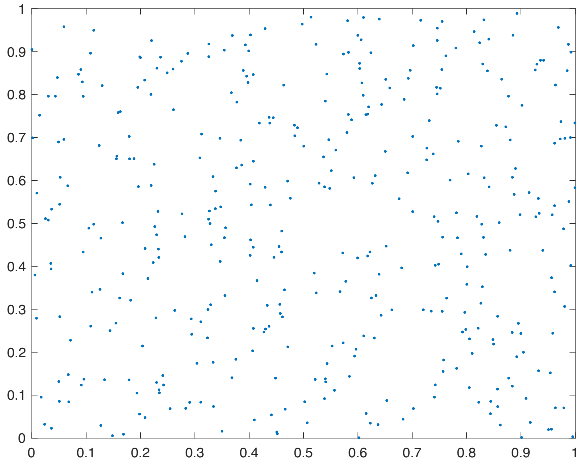
Least Square Method

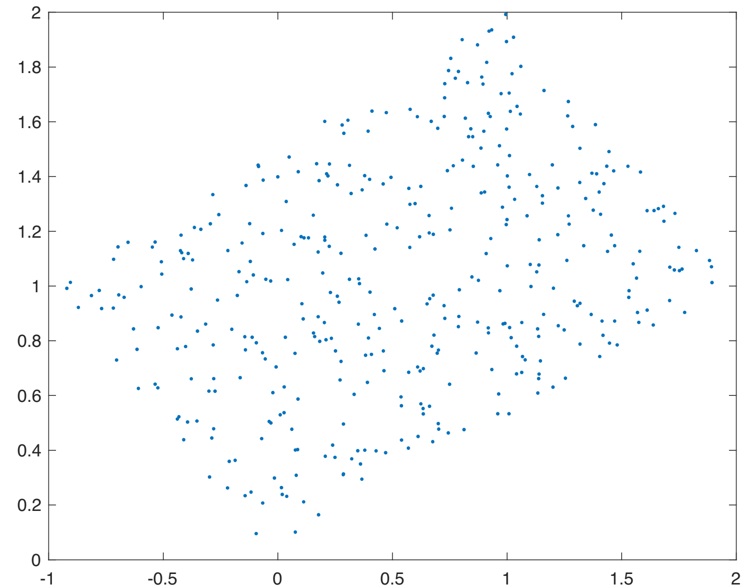
- Transformation Reconstruction
  - Linear Transformation
  - Nonlinear Transformation
- Hyper-plane fitting

•Linear Transformation

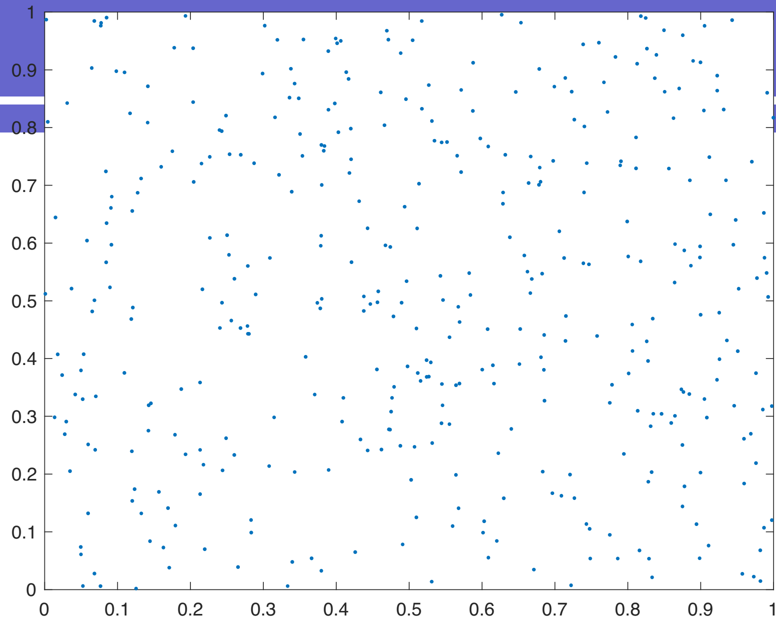


$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} 2 & 1 & -1 \\ 1 & -1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix} = A \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}$$

```
z(:,1) = 2*x(:,1)+x(:,2)-1;
z(:,2)=x(:,1)-x(:,2)+1;
```



