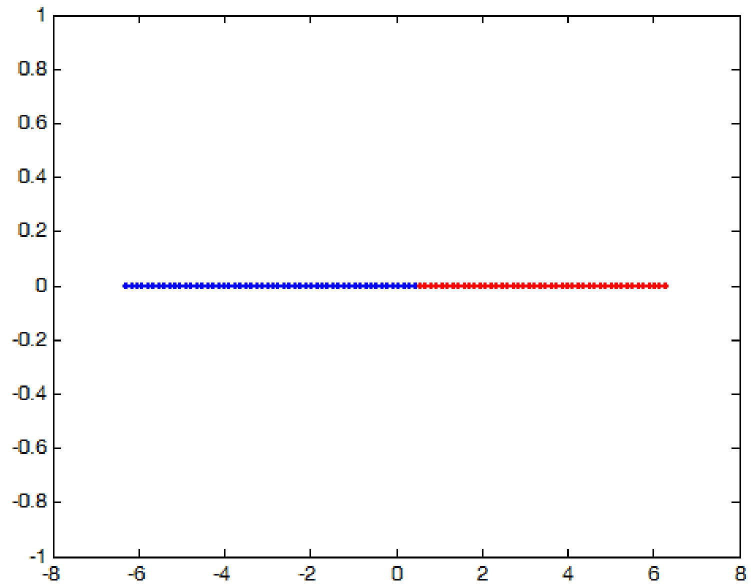
Lecture 10 Function Approximation

- 1D Classification
- Polynomial Fitting
- 1DFA
- Multi-dimensional FA

Two colors

- ```
  load 1dclass
  ind = find(y==0);
  plot(x(ind),y(ind)*0,'.'); hold on
  ind = find(y==1);
  plot(x(ind),y(ind)*0,'r.')
  ```

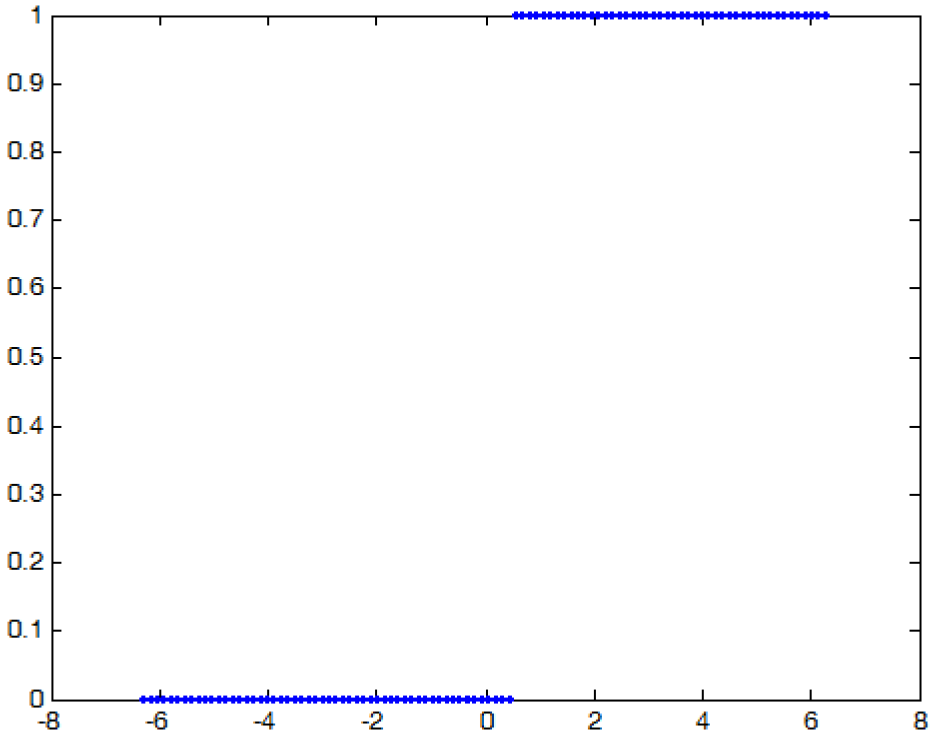

  ```

1D Function Approximation

- 1dclass.zip

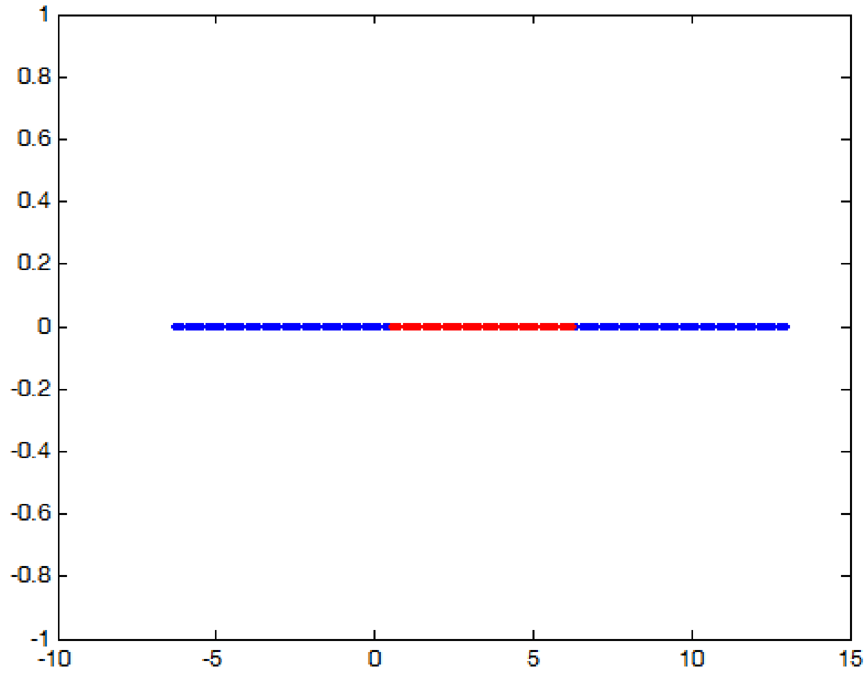
  >> load 1dclass
  >> plot(x,y,'.')

y = 1 for a red point
y = 0 for a blue point



1D Classification

- load 1dclass2
  ind = find(y==0);
  plot(x(ind),y(ind)*0,'.'); hold on
  ind = find(y==1);
  plot(x(ind),y(ind)*0,'r.')

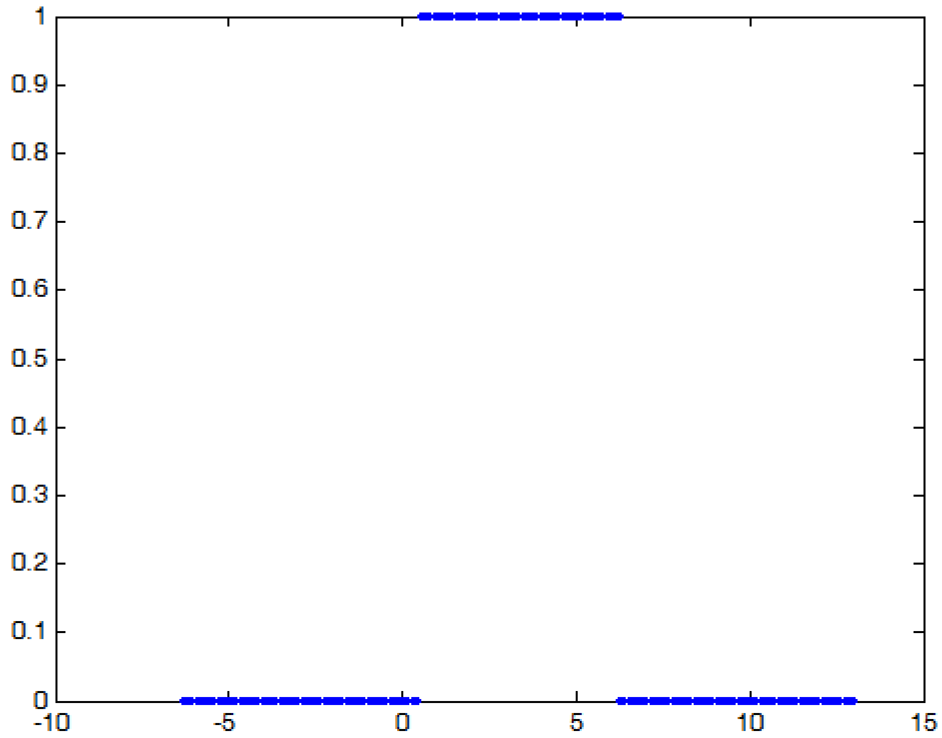


1D Function Approximation

- 1dclass2.zip

>> load 1dclass2
>> plot(x,y,'.')

y = 1 for a red point
y = 0 for a blue point

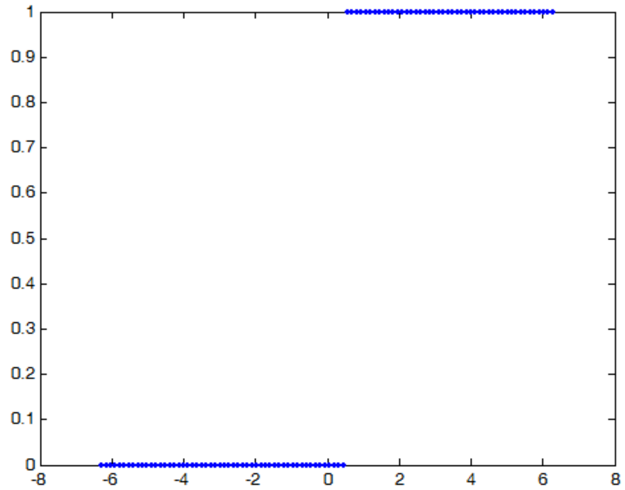


POLYFIT: Fit polynomial to data

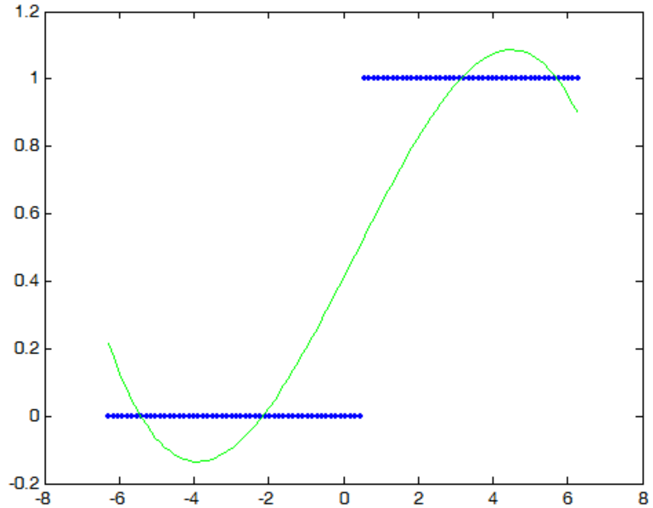
- polyfit(x,y,m)
  - x : input vectors or predictors
  - y : desired outputs
  - m : degree of interpolating polynomial

Example 1

