MLP(multilayer perceptrons) learning by the Levenberg-Marquardt method

Part II

M perceptrons

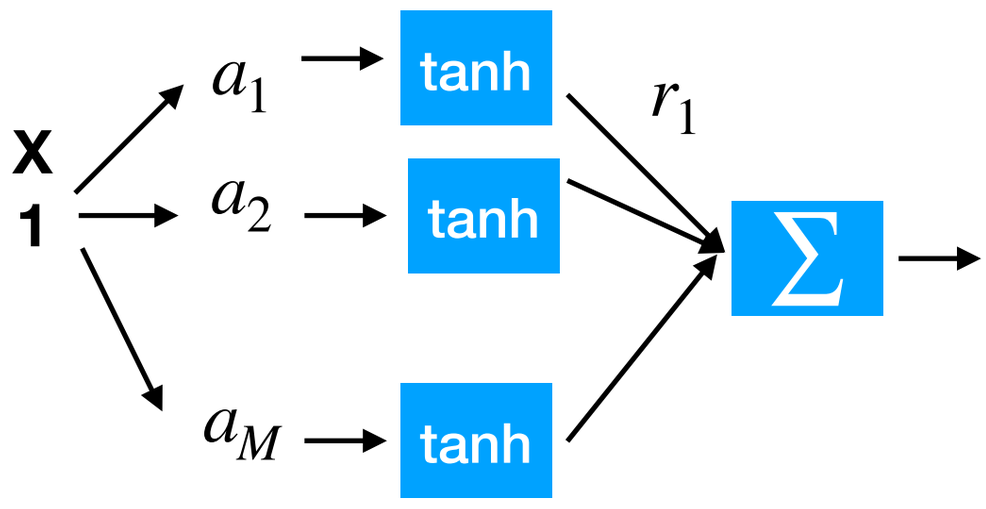
Generalized from two perceptrons

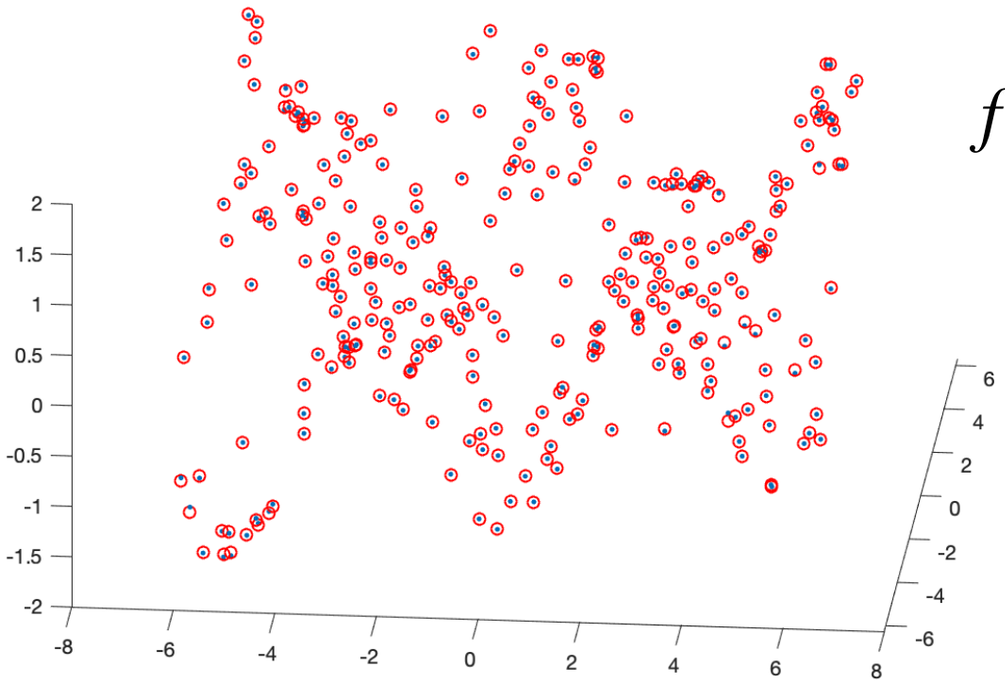
- How to revise mlp_learning for the case of learning a network of M perceptrons?

- How to add adaptable posterior weights?

Smooth function

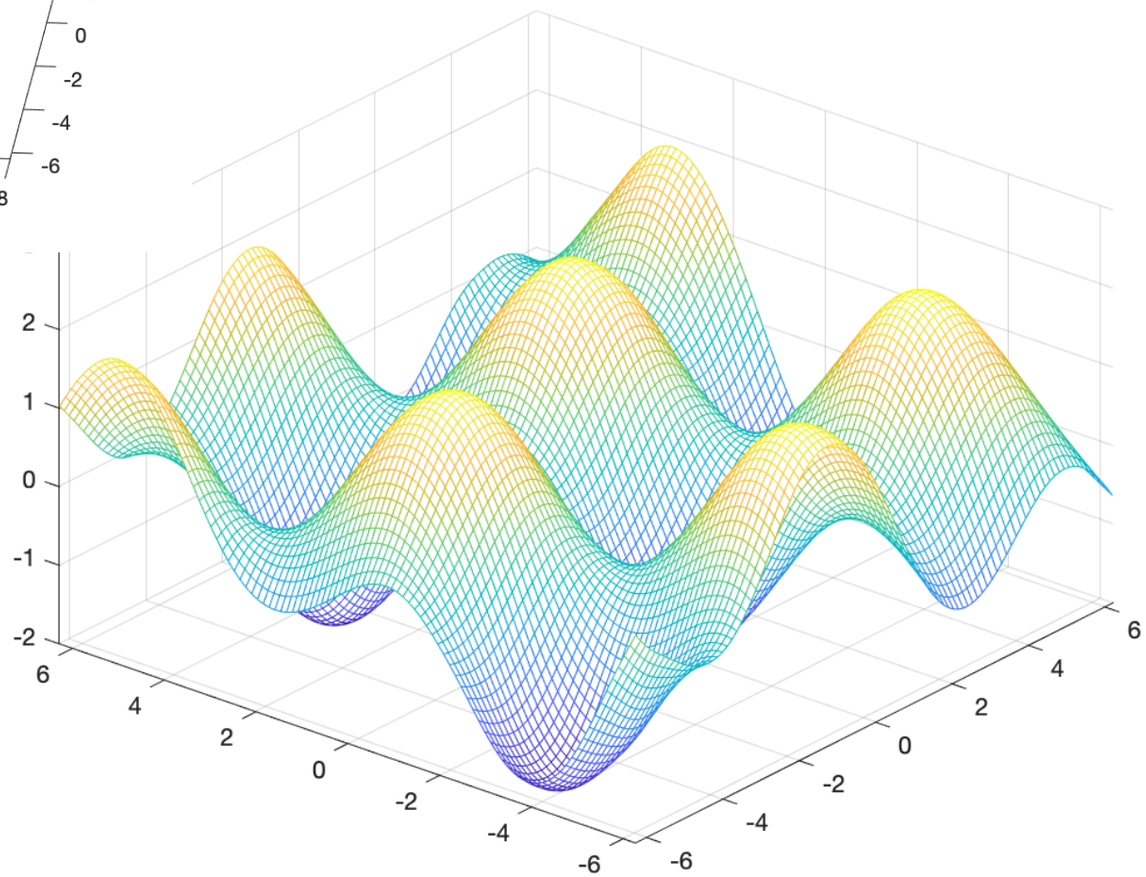
- How to approximate a target function, sin(x1+x2) +cos(x1+x2)?