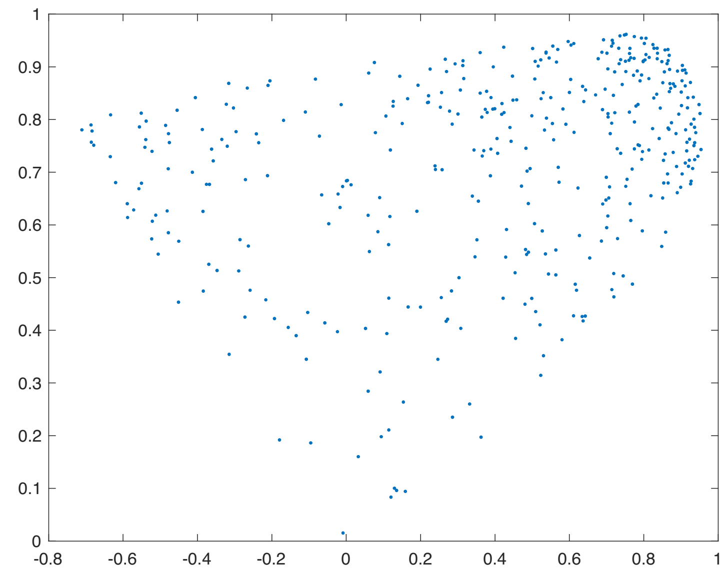
•Nonlinear Transformation



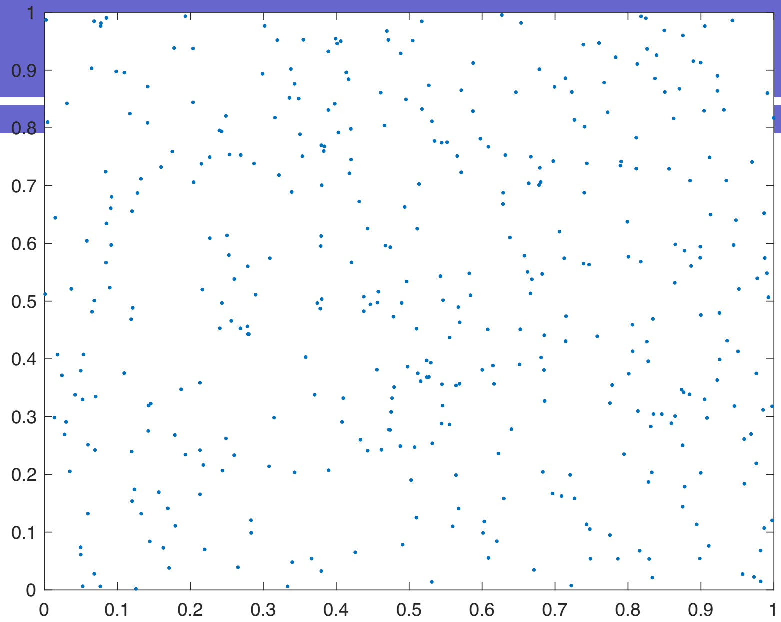
$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = tanh\left(\begin{bmatrix} 2 & 1 & -1 \\ 1 & -1 & 1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}\right)$$

$$= tanh\left(A\begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}\right)$$

```
z(:,1) = tanh(2*x(:,1)+x(:,2)-1);
z(:,2)=tanh(x(:,1)-x(:,2)+1);
```



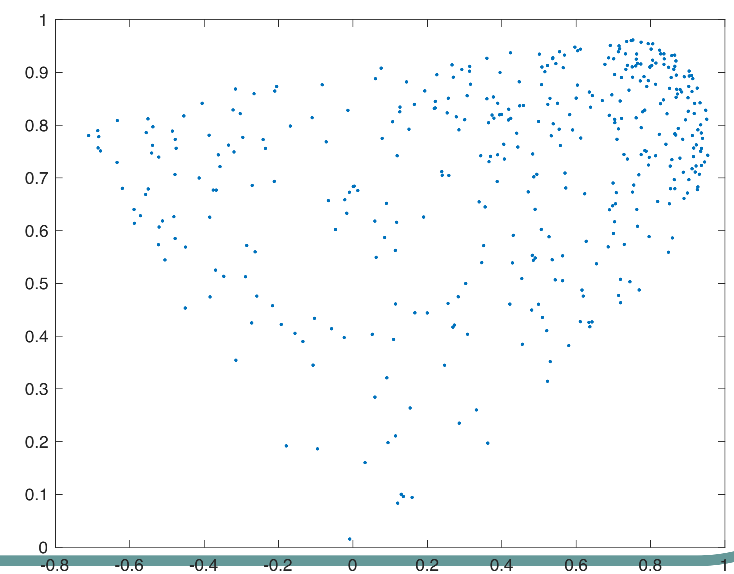
•Nonlinear Transformation



$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = tanh\left(\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a21 & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x2 \\ 1 \end{bmatrix} \right)$$

$$= tanh\left(A \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix} \right)$$

```
z(:,1) = tanh(2*x(:,1)+x(:,2)-1);
z(:,2)=tanh(x(:,1)-x(:,2)+1);
```



Deep Linear-Nonlinear Transformation

Deep Neural Networks

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = tanh\left(A_3 \quad tanh\left(A_2 \quad tanh\left(A_1 \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}\right)\right)\right)$$

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = tanh\left(A_n \cdots \quad tanh\left(A_2 \quad tanh\left(A_1 \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}\right)\right)\right)$$

Data Transformation

$$x[t] = \begin{bmatrix} x_1[t] \\ x_2[t] \end{bmatrix}$$

↓

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = tanh\left(A \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix} \right)$$