-
```
>> load 1dclass
>> plot(x,y,'.')
```

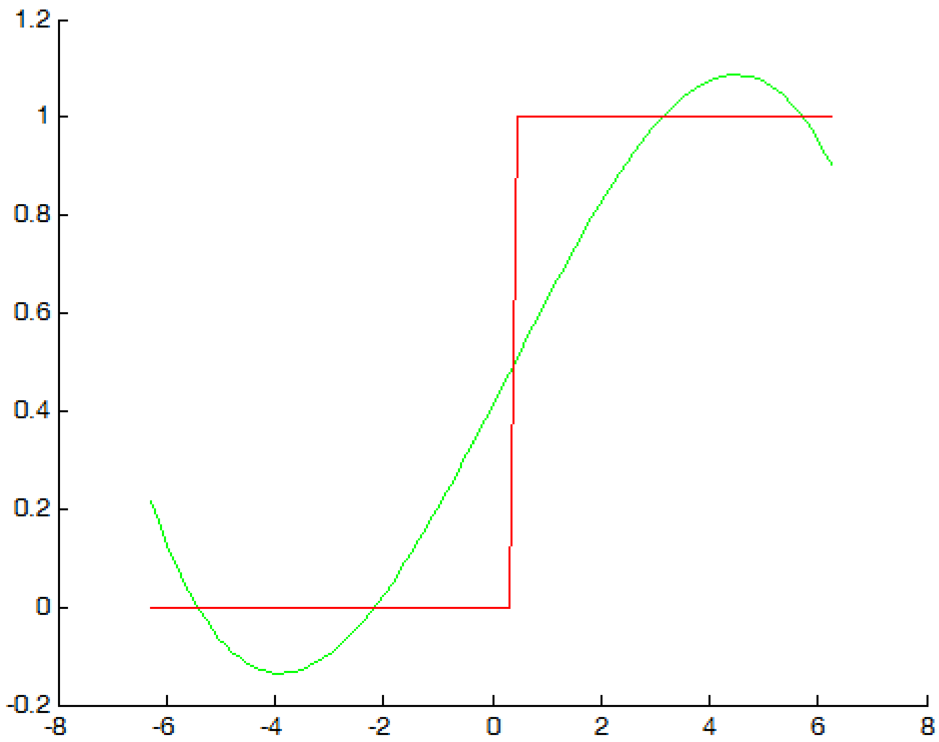