$$f(x; a, b, r) = \sum_{m=1}^{M} r_m tanh(a_m^T x + b_m) + r_0,$$

M >> 2



Mathematical modeling for nonlinear transformation, M = 10

$$y = c_{31}tanh(c_1x_1 + c_2x_2 + c_3) + c_{32}tanh(c_4x_1 + c_5x_2 + c_6)$$

$$+ \ldots + c_{39}tanh(c_{25}x_1 + c_{26}x_2 + c_{27}) + c_{40}tanh(c_{28}x_1 + c_{29}x_2 + c_{30}) + c_{41}$$

Nonlinear transformation of linear combination of input attributes

$$f(x; a, b, r) = \sum_{m=1}^{M} r_m tanh(a_m^T x + b_m) + r_0,$$

- $c_{3(m-1)+1:3m} = [a_m, b_m]$, m =1,…,M
- $c_{3M+1:4M} = [r_1, \ldots, r_M]$
- $c_{4M+1} = r_0$

Linear transformation
M = 10

$$y = c_{31}tanh(c_1x_1 + c_2x_2 + c_3) + c_{32}tanh(c_4x_1 + c_5x_2 + c_6)$$

$$+ \ldots + c_{39}tanh(c_{25}x_1 + c_{26}x_2 + c_{27}) + c_{40}tanh(c_{28}x_1 + c_{29}x_2 + c_{30}) + c_{41}$$

Elements $c_1, \ldots, c_{3M}$ in c are weights for linear transformation

4*10 + 1 = 41

M = 10

$$y = c_{31}tanh(c_1x_1 + c_2x_2 + c_3) + c_{32}tanh(c_4x_1 + c_5x_2 + c_6)$$

$$+ \ldots + c_{39}tanh(c_{25}x_1 + c_{26}x_2 + c_{27}) + c_{40}tanh(c_{28}x_1 + c_{29}x_2 + c_{30}) + c_{41}$$

Elements $c_{3M+1}, \ldots, c_{3M+M}$ in c are posterior weights

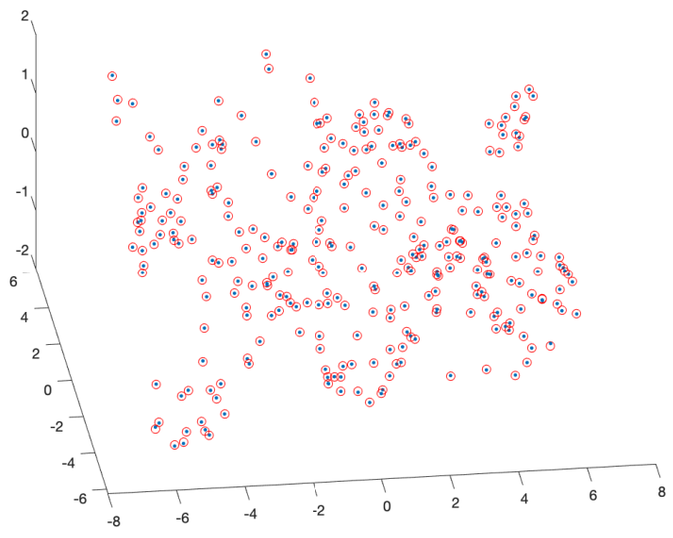
c_{4M+1} is a constant term

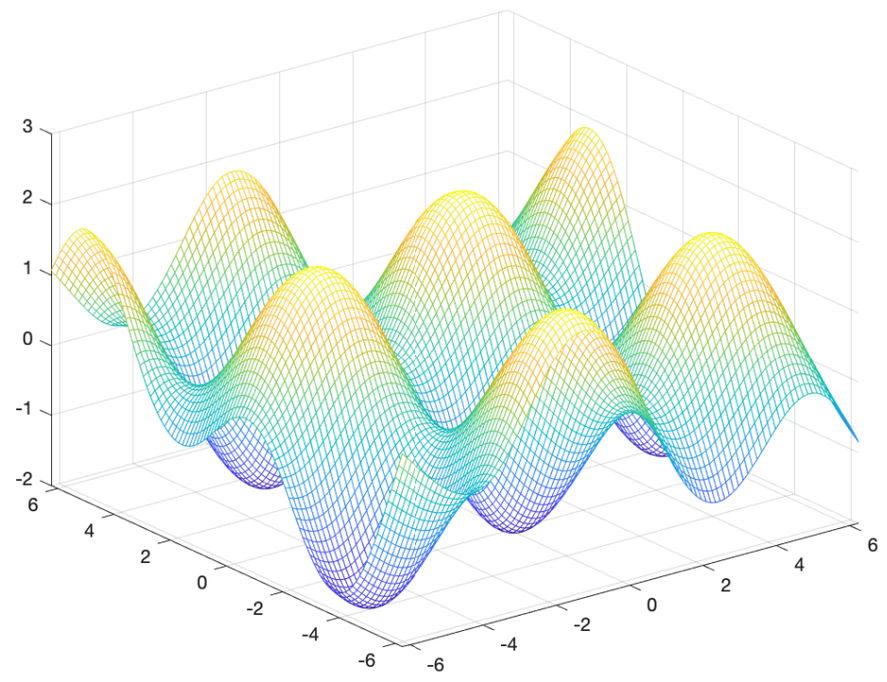
/ ▸ Users ▸ apple ▸ Desktop ▸ Jiann-Ming Wu ▸ 2023-I NA數值分析 ▸ codes ▸