↓

$$z[t] = \begin{bmatrix} z_1[t] \\ z_2[t] \end{bmatrix}$$

Data Transformation

$$\begin{bmatrix} x[1] & x[2] & \dots & x[N] \end{bmatrix}$$

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = tanh\left(A \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix} \right)$$

$$\begin{bmatrix} z[1] & z[2] & \dots & z[N] \end{bmatrix}$$

Data Driven Learning

Given input data $\begin{bmatrix} x[1] & x[2] & \dots & x[N] \end{bmatrix}$

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = tanh\left(A \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix} \right)$$

FIND A

Given output data $\begin{bmatrix} z[1] & z[2] & \dots & z[N] \end{bmatrix}$

Deep Learning

Given input data $\begin{bmatrix} x[1] & x[2] & \dots & x[N] \end{bmatrix}$

n>2

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = tanh\left(A_n \cdots \quad tanh\left(A_2 \quad tanh\left(A_1 \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}\right)\right)\right)$$

FIND $A_1, A_2, \dots, A_n$

Given output data $\begin{bmatrix} z[1] & z[2] & \dots & z[N] \end{bmatrix}$

Problem: Learning Linear Transformation

x denotes a matrix of N-by-2

z denotes a matrix of N-by-2

$$E_S(A) = \frac{1}{2N} \sum_t \left\| z[t] - A \begin{bmatrix} x[t] \\ 1 \end{bmatrix} \right\|^2$$

$$minimize \quad E_S(A) \quad with \quad respect \quad to \quad A$$

$$\|x[t]\|^2 = x^T[t]x[t]$$

$$= x_1[t]^2 + x_2[t]^2$$

Problem: Learning NonLinear Transformation

x denotes a matrix of N-by-2

z denotes a matrix of N-by-2

$$E_S(A) = \frac{1}{2N} \sum_t \left\| z[t] - tanh\left(A \begin{bmatrix} x[t] \\ 1 \end{bmatrix} \right) \right\|^2$$

$$minimize \quad E_S(A) \quad with \quad respect \quad to \quad A$$

Problem: Learning NonLinear Transformation

x denotes a matrix of N-by-2

z denotes a matrix of N-by-2

$$E_S(A) = \frac{1}{2N} \sum_t \left\| z[t] - \ tanh\left(A \begin{bmatrix} x[t] \\ 1 \end{bmatrix} \right) \right\|^2$$

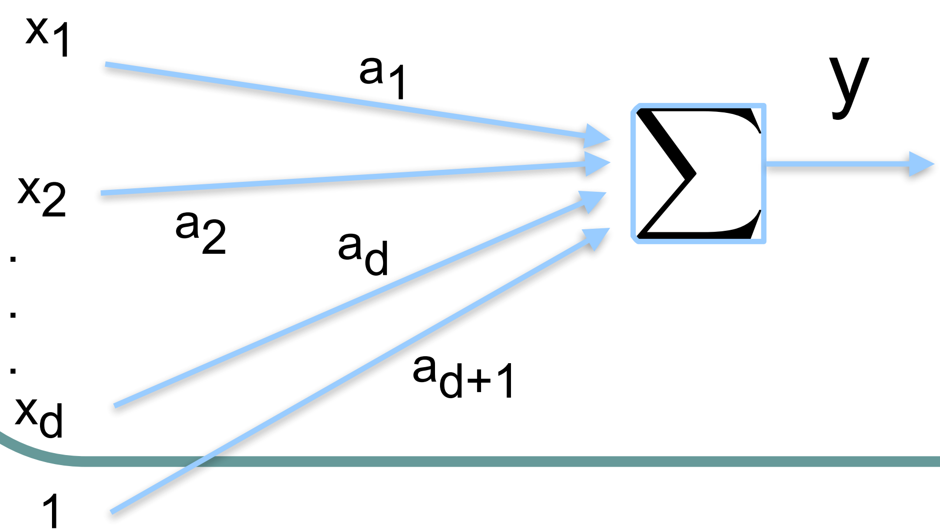
$$minimize \quad E_S(A) \quad with \quad respect \quad to \quad A$$

$$tanh\left(A_n \cdots \ tanh\left(A_2 \ \ tanh\left(A_1 \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix} \right)\right)\right)$$

Learning a linear relation

$$y = x_1 a_1 + x_2 a_2 + x_3 a_3 + \ldots + x_d a_d + a_{d+1}$$

$$y = \begin{bmatrix} a_1 & a_2 & \cdots & a_d & a_{d+1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \\ 1 \end{bmatrix}$$