```
m=3;
p=polyfit(x,y,m);
v = linspace(min(x),max(x));
y_hat = polyval(p,v);
hold on
plot(v, y_hat, 'g');
```



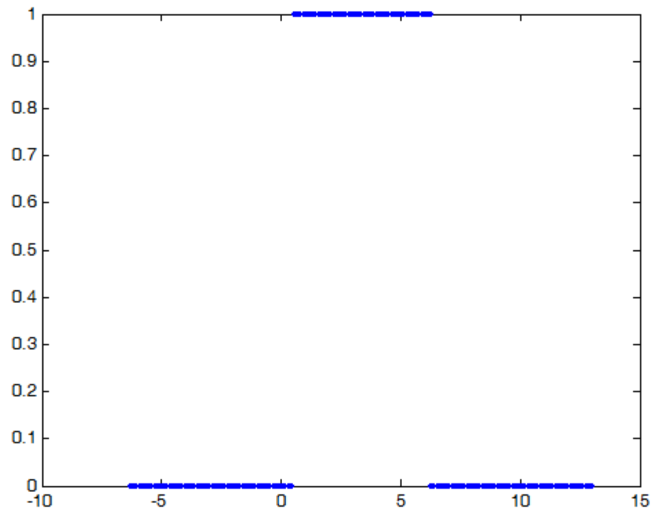
Threshold

```
y_hat = y_hat > 0.5;
hold on
plot(v, y_hat, 'r');
```

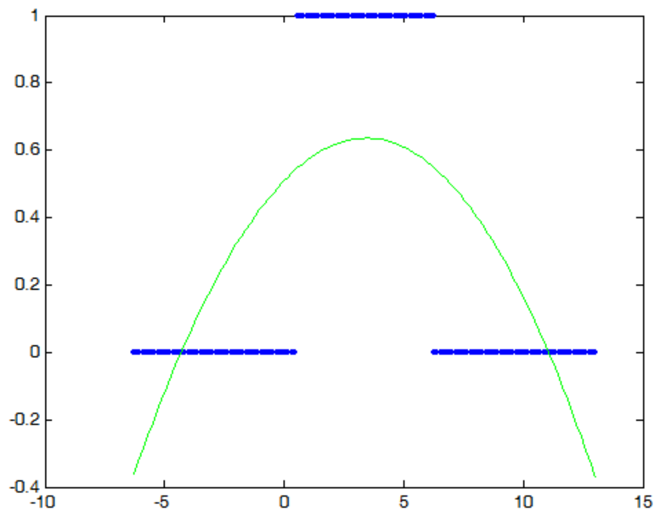


Example 2

```
>> load 1dclass2
>> plot(x,y,'.')
```



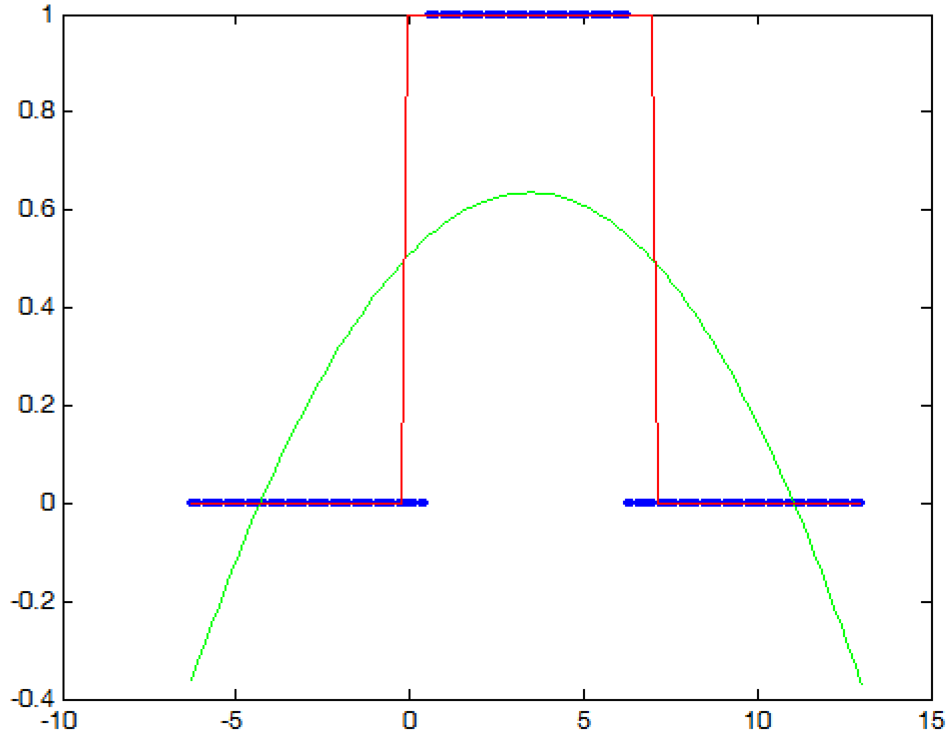

```
m=3;
p=polyfit(x,y,m);
v = linspace(min(x),max(x));
y_hat = polyval(p,v);
hold on
plot(v, y_hat, 'g');
```



Threshold

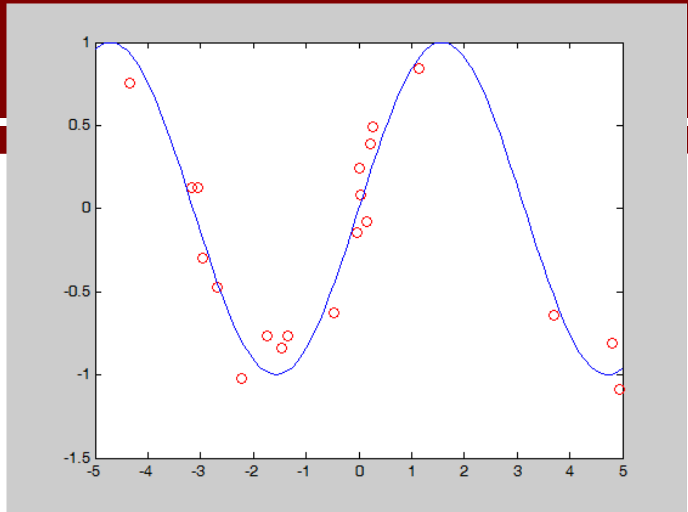
-