Editor - /Users/apple/Desktop/Jiann-Ming Wu/2023-I NA數值分析/codes/demo_mlp_learning_M_units.m

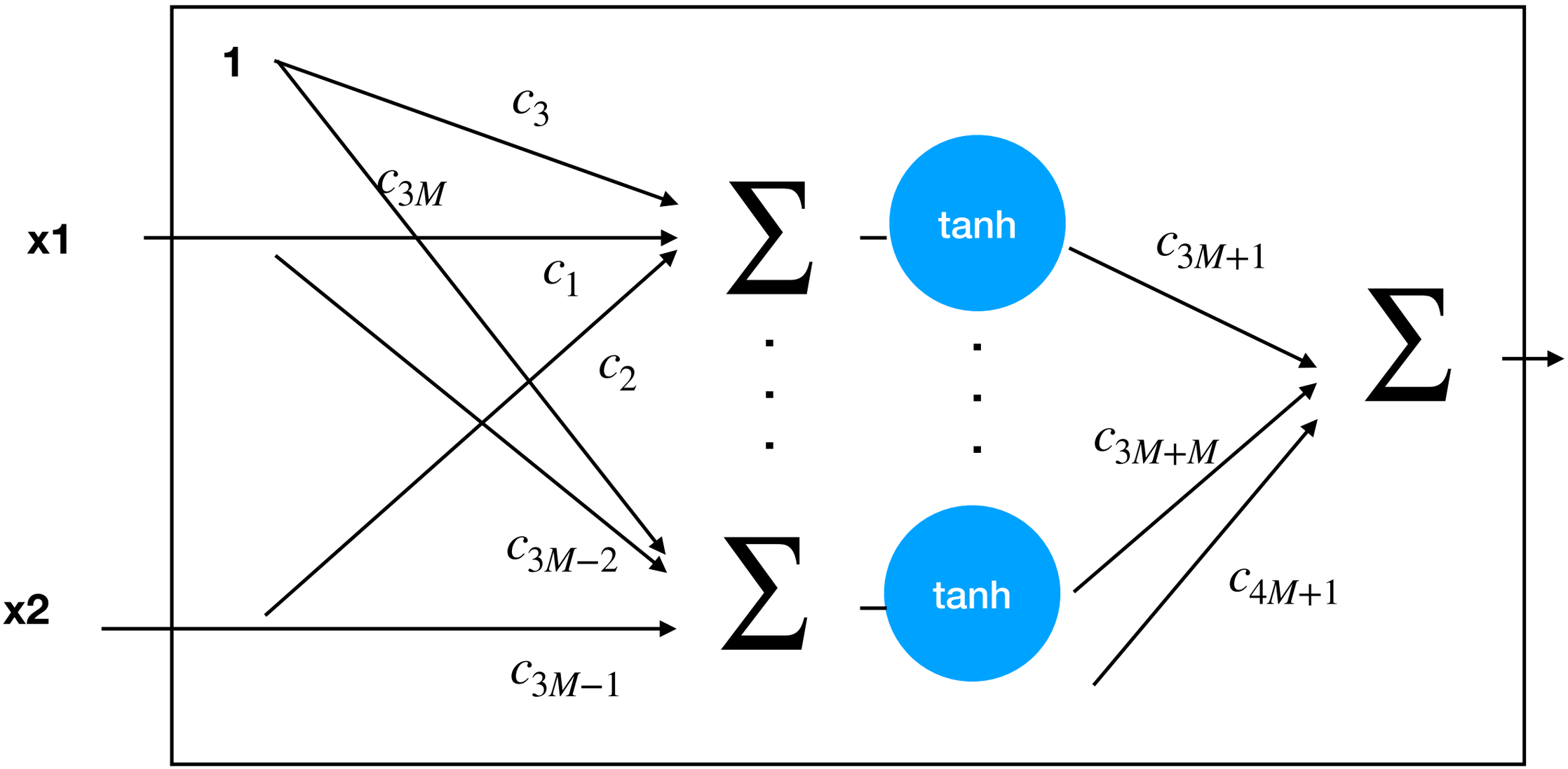
+5 | demo_mlp_learning.m | generateTraningData2.m | demo_Newton_mlp_learning.m | demo_mlp_learning_M_units.m | +

```
1   %% Solving a nonlinear system for MLP learning, created by Jiani
2   %    Department of Applied Mathematics, National Dong Hwa Univers
3   %
4   %  Using Matlab (R)
5   %  2021 11 3
6   %  MLP learning is resolved by the Levenberg-Marquardt method
7   %
8   function demo_mlp_learning_M_units()
9       syms c   % adaptable parameters in an MLP network
10      % preparation of training data
11      % uniform sampling
12      % Substitute to the target function g
```