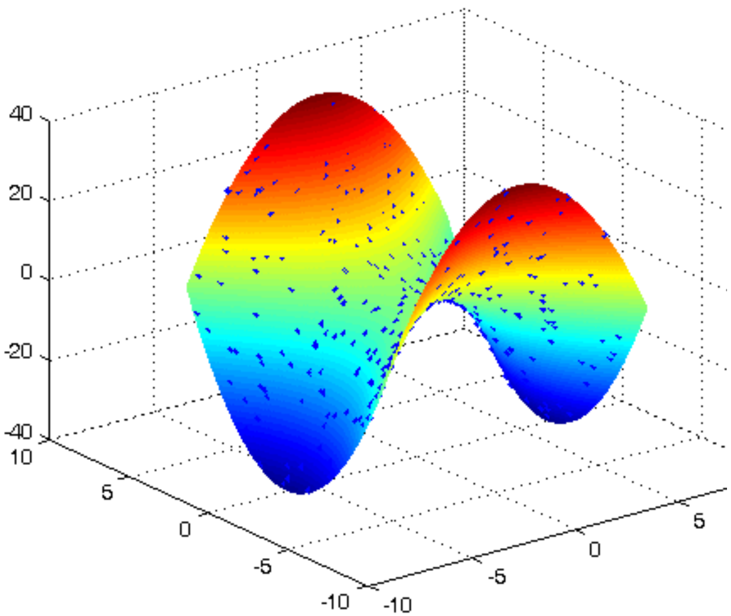
d < n

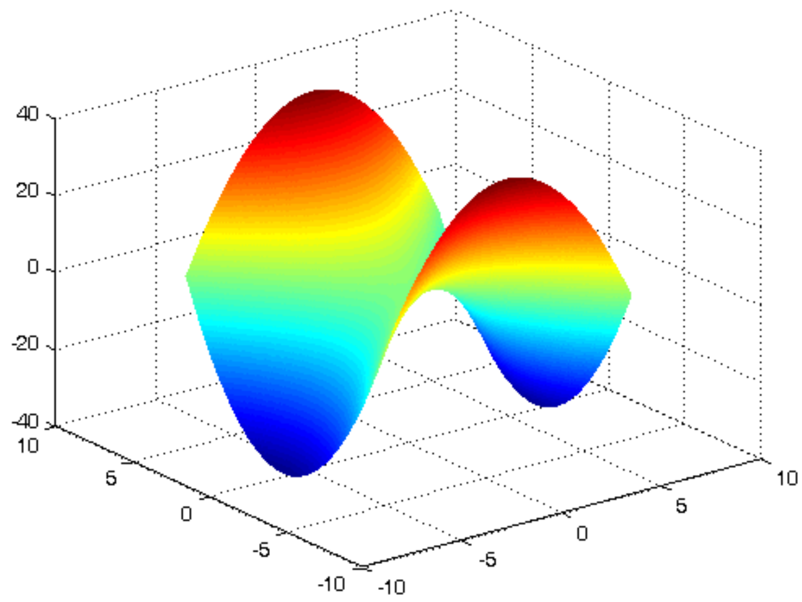
- The number of unknowns is less than the constraint number
  - 400 data points in R^3 space
- Data point on a hyper-plane
  - $x_1[t]a_1+x_2[t]a_2+a_3 - y[t] = e_k$