```
y_hat = y_hat > 0.5;
hold on
plot(v, y_hat, 'r');
```



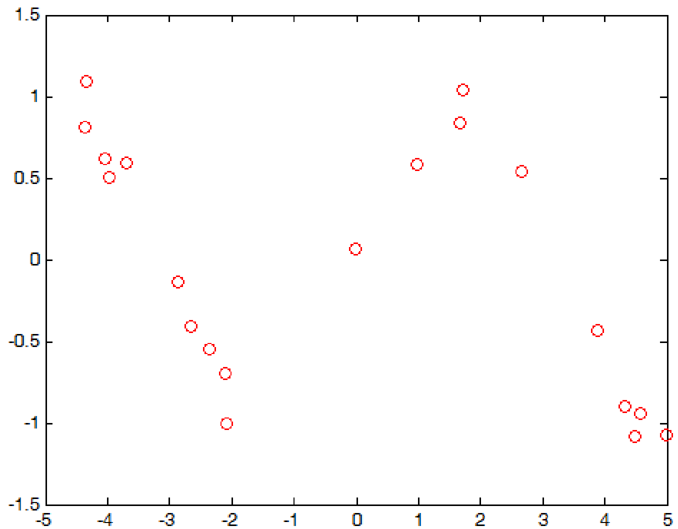
Non-polynomial

- sin



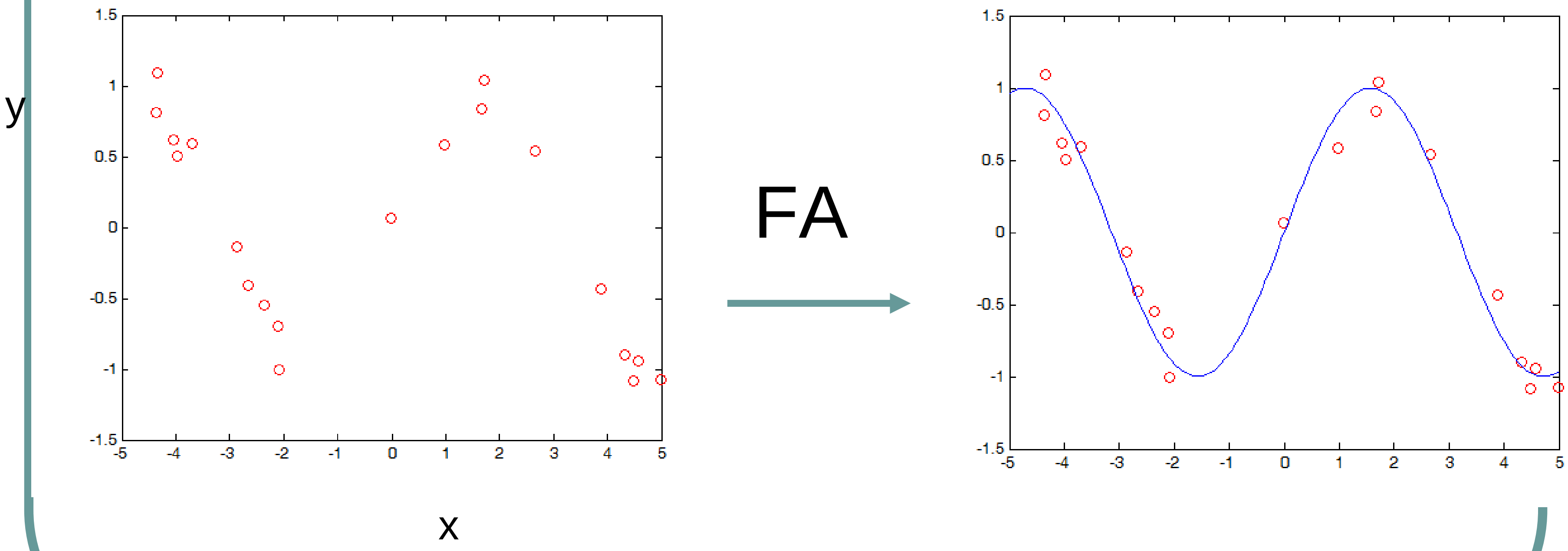

```
fx=inline('sin(x)'); n=20

x=rand(1,n)*10-5;
nois=rand(1,n)*0.5-0.25;
y=fx(x);
y = y+nois
plot(x,y, 'ro');hold on
```



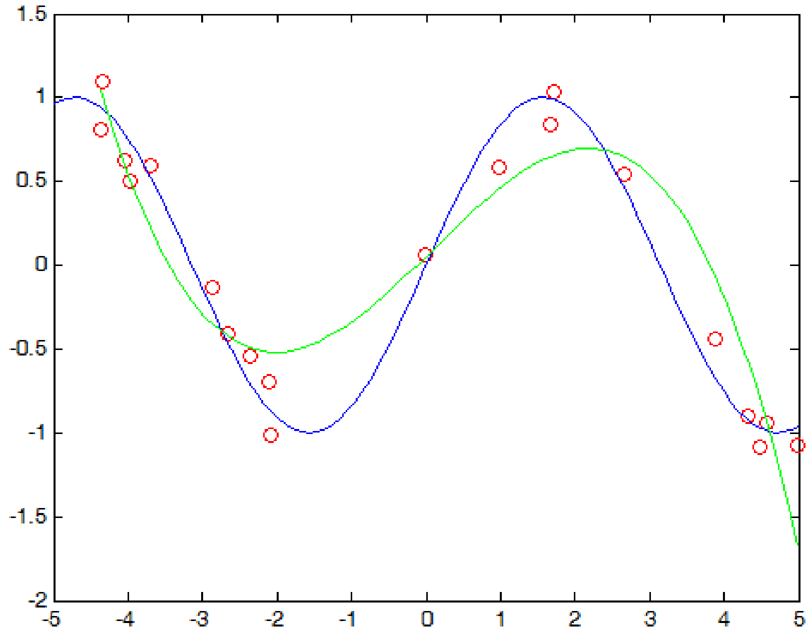
Function approximation

Approximate the original target function subject to given paired data



Polynomial fitting

```
m=3
v=linspace(min(x),max(x));
p=polyfit(x,y,m);hold on;
plot(v,polyval(p,v), 'g');
```



Under-fitting due to approximating non-polynomial by low-degree polynomials

MSE (mean square error)