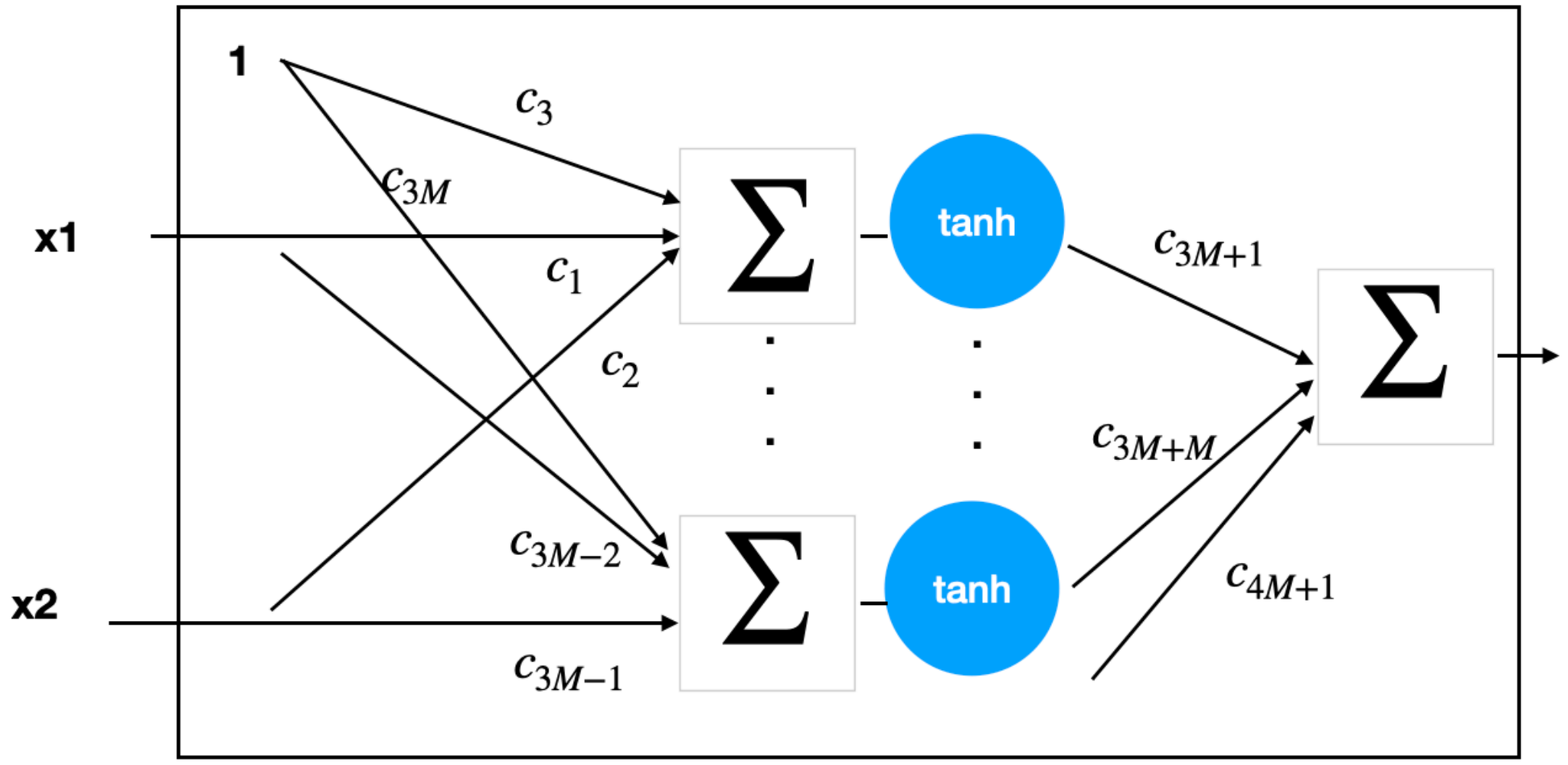


Neural Architecture: Perceptrons 認知機



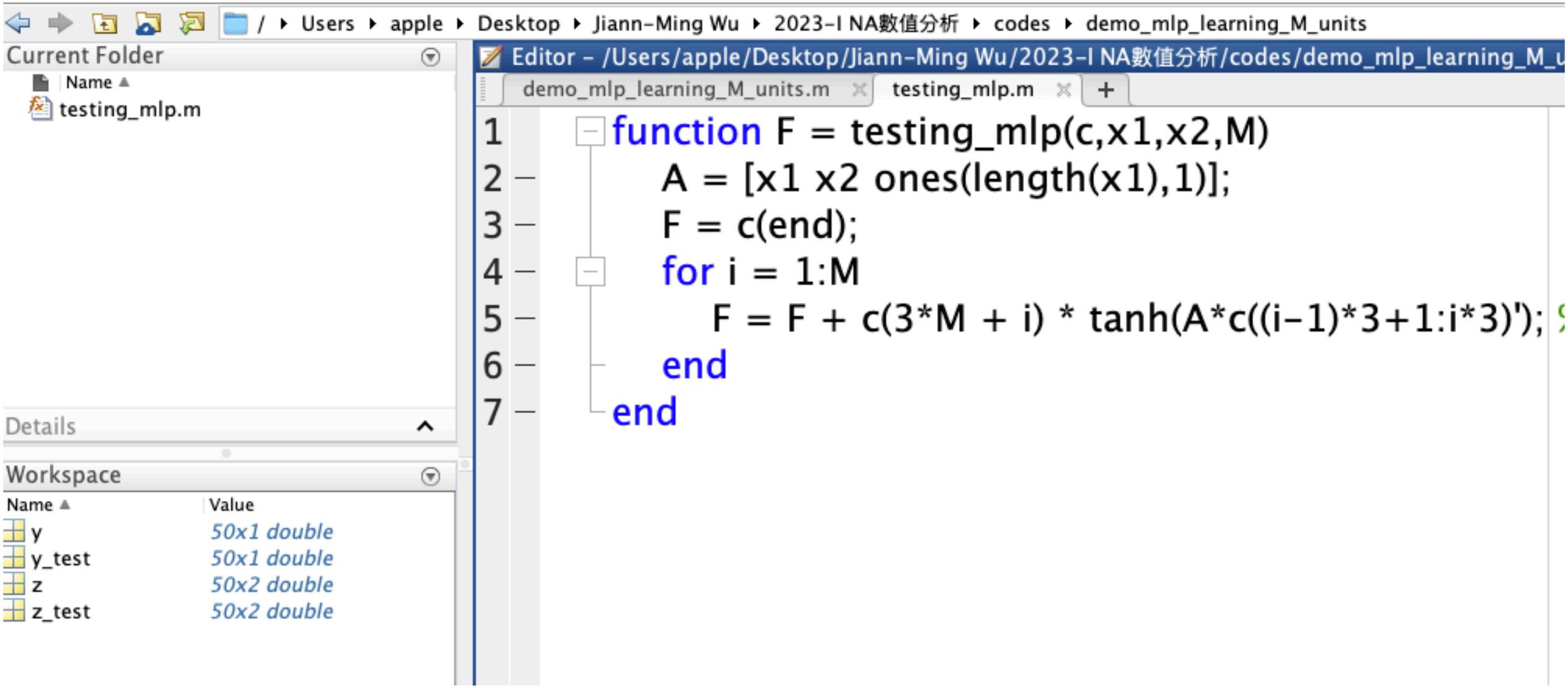
h = g_hat(x1,x2,c)

g_hat is a parametric function
c: adaptable parameters




```
function F = testing_mlp(c,x1,x2,M)
    A = [x1 x2 ones(length(x1),1)];
    F = c(end);
    for i = 1:M
        F = F + c(3*M + i) * tanh(A*c((i-1)*3+1:i*3)');
    end
end
```

- $c_{3(m-1)+1:3m} = [a_m, b_m]$, m =1,…,M
- $c_{3M+1:4M} = [r_1, \ldots, r_M]$
- $c_{4M+1} = r_0$