Minimization of the mean square error

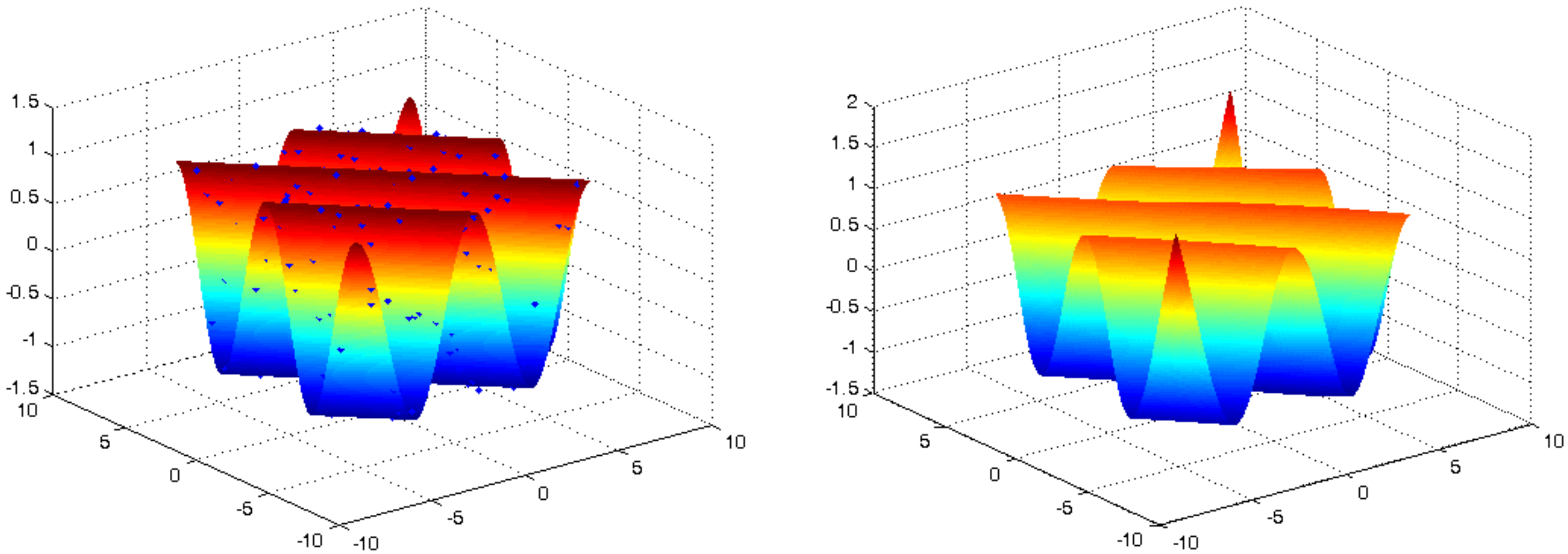
$$\langle e^2 \rangle$$

Surface fitting





Surface fitting



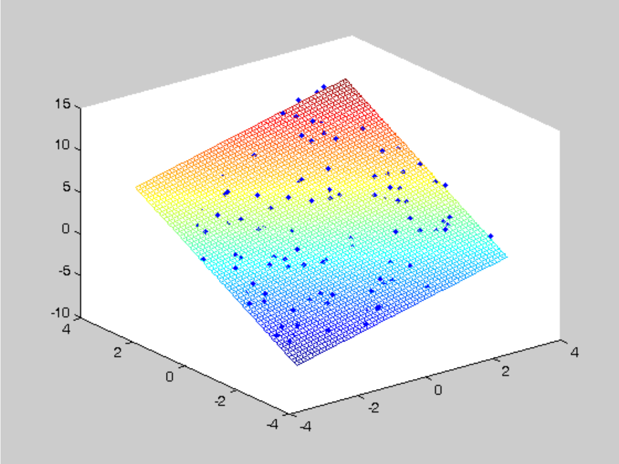
Approximating function

- Dimensionality
  - One dimensional curve
  - High dimensional surface
  - Extremely high dimensional functions
- Linear functions, quadratic functions and nonlinear functions

Sampling

- A mapping from R^2 to R
  - Paired data:

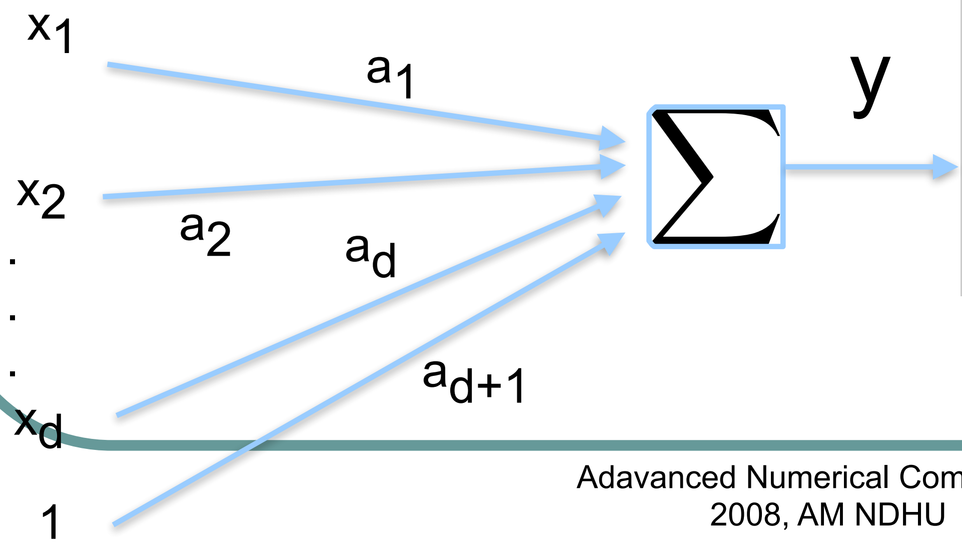
$$S_i = \{(\mathbf{x}[t], y[t]) | \mathbf{x}[t] = (x_1[t], x_2[t]), y[t] = z_i[t]\}_t$$

