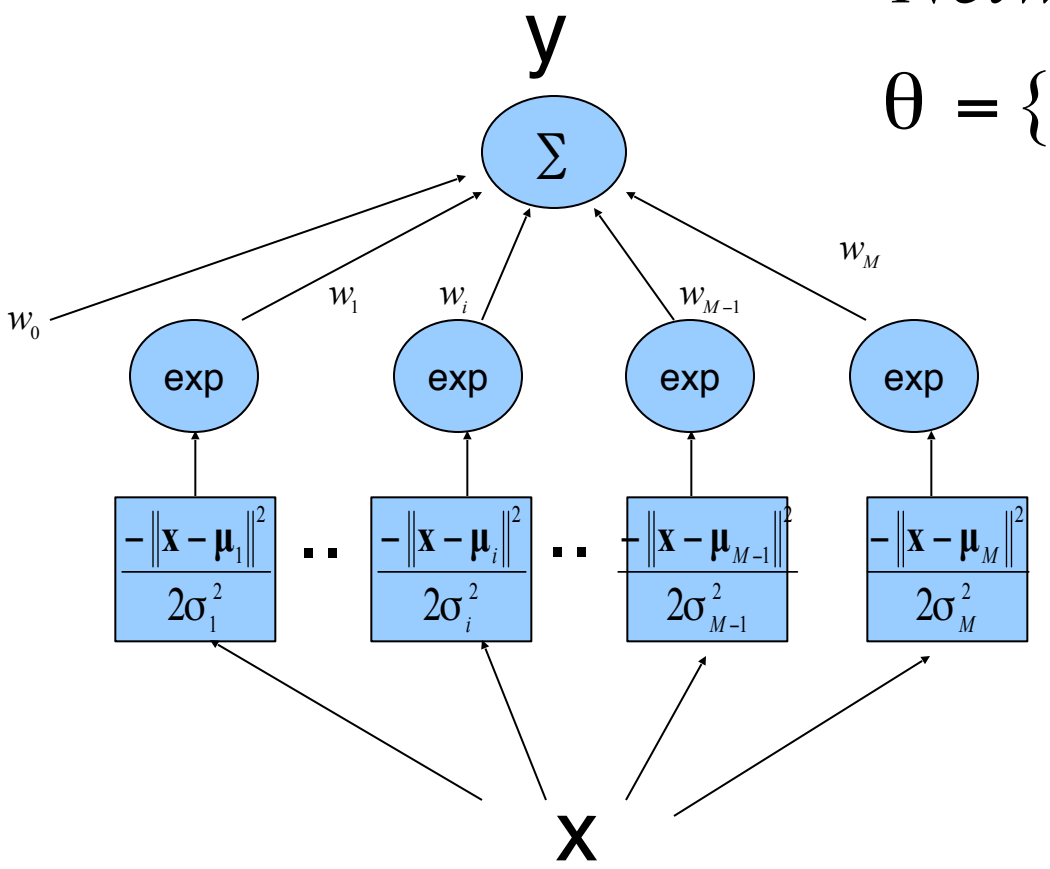
$$\tau = 2, M = 5,10,15,20$$

$$\tau = 5, M = 5,10,15,20$$

$$\tau = 10, M = 5,10,15,20$$

- Summarize your numerical results in a table

RBF



Network parameter

$$\theta = \{w_i\}_i \cup \{\boldsymbol{\mu}_i\}_i \cup \{\sigma_i\}_i$$

```
addpath('..\..\..\Methods\FunAppr\LM_RBF');
% x : Nxd
% y : 1xN
N=200;d=2;
x=rand(N,d)*2-1;
y=tanh(-x(:,1).^2-x(:,2).^2);
eval(['load data\sunspot_train.mat']);  %([1 2 4 6 7 10],:)
tx = X';       ty = Y;
Net=LM_RBF(tx,ty);
[yhat,D]=evaRBF(tx,Net);
mean((ty-yhat).^2)/2
```

One_step_look_ahead prediction

```
eval(['load data\sunspot_test_1.mat']);
tex1 = X';     tey1 = Y;
[yhat,D]=evaRBF(tex1,Net);
mean((tey1-yhat).^2)/2
```