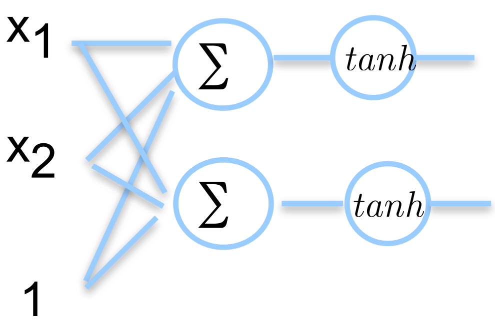
Least Square Method

Nonlinear Transformation Reconstruction

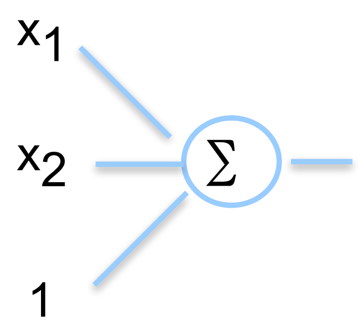
Learning nonlinear transformation



```
x=rand(400,2);
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 ))
```

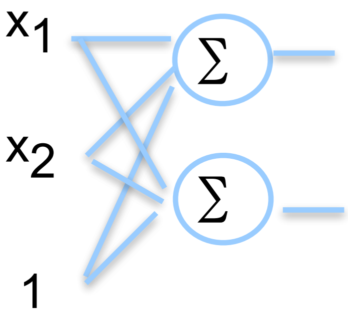
Learning a linear relation

```
function a=gradient_descent(x,y)
max_loop=2000;
[N,d]=size(x);
X=[x ones(N,1)];
a=rand(1,d+1); hc=0; c=0.1;e=y-X*a';
E=100; loop=1;
while ~hc
    G=mean(X.*(e*ones(1,d+1)));
    a_new=a-c*G;
    y_hat=X*a_new';
    e_new=y_hat-y;
    E_new=mean(e_new.^2);
    if mod(loop,100)==0
    fprintf('loop %d  mse %f\n',loop,E_new);
    end
    if E_new < E & loop < max_loop
        a=a_new;e=e_new;
        E=E_new;
    else
        hc=1;
    end
    loop=loop+1;
end
```



Two linear relations

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




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

Vector Form of gradients

$$Minimize$$

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

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

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

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

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

Gradient descent method

- Updating

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

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

Learning many linear relations

```
a=rand(1,d+1); hc=0; c=0.1;e=y-X*a';
E=100; loop=1;
while ~hc
    G=mean(X.*(e*ones(1,d+1)));
    a_new=a-c*G;
    y_hat=X*a_new';
    e_new=y_hat-y;
    E_new=mean(e_new.^2);
```

error, e, is a column vector

1. One linear relation
2. Multiple input single output

error matrix, e, contains two column vectors

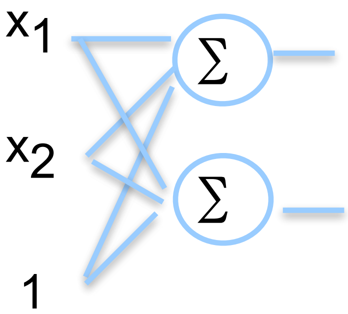
1. Many linear relations
2. Multiple input multiple output

```
a=rand(2,d+1);
hc=0; c=0.1;e=y-X*a';
E=100; loop=1;
while ~hc
 G=[mean(X.*(e(:,1)*ones(1,d+1))); mean(X.*(e(:,2)*ones(1,d+1)))];
 a_new=a-c*G;
 y_hat=X*a_new';
 e_new=y_hat-y;
 E_new=sum(mean(e_new.^2));
```