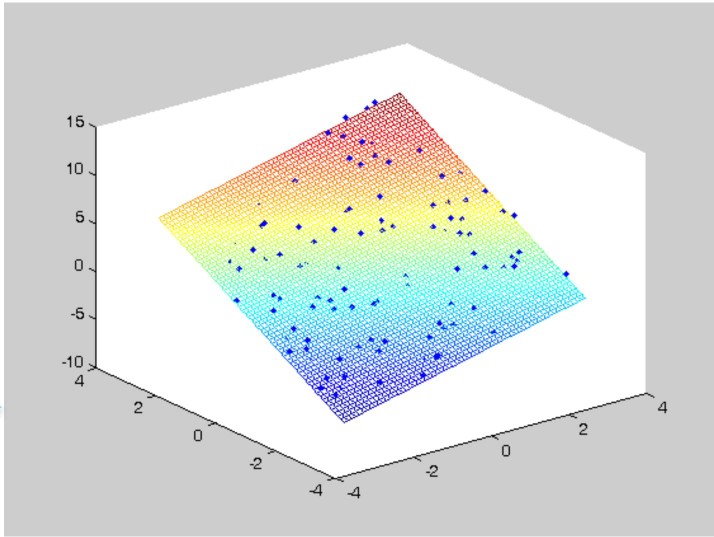


Linear relation

- N=400, d=2
- General coordinate of points $([x_1,x_2],y)$
- Linea relation

$$x_1a_1+x_2a_2+a_3=y$$

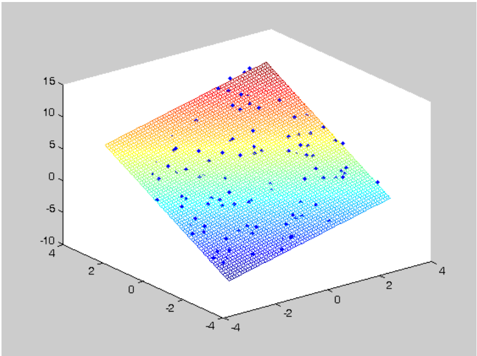




Hyper-plane fitting

- n=400, d=2
- General coordinate of points $([x_1,x_2],y)$
- Linea relation

$$x_1a_1+x_2a_2+a_3=y$$



- n=400, d>2
- General coordinate of points
- $([x_1,x_2,\ldots,x_d],y)$
- Linea relation

$$x_1a_1+x_2a_2+\ldots+x_da_d+a_{d+1}=y$$

data

$$\mathbf{Xa} = \mathbf{b}$$

$$\mathbf{X}_{N\times m} = \begin{pmatrix} x_{11} & x_{12} & x_{13} & \cdots & x_{1m} \\ x_{21} & x_{22} & x_{23} & \cdots & x_{2m} \\ x_{31} & x_{32} & x_{33} & \cdots & x_{3m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & x_{N3} & \cdots & x_{Nm} \end{pmatrix} \quad \mathbf{a} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_m \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ ? \\ b_n \end{pmatrix}$$

$$m = d + 1$$

$$\mathbf{Xa} = \mathbf{b}$$

$$\mathbf{X}_{N\times m} = \begin{pmatrix} x_{11} & x_{12} & x_{13} & \cdots & x_{1m} \\ x_{21} & x_{22} & x_{23} & \cdots & x_{2m} \\ x_{31} & x_{32} & x_{33} & \cdots & x_{3m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & x_{N3} & \cdots & x_{Nm} \end{pmatrix} \quad \mathbf{a} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ ? \\ a_m \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_N \end{pmatrix} = \begin{pmatrix} y[1] \\ y[2] \\ y[3] \\ \vdots \\ y[N] \end{pmatrix}$$

m=d+1 and $x_{im} = 1$

$$\mathbf{x}_i^T \mathbf{a} = \mathrm{b}_\mathrm{i} \quad \text{for i} = 1,\ldots,\mathrm{n}$$

Strategy I : Pseudo Inverse

$$\mathbf{Xa} = \mathbf{b}$$

$$\mathbf{a} = \mathrm{pinv}(\mathrm{X})\mathbf{b}$$

Strategy II: minimizing mean square errors

$$Minimize$$

$$E(\mathbf{a}) = \left\langle e^2 \right\rangle$$

$$= \frac{1}{2n}\sum_{i=1}^{n} e_i^2 = \frac{1}{2n}\sum_{i=1}^{n} (\mathbf{x}_i^T \mathbf{a} - b_i)^2$$

$$n = N$$

Minimization

$$\frac{dE(\mathbf{a})}{da_j} = 0 \text{ for j} = 1, \ldots, \text{m}$$

Derivative

Minimize

$$E(\mathbf{a}) = \frac{1}{2n}\sum_{i=1}^{n}(\mathbf{x}_i^T\mathbf{a} - b_i)^2$$

$$\frac{dE(\mathbf{a})}{da_j} = \frac{2}{2n}\sum_{i=1}^{n}(\mathbf{x}_i^T\mathbf{a} - b_i)x_{ij} = 0$$

Vector Form

Minimize

$$E(\mathbf{a}) = \frac{1}{2n}\sum_{i=1}^{n}(\mathbf{x}_i^T\mathbf{a} - b_i)^2$$

$$\frac{dE(\mathbf{a})}{d\mathbf{a}} = \frac{2}{2n}\sum_{i=1}^{m}(\mathbf{x}_i^T\mathbf{a} - b_i)\mathbf{x}_i = 0$$