```
>> mean((polyval(p,x)-y).^2)

ans =

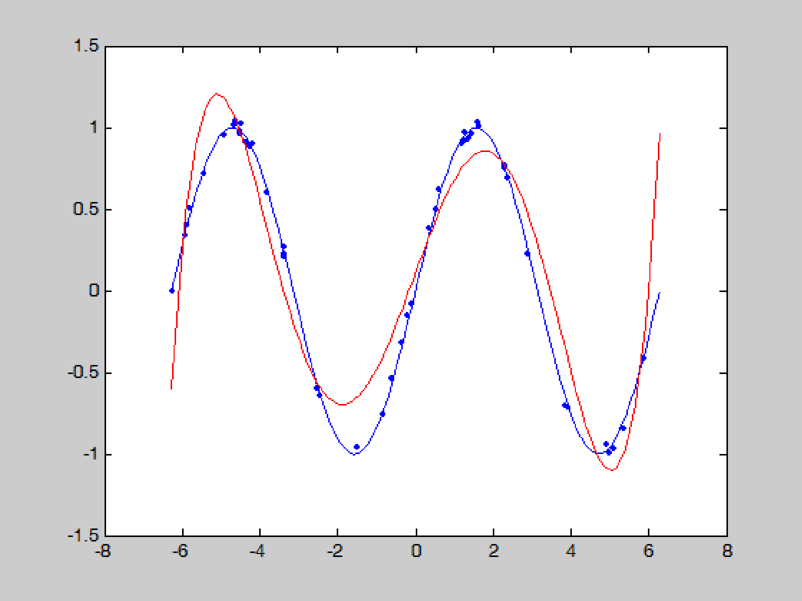
    0.0699
```

Fitting non-polynomial

```
>> fa1d_polyfit
input a function: x.^2+cos(x) :sin(x)
keyin sample size:50
polynomial degree:5

E =

    0.0365
```

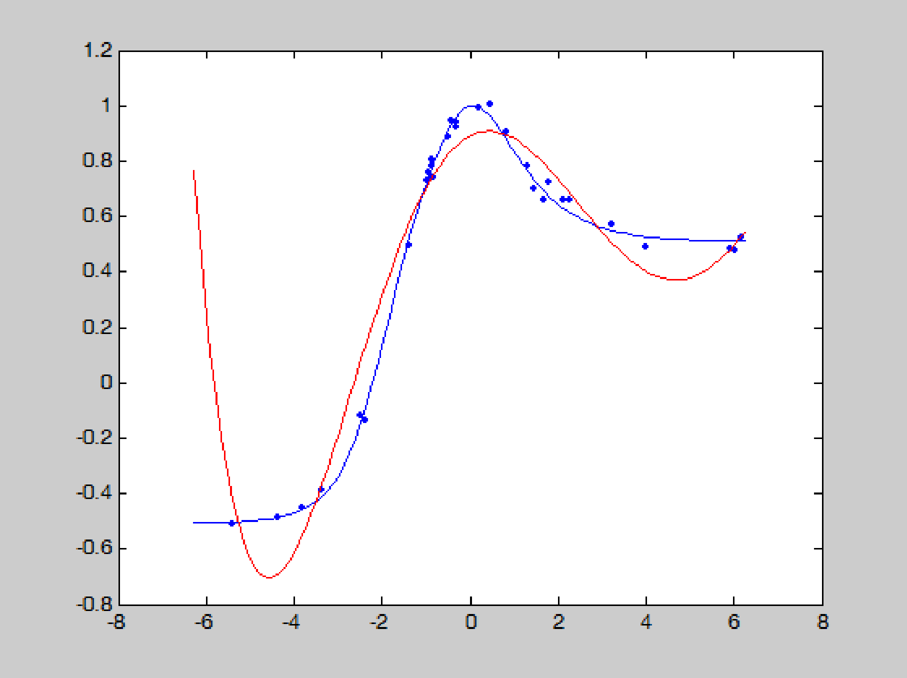


Fitting non-polynomial

```
>> fa1d_polyfit
input a function: x.^2+cos(x) :tanh(x+2)+sech(x)
keyin sample size:30
polynomial degree:5

E =

    0.0097
```



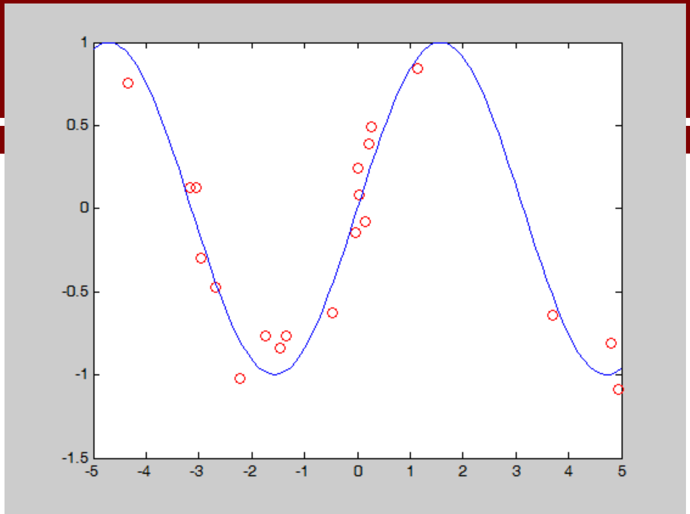
my_kmeans.m
cross_distance.m

K-means based FA

- Apply K-means to find K centers
- Calculate cross distances between K means and N data points
- Categorizing red and blue points based on cross distances

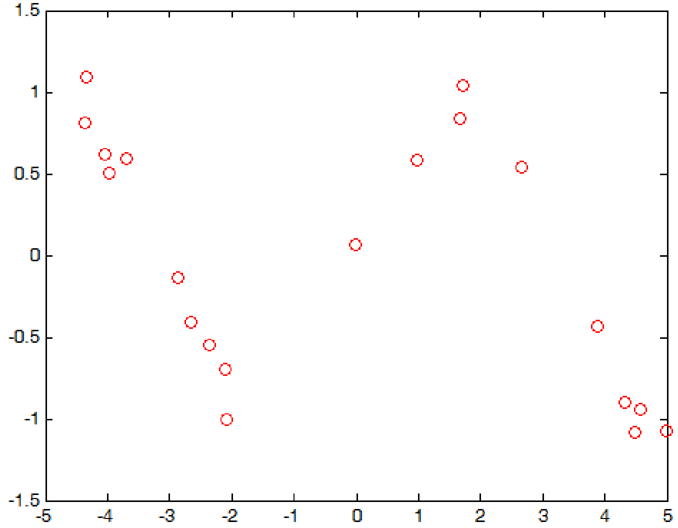
Non-polynomial

- sin



fx=inline('sin(x)'); n=100

x=rand(1,n)*10-5;
nois=rand(1,n)*0.5-0.25;
y=fx(x);
y = y+nois
plot(x,y, 'ro');hold on



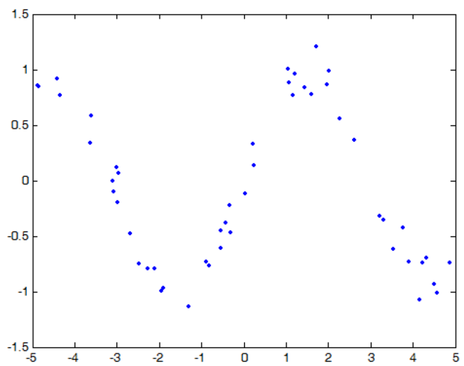
Sampling training data

sinfa.zip