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

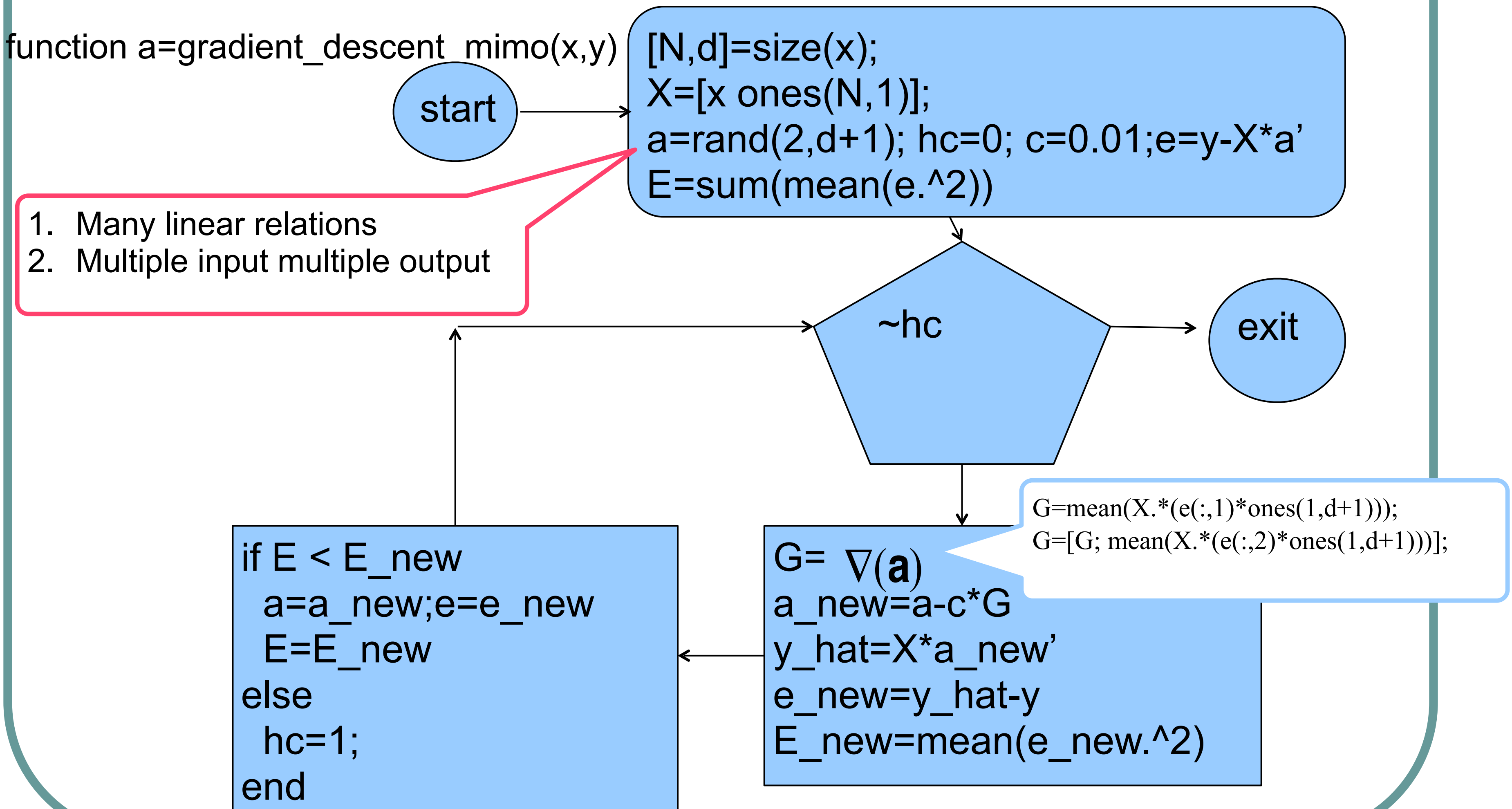
MIMO

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



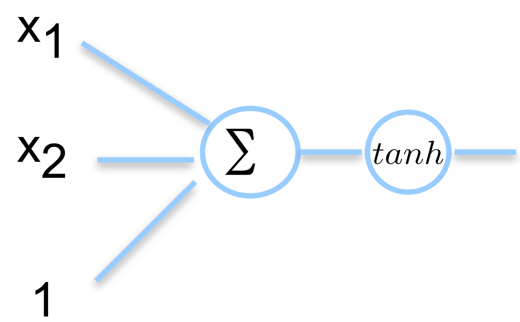

```
z(:,1) = 2*x(:,1)+x(:,2)-1;
z(:,2)=x(:,1)-x(:,2)+1;
a=gradient_descent(x,z );
```