Linear system: normal equations

$$\frac{dE(\mathbf{a})}{d\mathbf{a}} = \frac{2}{2n}\sum_{i=1}^{n}(\mathbf{x}_i^T\mathbf{a} - b_i)\mathbf{x}_i = 0$$

$$\Rightarrow \sum_{i=1}^{n}\mathbf{x}_i^T\mathbf{a}\mathbf{x}_i = \sum_{i=1}^{n} b_i\mathbf{x}_i$$

$$\sum_{i=1}^{n} \mathbf{x}_i^T \mathbf{a} \mathbf{x}_i = \sum_{i=1}^{n} b_i \mathbf{x}_i$$

$$\Rightarrow$$

$$\sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^T \mathbf{a} = \sum_{i=1}^{n} b_i \mathbf{x}_i$$

$$X = \begin{pmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \boxed{?} \\ \mathbf{x}_n^T \end{pmatrix} \qquad X^T = \begin{pmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \boxed{?} & \mathbf{x}_n \end{pmatrix}$$

$$X^T X = \begin{pmatrix} \mathbf{x}_1 & \mathbf{x}_2 & ? & \mathbf{x}_n \end{pmatrix} \begin{pmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ ? \\ \mathbf{x}_n^T \end{pmatrix}$$

$$= \sum_j \mathbf{x}_j \mathbf{x}_j^T$$

$$\sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^T \mathbf{a} = \sum_{i=1}^{n} b_i \mathbf{x}_i$$

$$\Rightarrow$$

$$\mathbf{X}^{\mathrm{T}} \mathbf{X} \mathbf{a} = \mathbf{X}^{\mathrm{T}} \mathbf{b}$$

$$\mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{a} = \mathbf{X}^{\mathrm{T}}\mathbf{b}$$

$$\mathbf{a} = (\mathbf{X}^{T}\mathbf{X})^{-1}\mathbf{X}^{T}\mathbf{b}$$

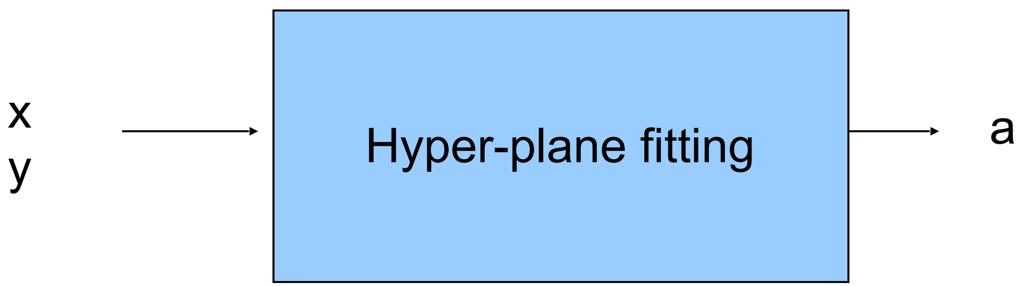
Comparison

- d=2,n=100

```
d=2;n=100;
x=rand(d, n); y=rand(1,d)*x+2+rand(1, n)*0.1-0.05;
plot(x,y,'.');
X=[x' ones(n,1)];
b=y';
tstart = tic;
a=pinv(X)*b
telapsed = toc(tstart)
```

Hyper-plane fitting

- Step 1. Input paired data, (**x**[t] , y[t]), t=1…n
- Step 2. Form matrix X and vector b
- Step 3. Set a to pinv(X)*b
- Step 4. Set c to $(X^T X)^{-1} X^T \mathbf{b}$



```
>> n=30;S=rand(n,2);y=S*[1 2]'+1;
>> b=y;
>> X=[S ones(n,1)];
>> a=pinv(X)*b; c=inv(X'*X)*(X'*b);
>> sum(abs(a-c))

ans =

  1.0547e-015
```

Vector Form

Minimize

$$E(\mathbf{a}) = \frac{1}{2n}\sum_{i=1}^{n}(\mathbf{x}_i^T\mathbf{a} - b_i)^2$$

$$\nabla(\mathbf{a}) = \frac{dE(\mathbf{a})}{d\mathbf{a}} = \frac{2}{2n}\sum_{i=1}^{n}(\mathbf{x}_i^T\mathbf{a} - b_i)\mathbf{x}_i$$

Gradient descent method

- Updating

$$\nabla(\mathbf{a}) = \frac{dE(\mathbf{a})}{d\mathbf{a}} = \frac{2}{2n}\sum_{i=1}^{n}(\mathbf{x}_i^T\mathbf{a} - b_i)\mathbf{x}_i$$