/ ▸ Users ▸ apple ▸ Desktop ▸ Jiann-Ming Wu ▸ 2023-I NA數值分析 ▸ codes ▸ demo_mlp_learning_M_units
Current Folder
Name
testing_mlp.m
Details
Workspace
Name
Value
y
50x1 double
y_test
50x1 double
z
50x2 double
z_test
50x2 double
Editor - /Users/apple/Desktop/Jiann-Ming Wu/2023-I NA數值分析/codes/demo_mlp_learning_M_u
demo_mlp_learning_M_units.m
testing_mlp.m
1 function F = testing_mlp(c,x1,x2,M)
2 A = [x1 x2 ones(length(x1),1)];
3 F = c(end);
4 for i = 1:M
5 F = F + c(3*M + i) * tanh(A*c((i-1)*3+1:i*3)');
6 end
7 end

$$f(x; a, b, r) = \sum_{m=1}^{M} r_m tanh(a_m^T x + b_m) + r_0,$$

- $c_{3(m-1)+1:3m} = [a_m, b_m]$, m =1,…,M

- $c_{3M+1:4M} = [r_1, \ldots, r_M]$

- $c_{4M+1} = r_0$

Target function I

```
function h = g(x1,x2)
      C1=[1 1/2 -1/2]'; %weight
      C2=[1/3 -1 1]'; %weight
      A = [x1 x2 ones(length(x1),1)];
      h = tanh(A*C1)+tanh(A*C2); %activation function
end
```

- High dimension and nonlinear target function

Target function II

Replace the first tanh with sin and the second tanh with cos

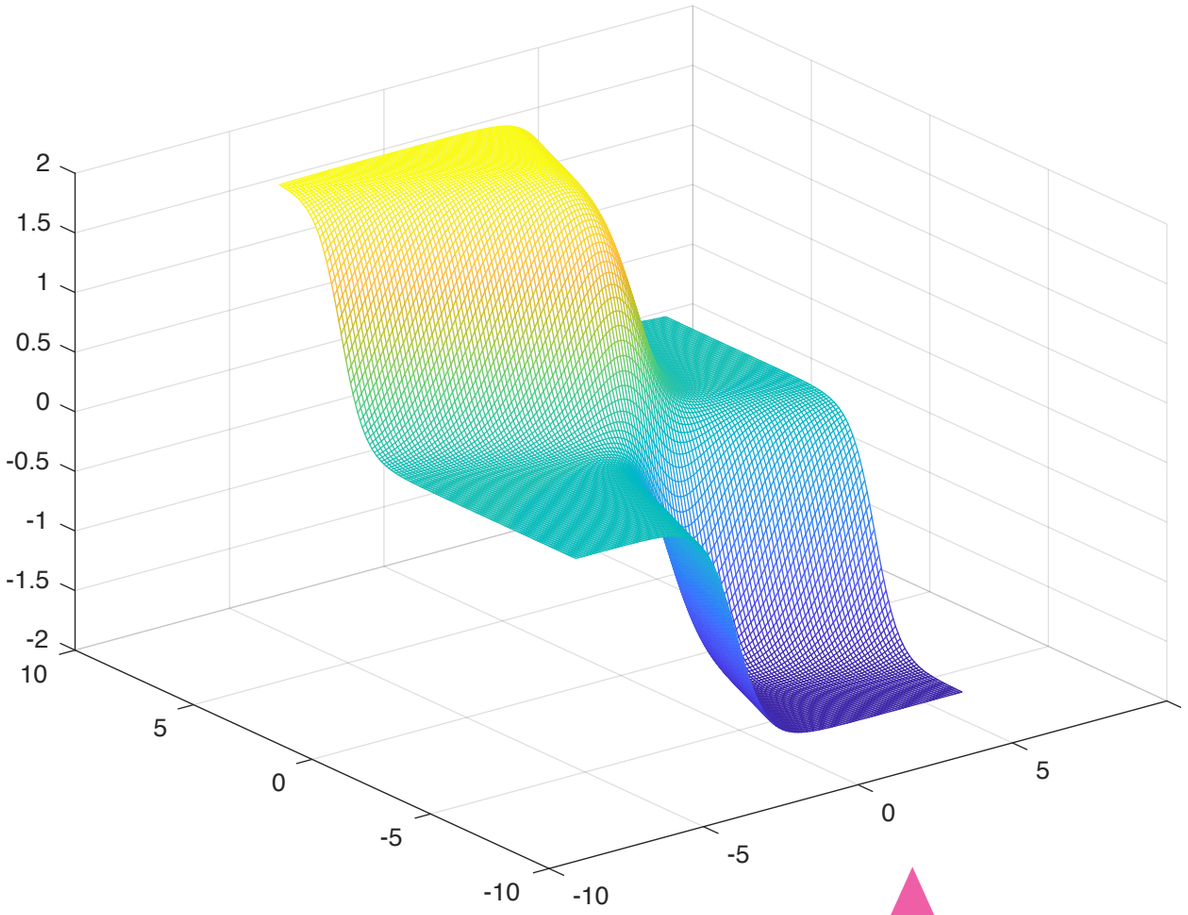
```
function h = g2(x1,x2)
      C1=[1 1/2 -1/2]'; %weight
      C2=[1/3 -1 1]'; %weight
      A = [x1 x2 ones(length(x1),1)];
      h = sin(A*C1)+cos(A*C2); %activation function
end
```