```
load sinfa
```

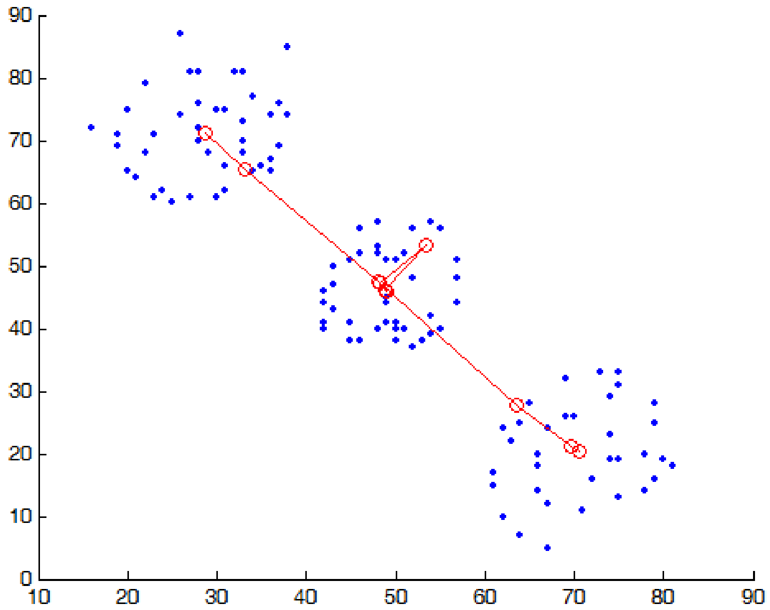
```
N=length(x);
ind=randperm(N);
N2=floor(N/2);
X_TRAIN=x(ind(1:N2));
Y_TRAIN=y(ind(1:N2));
figure
```

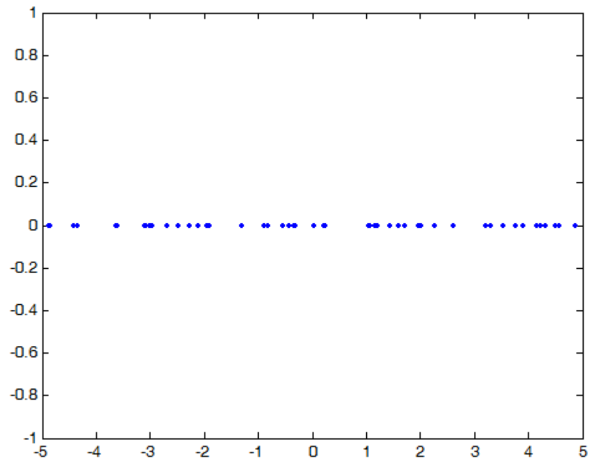
```
plot(X_TRAIN, Y_TRAIN, 'b. '); hold on
```



Step 1

- K-means

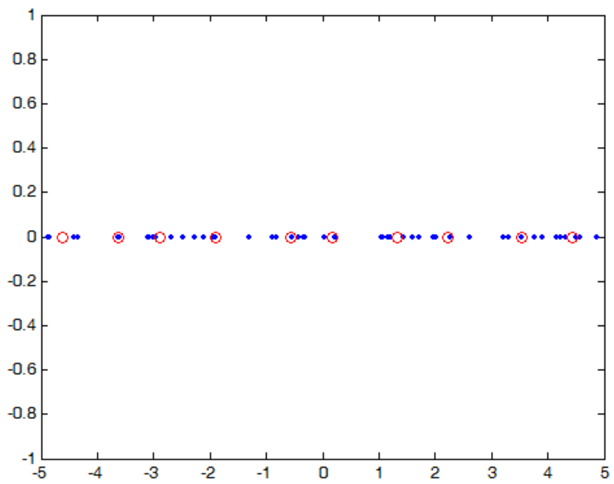




plot(X_TRAIN,zeros(size(X_TRAIN)),'.');hold on

[cind centers]= kmeans(X_TRAIN',10);

plot(centers,zeros(size(centers)),'ro')



Step 2 Cross Distances

- Cross distances between centers and data points

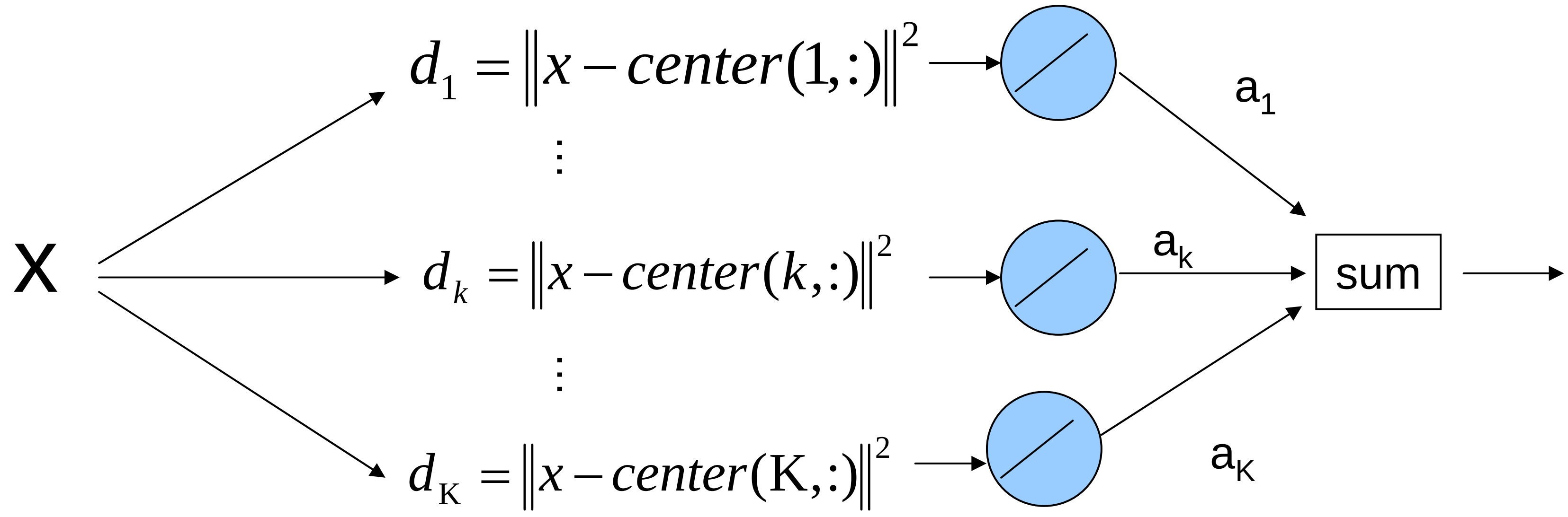
  cross_distance.m