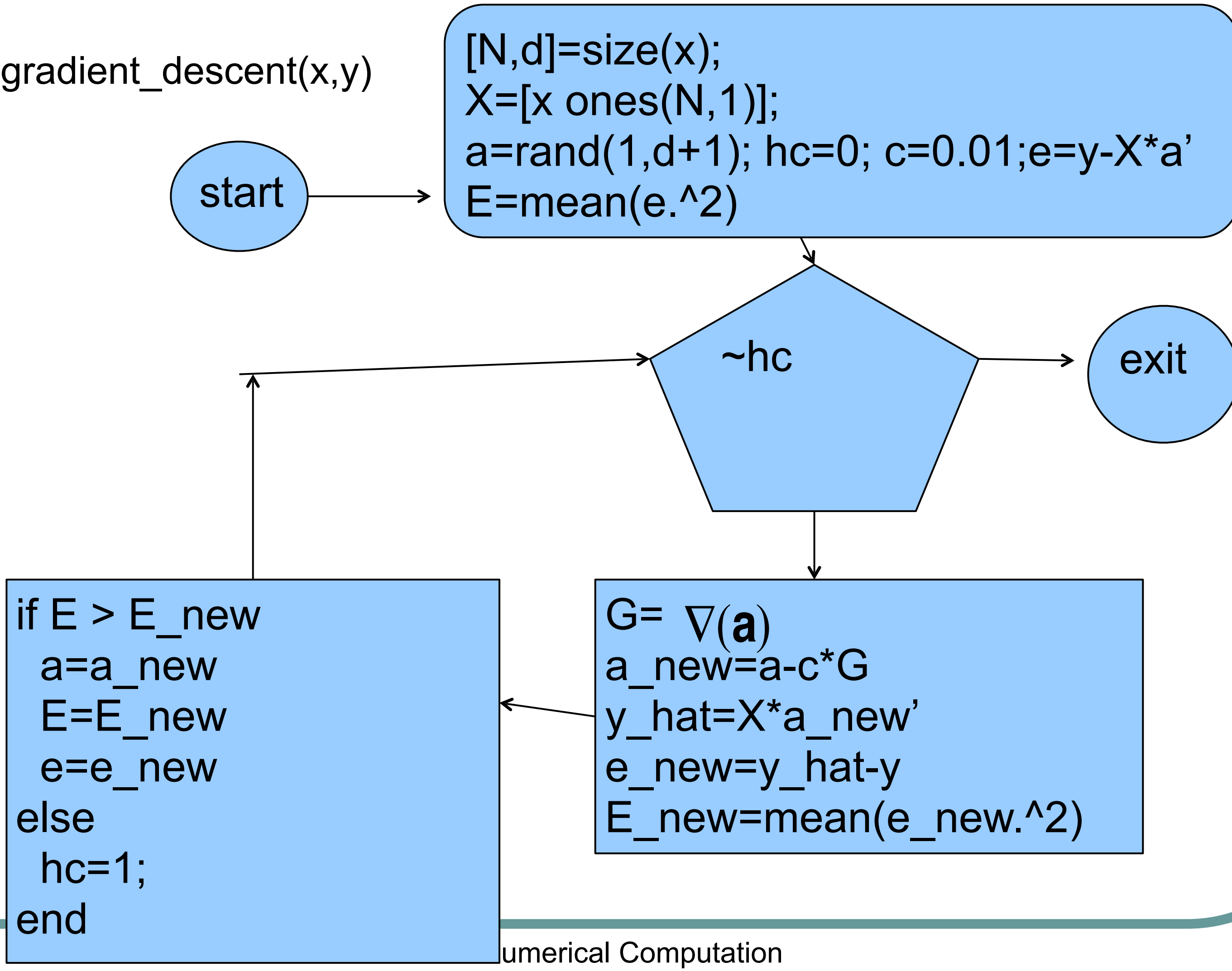
$$= \frac{1}{n}\sum_{i=1}^{n} e_i \mathbf{x}_i$$

```
G=mean(X.*(e*ones(1,d+1)))
```

$$\mathbf{a} \leftarrow \mathbf{a} - \lambda\nabla(\mathbf{a})$$

Flow chart

function a=gradient_descent(x,y)



Vector Form

Minimize

$$E(\mathbf{a}) = \frac{1}{2n}\sum_{i=1}^{n}\sum_{j=1}^{2}(\mathbf{x}_i^T\mathbf{a}_j - b_{ji})^2$$

$$\nabla(\mathbf{a}) = \frac{dE(\mathbf{a})}{d\mathbf{a}}$$

$$= [\frac{dE(\mathbf{a})}{d\mathbf{a}_1} \quad \frac{dE(\mathbf{a})}{d\mathbf{a}_2}]$$

$$= [\frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i^T\mathbf{a}_1 - b_{1i})\mathbf{x}_i \quad \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i^T\mathbf{a}_2 - b_{2i})\mathbf{x}_i]$$

$$= \frac{1}{n}[\sum_{i=1}^{n} e_{1i}\mathbf{x}_i \quad \sum_{i=1}^{n} e_{2i}\mathbf{x}_i]$$

Gradient descent method

- Updating

$$\nabla(\mathbf{a}) = \frac{dE(\mathbf{a})}{d\mathbf{a}}$$

$$\mathbf{a} \leftarrow \mathbf{a} - \lambda \nabla(\mathbf{a})$$

MIMO

- Two-input-two-output
- Revise the flow chart
- Execute twice

```
z(:,1) = 2*x(:,1)+x(:,2)-1;
z(:,2)=x(:,1)-x(:,2)+1;
a(1,: )=gradient_descent(x,z(:,1 ));
a(2,: )=gradient_descent(x,z(:,2 ));
```

Flow chart

function a=gradient_descent(x,y)