- High dimension and nonlinear target function

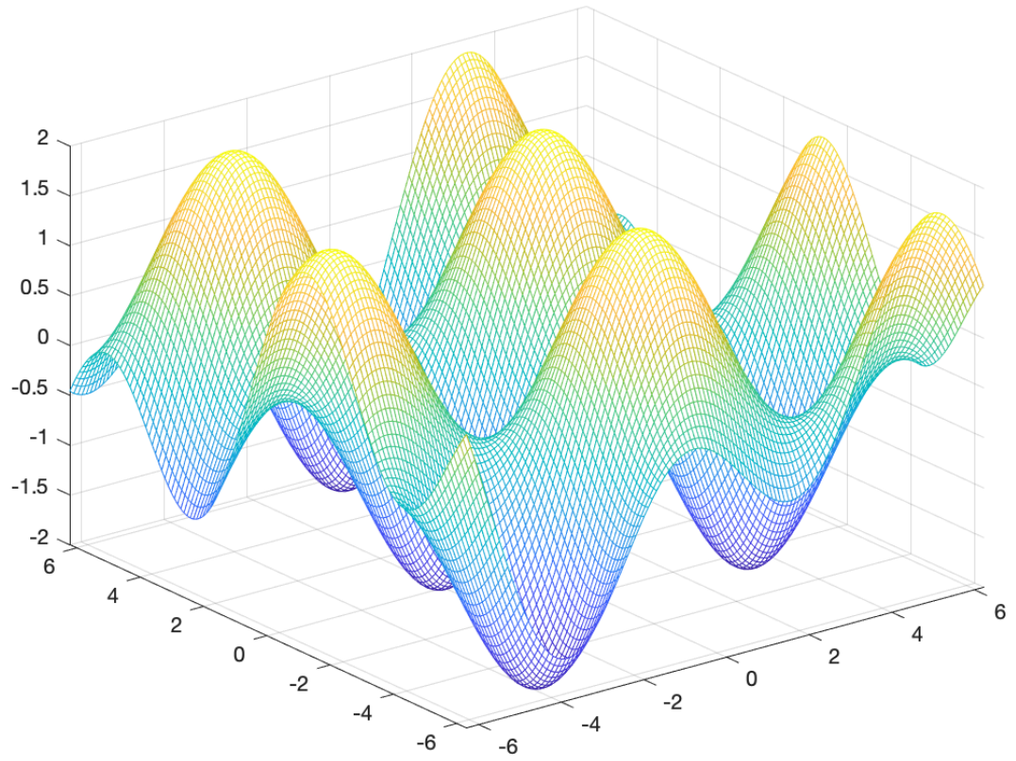
demo_plot2d

```
function demo_plot2d()

    range=2*pi;
    x1=-range:0.1:range;
    x2=x1;
    for i=1:length(x1)
        C(i,:)=g(x1(i)*ones(length(x2),1),x2');
    end
    mesh(x1,x2,C);
end

function h = g(x1,x2)
    C1=[1 1/2 -1/2]'; %weight
    C2=[1/3 -1 1]'; %weight
    A = [x1 x2 ones(length(x1),1)];
    h = tanh(A*C1)+tanh(A*C2); %activation function
end
```

```
function h = g(x1,x2)
    C1=[1 1/2 -1/2]'; %weight
    C2=[1/3 -1 1]'; %weight
    A = [x1 x2 ones(length(x1),1)];
    h = tanh(A*C1)+tanh(A*C2); %activation function
end
```



Target function I



Target function II

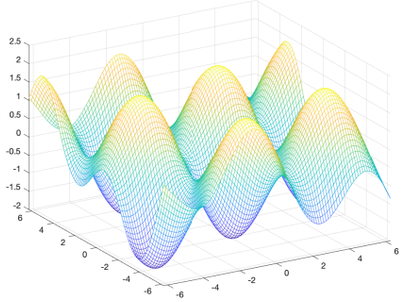
Write an approximating function with M perceptrons

- A multilayer neural network performs a parametric function

```
function h = g_hat(x1,x2,c)
      A = [x1 x2 ones(length(x1),1)];
      .
      .
end
```

Adaptable parameters

- Approximating function g by g_hat
- The problem is how to estimate adaptable parameters in c.



/ ▸ Users ▸ apple ▸ Desktop ▸ Jiann-Ming Wu ▸ 2023-I NA數值分析 ▸ codes ▸ demo_mlp_learning_M_units

Current Folder

Name ▲
demo_plot2d...
testing_mlp.m

Details

Workspace

Name ▲
y
y_test
z
z_test

Editor - /Users/apple/Desktop/Jiann-Ming Wu/2023-I NA數值分析/codes/demo_mlp_learning_M_units/demo

demo_mlp_learning_M_units.m | testing_mlp.m | demo_plot2d.m*

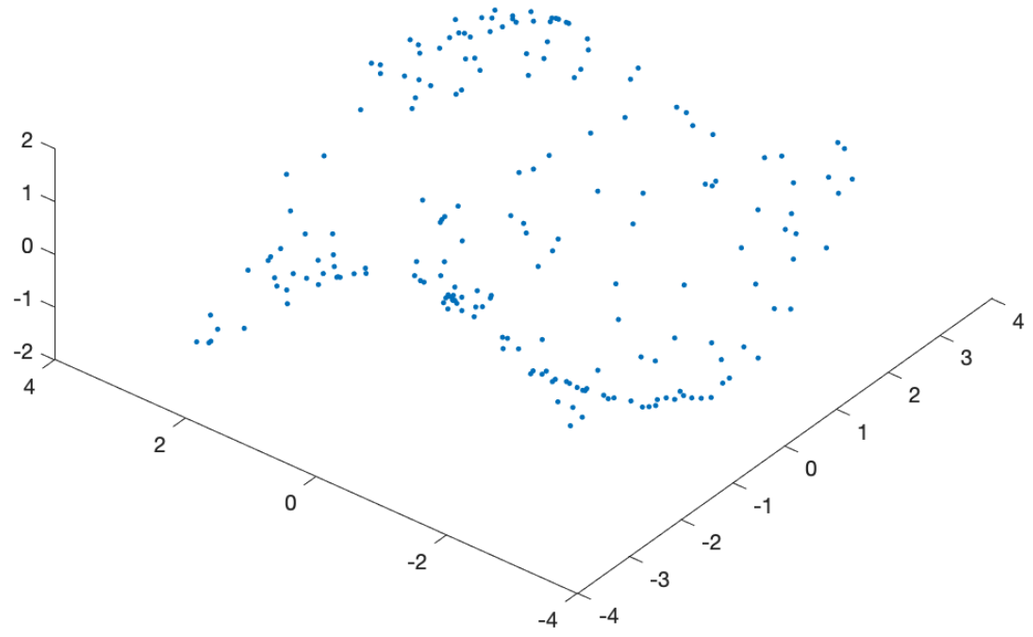
```
1   function demo_plot2d()
2       range=2*pi;
3       x1=-range:0.1:range;
4       x2=x1;
5       for i=1:length(x1)
6           C(i,:)=g(x1(i)*ones(length(x2),1),x2');
7       end
8       mesh(x1,x2,C);
9   end
10  function h = g(x1,x2)
11      C1=[1 1/2 -1/2]'; %weight
12      C2=[1/3 -1 1]'; %weight
13      A = [x1 x2 ones(length(x1),1)];
14      h = sin(A*C1)+cos(A*C2); %activation function
15  end
```