```
D = cross_distance(X_TRAIN',centers);
```

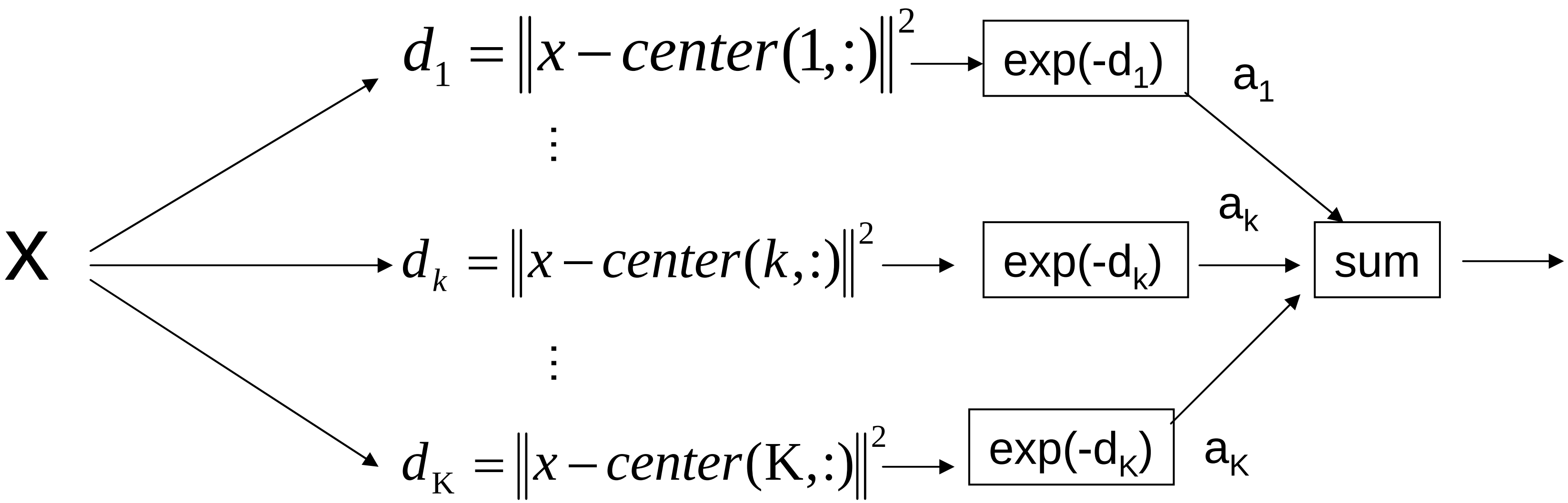
- D(i,j) stores the distance between the ith data point and the jth center

Step 3 Posterior weights

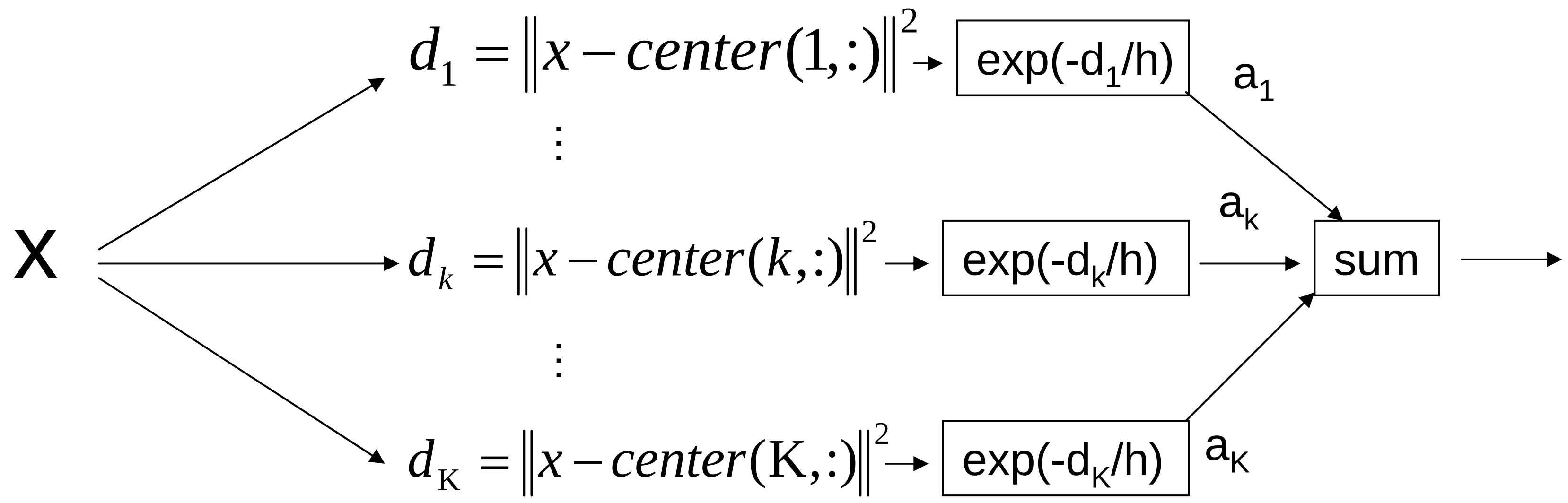
- Radial basis function



Radial basis function



Radial basis function



Approximating function

Substitute the ith data point

$$y(i) = f(x(i,:))$$

$$= \sum_{k=1}^{K} a_k \exp(-\|x(i,:) - center(k,:)\|^2 / h)$$

$$= \sum_{k=1}^{K} a_k \exp(-d_{ik} / h)$$

$$D = [d_{ik}], \mathbf{a} = \begin{bmatrix} a_1 \\ \vdots \\ a_k \\ \vdots \\ a_K \end{bmatrix}, \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_k \\ \vdots \\ y_K \end{bmatrix}$$

Substitute the ith data point

$$y(i) = f(x(i,:))$$

$$= \sum_{k=1}^{K} a_k \exp(-\|x(i,:) - center(k,:)\|^2)$$

$$= \sum_{k=1}^{K} a_k \exp(-d_{ik})$$

⇨ $$\exp(-D/h)\mathbf{a} = \mathbf{y}$$

$$\exp(-D/h)\mathbf{a} = \mathbf{y}$$

↓

$$\mathbf{a} = pinv(\exp(-D/h)) * \mathbf{y}$$

D, Y_TRAIN

↓