Linear MIMO transformation



Nonlinear Transformation

- tanh based nonlinear relation



$$y = \tanh(\mathbf{x}^T \mathbf{a})$$

$$a \in R^3$$

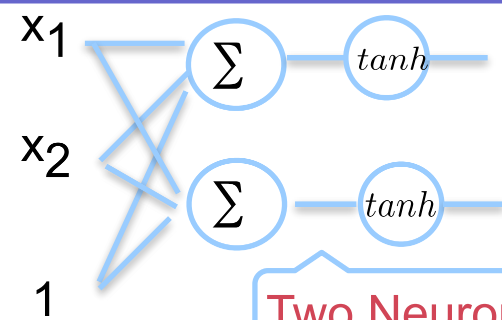
$$\frac{dy}{d\mathbf{a}} = \frac{d\tanh(\mathbf{x}^T \mathbf{a})}{d\mathbf{a}}$$

$$= (1 - \tanh^2(\mathbf{x}^T \mathbf{a}))\mathbf{x}$$

Two tanh based nonlinear relations

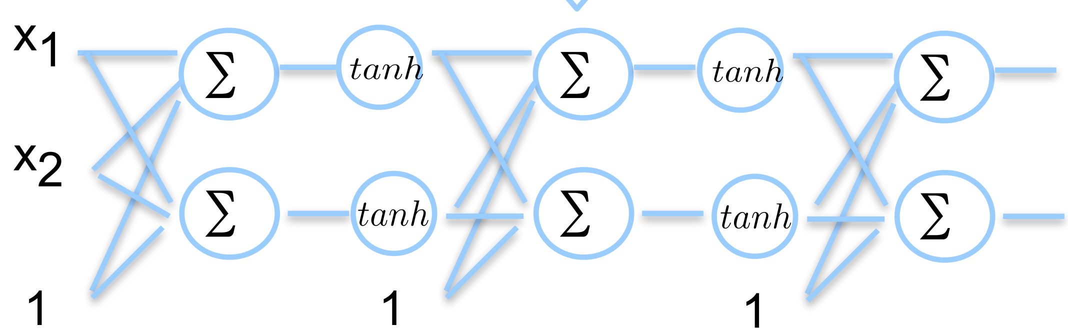
$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} \tanh(x^T a_1) \\ \tanh(x^T a_2) \end{bmatrix}$$

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



Two Neurons
Two neural elements

Deep Neural Networks



Learning two nonlinear relations

Minimize

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

a 3x2 matrix

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

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

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

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

Learning two nonlinear relations

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

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

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

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

$$y = \tanh(\mathbf{x}^T\mathbf{a})$$

$$\frac{dy}{d\mathbf{a}} = \frac{d\tanh(\mathbf{x}^T\mathbf{a})}{d\mathbf{a}}$$

$$= (1 - \tanh^2(\mathbf{x}^T\mathbf{a}))\mathbf{x}$$

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

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

```
a=rand(2,d+1);
hc=0; c=0.1;e=y-X*a';
E=100; loop=1;
while ~hc
 G=[mean(X.*(e(:,1)*ones(1,d+1))); mean(X.*(e(:,2)*ones(1,d+1)))];
 a_new=a-c*G;
 y_hat=X*a_new';
 e_new=y_hat-y;
```

error calculation

y_hat calculation

gardent calculation

Learning two linear relations

Learning two non-linear relations

```
a=rand(2,d+1);
hc=0; c=0.2;e=y-tanh(X*a');
E=100; loop=1;
while ~hc
 dy=1-tanh(X*a').^2;
 G=[mean(X.*(dy(:,1)*ones(1,d+1)).*(e(:,1)*ones(1,d+1)))];
 G=[G; mean(X.*(dy(:,2)*ones(1,d+1)).*(e(:,2)*ones(1,d+1)))];
 a_new=a-c*G;
 y_hat=tanh(X*a_new');
```