Data Driven Learning and Testing

Training and Testing Data Sets

training data set

```
z=rand(50,2)*2*pi-pi;
y=g2(z(:,1),z(:,2));
plot3(z(:,1),z(:,2),y,'.');
```



- Two steps
  - Generate a uniform sample from the domain
  - Substitute z(:,1) and z(:,2) to g

/ ▸ Users ▸ apple ▸ Desktop ▸ Jiann-Ming Wu ▸ 2023-I NA數值分析 ▸ codes ▸ demo_mlp_learning_M_units

Current Folder

Name ▲
- demo_plot2d.m
- g.m
- generate_data.m
- testing_mlp.m

g.m (Function)

Editor - /Users/apple/Desktop/Jiann-Ming Wu/2023-I NA數值分析/codes/demo_mlp_learning_M

demo_mlp_learning_M_units.m | testing_mlp.m | demo_plot2d.m | generate_data.m

```
1  function h = g(x1,x2)
2      C1=[1 1/2 -1/2]'; %weight
3      C2=[1/3 -1 1]'; %weight
4      A = [x1 x2 ones(length(x1),1)];
5      h = sin(A*C1)+cos(A*C2); %activation function
6  end
```

3

/ ▸ Users ▸ apple ▸ Desktop ▸ Jiann-Ming Wu ▸ 2023-I NA數值分析 ▸ codes ▸ demo_mlp_learning_M_u

Current Folder

Name ▲
- demo_plot2d.m
- g.m
- generate_data.m
- testing_mlp.m

generate_data....

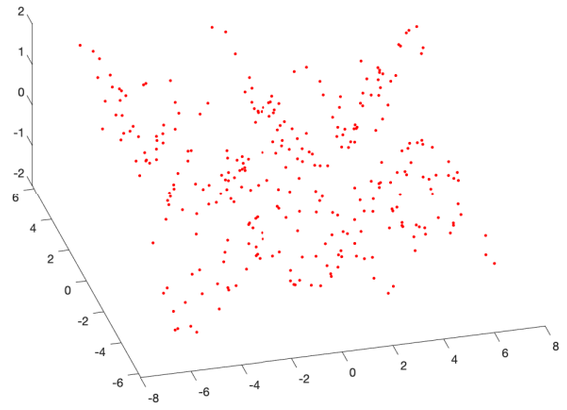
Workspace

Name ▲	Value
y	50x1

Editor - /Users/apple/Desktop/Jiann-Ming Wu/2023-I NA數值分析/codes/demo_mlp_learning_

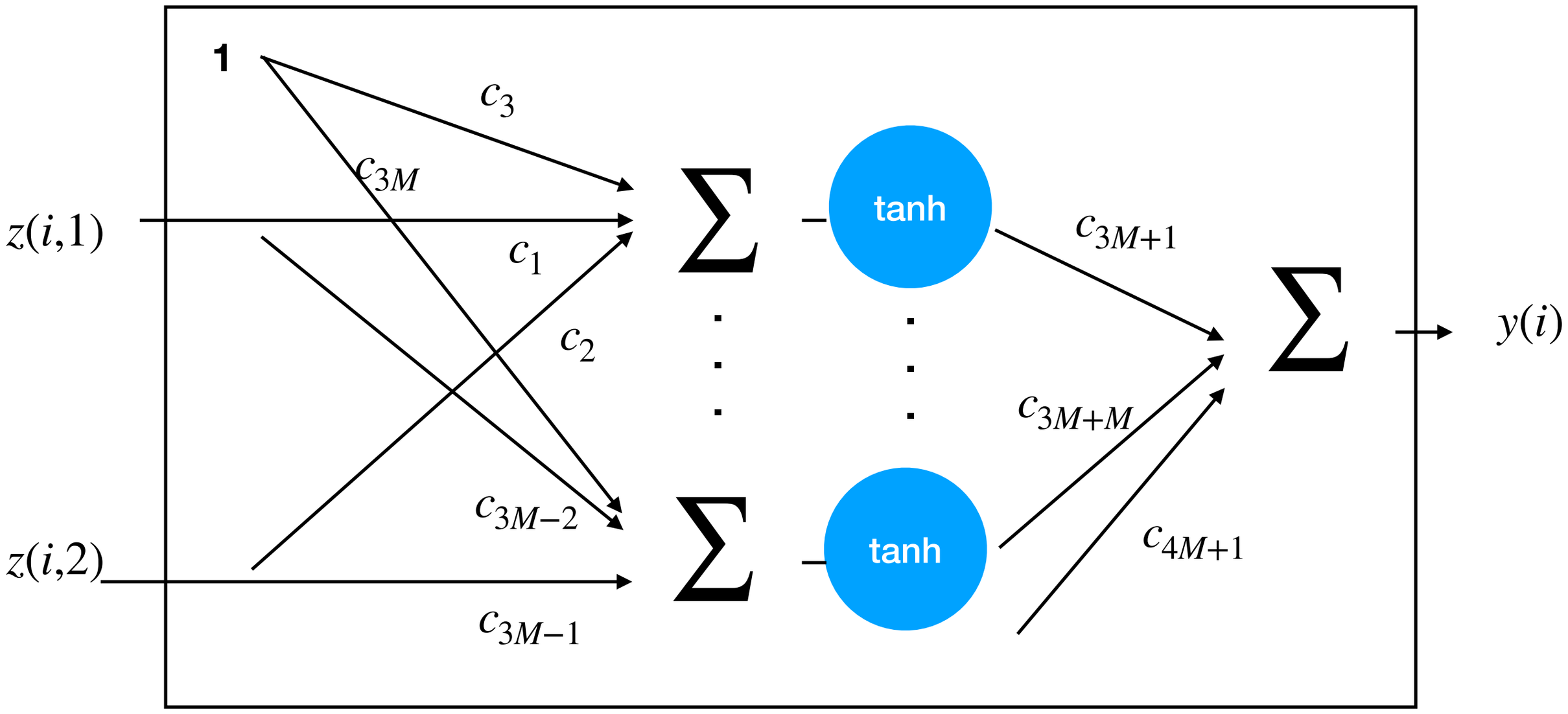
demo_mlp_learning_M_units.m | testing_mlp.m | demo_plot2d.m | generate_data.m

```
1  z=rand(300,2)*4*pi-2*pi;
2  y=g(z(:,1),z(:,2));
3  plot3(z(:,1),z(:,2),y,'.');
4
5  z_test=rand(300,2)*4*pi-2*pi;
6  y_test=g(z_test(:,1),z_test(:,2));
7  figure
8  plot3(z_test(:,1),z_test(:,2),y_test,'r.')
```