```
a = pinv(exp(-D/200))*Y_TRAIN';
```

↓

```
Y_HAT=exp(-D/200)*a;
```

↓

```
N= length(Y_TRAIN);
sum((Y_TRAIN-Y_HAT').^2)/N
```

mse
ans =

0.0141

Approximating function

X_TRAIN

centers

```
D = cross_distance(X_TRAIN',centers);
```

a,h

```
Y_HAT=exp(-D/200)*a;
```

Y_HAT

Approximating function

- X=linspace(-5,5);

centers → D = cross_distance(X',centers);

a,h → Y=exp(-D/200)*a;

plot(X,Y); hold on
plot(X_TRAIN,Y_TRAIN, 'o')

Radial basis function

x

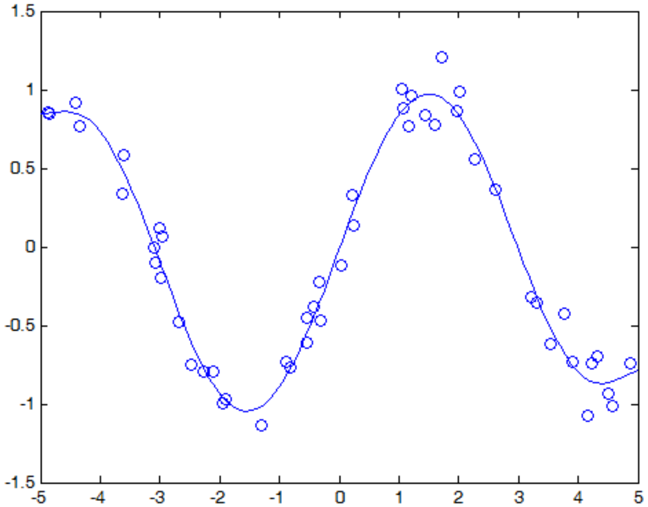
$$d_1 = \|x - center(1,:)\|^2 \rightarrow \exp(-d_1/h) \xrightarrow{a_1}$$

⋮

$$d_k = \|x - center(k,:)\|^2 \rightarrow \exp(-d_k/h) \xrightarrow{a_k} \text{sum} \rightarrow$$

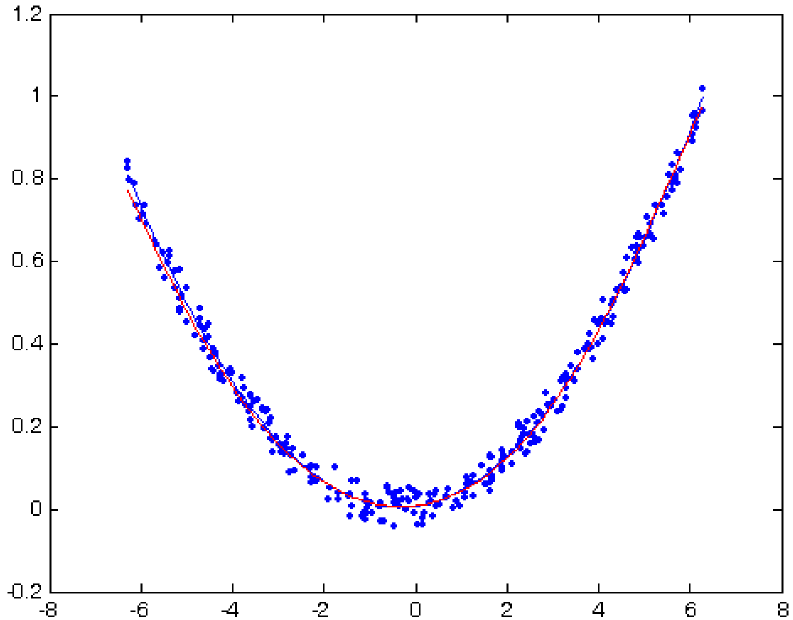
⋮

$$d_K = \|x - center(K,:)\|^2 \rightarrow \exp(-d_K/h) \xrightarrow{a_K}$$



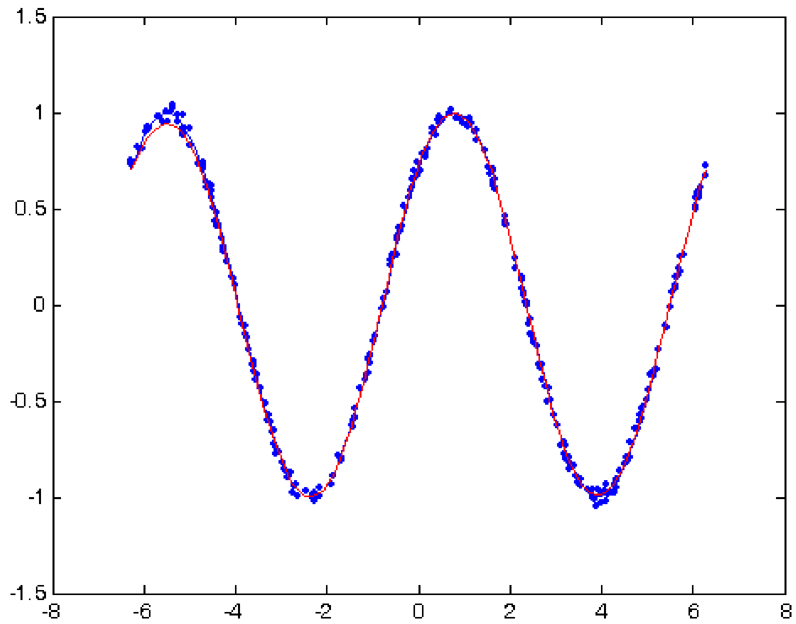
Example

```
>> fa1d
input a function: x.^2+cos(x) :3*x.^2+2*x+1
keyin sample size:300
keyin the number of hidden units:5
```



Example

>> fa1d
input a function: x.^2+cos(x) :cos(x)+sin(x)
keyin sample size:300
keyin the number of hidden units:



Example

```
>> fa2d
input a 2D function: x1.^2+x2.^2+cos(x1) :x1.^2-x2.^2
keyin sample size:300
keyin the number of hidden units:20
```