Learning two nonlinear relations

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

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

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

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

$$y = \tanh(\mathbf{x}^T\mathbf{a})$$

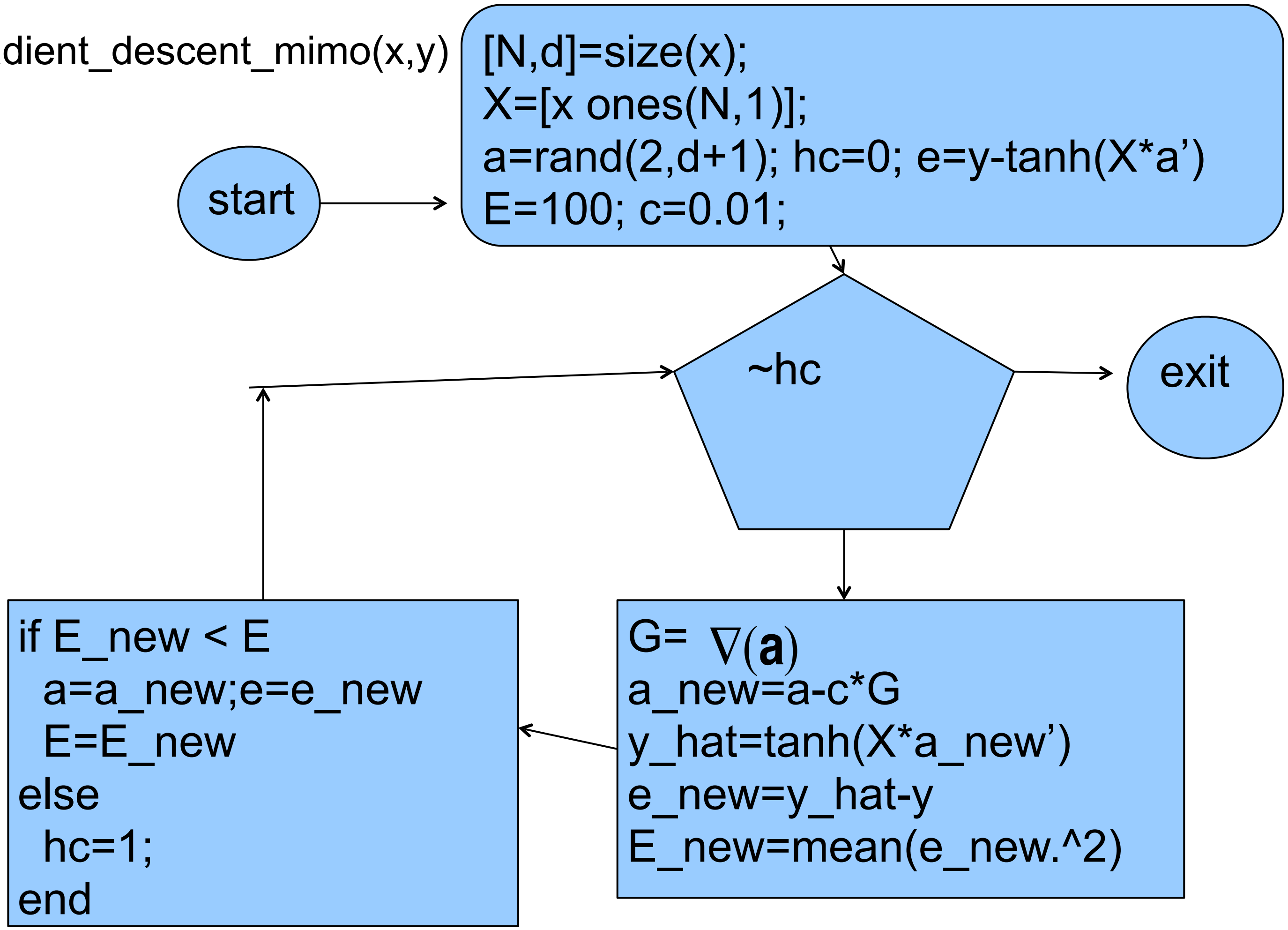
$$\frac{dy}{d\mathbf{a}} = \frac{d\tanh(\mathbf{x}^T\mathbf{a})}{d\mathbf{a}}$$

$$= (1 - \tanh^2(\mathbf{x}^T\mathbf{a}))\mathbf{x}$$

```
dy=1-tanh(X*a').^2;
G=[mean(X.*(dy(:,1)*ones(1,d+1)).*(e(:,1)*ones(1,d+1)))];
G=[G; mean(X.*(dy(:,2)*ones(1,d+1)).*(e(:,2)*ones(1,d+1)))];
```

Learning multiple nonlinear relations

function a=gradient_descent_mimo(x,y)



Deep learning

How to learn a deep neural network ?