$z_i = (z(i,1), z(i,2))^T$ denotes an input

y_i denotes the correspondent output



Approximating a target function by solving a nonlinear system

$z(i,1) \longrightarrow$ $y = c_{31}tanh(c_1x_1 + c_2x_2 + c_3) + c_{32}tanh(c_4x_1 + c_5x_2 + c_6)$ $\longrightarrow y(i)$

$z(i,2) \longrightarrow$ $+ \ldots + c_{39}tanh(c_{25}x_1 + c_{26}x_2 + c_{27}) + c_{40}tanh(c_{28}x_1 + c_{29}x_2 + c_{30}) + c_{41}$

A Nonlinear Equation

A Nonlinear transformation

$$f(z_1[i], z_2[i]) = c_{31}tanh(c_1z_1[i] + c_2z_2[i] + c_3) + c_{32}tanh(c_4z_1[i] + c_5z_2[i] + c_6)$$

$$+ \ldots + c_{39}tanh(c_{25}z_1[i] + c_{26}z_2[i] + c_{27}) + c_{40}tanh(c_{28}z_1[i] + c_{29}z_2[i] + c_{30}) + c_{41} = y(i)$$

N = 200
Nonlinear Equations

$$f(z_1[1], z_2[1]) = c_{31}tanh(c_1z_1[1] + c_2z_2[1] + c_3) + c_{32}tanh(c_4z_1[1] + c_5z_2[1] + c_6)$$

$$+ \ldots + c_{39}tanh(c_{25}z_1[1] + c_{26}z_2[1] + c_{27}) + c_{40}tanh(c_{28}z_1[1] + c_{29}z_2[1] + c_{30}) + c_{41} = y(1)$$

$$\vdots$$

$$f(z_1[i], z_2[i]) = c_{31}tanh(c_1z_1[i] + c_2z_2[i] + c_3) + c_{32}tanh(c_4z_1[i] + c_5z_2[i] + c_6)$$

$$+ \ldots + c_{39}tanh(c_{25}z_1[i] + c_{26}z_2[i] + c_{27}) + c_{40}tanh(c_{28}z_1[i] + c_{29}z_2[i] + c_{30}) + c_{41} = y(i)$$

$$\vdots$$

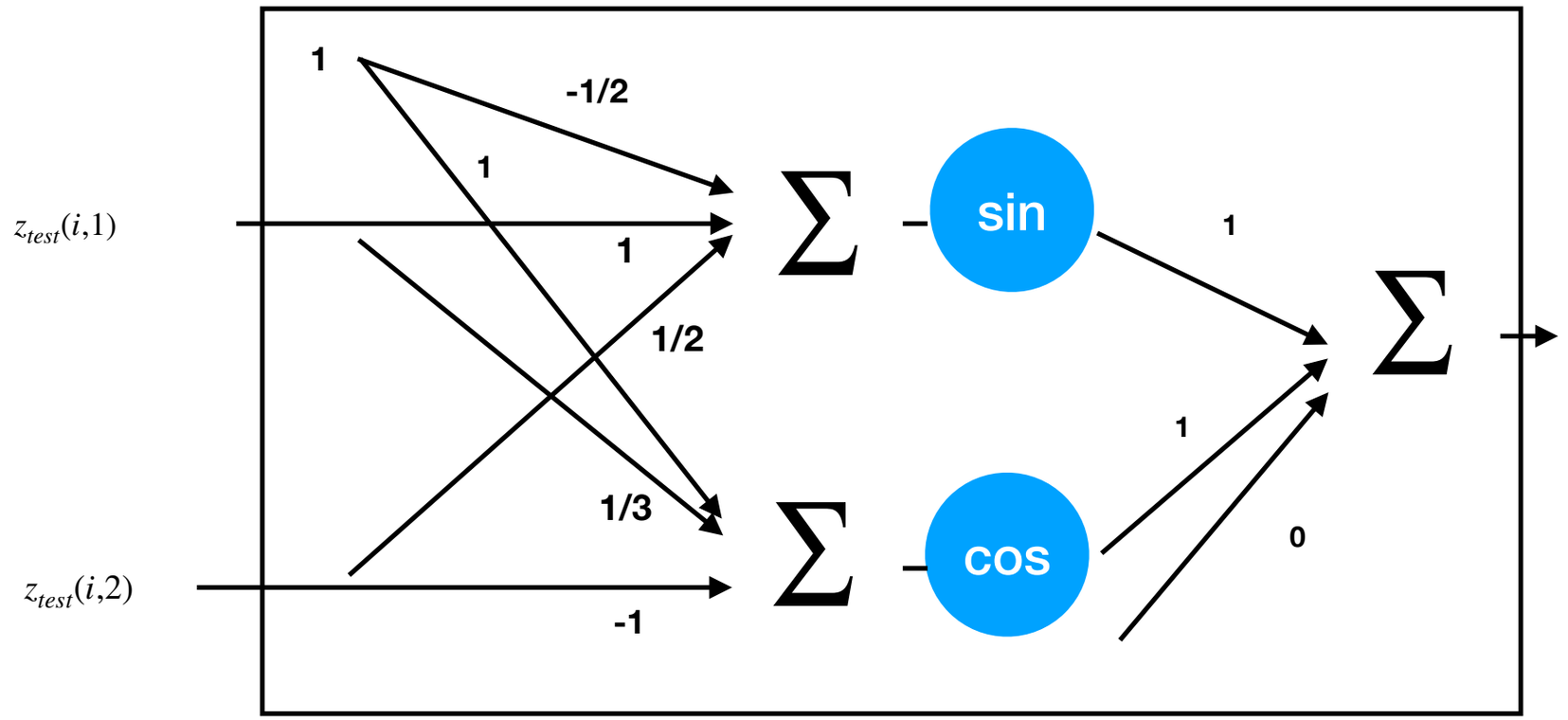
$$f(z_1[N], z_2[N]) = c_{31}tanh(c_1z_1[N] + c_2z_2[N] + c_3) + c_{32}tanh(c_4z_1[N] + c_5z_2[N] + c_6)$$

$$+ \ldots + c_{39}tanh(c_{25}z_1[N] + c_{26}z_2[N] + c_{27}) + c_{40}tanh(c_{28}z_1[N] + c_{29}z_2[N] + c_{30}) + c_{41} = y(N)$$

testing data

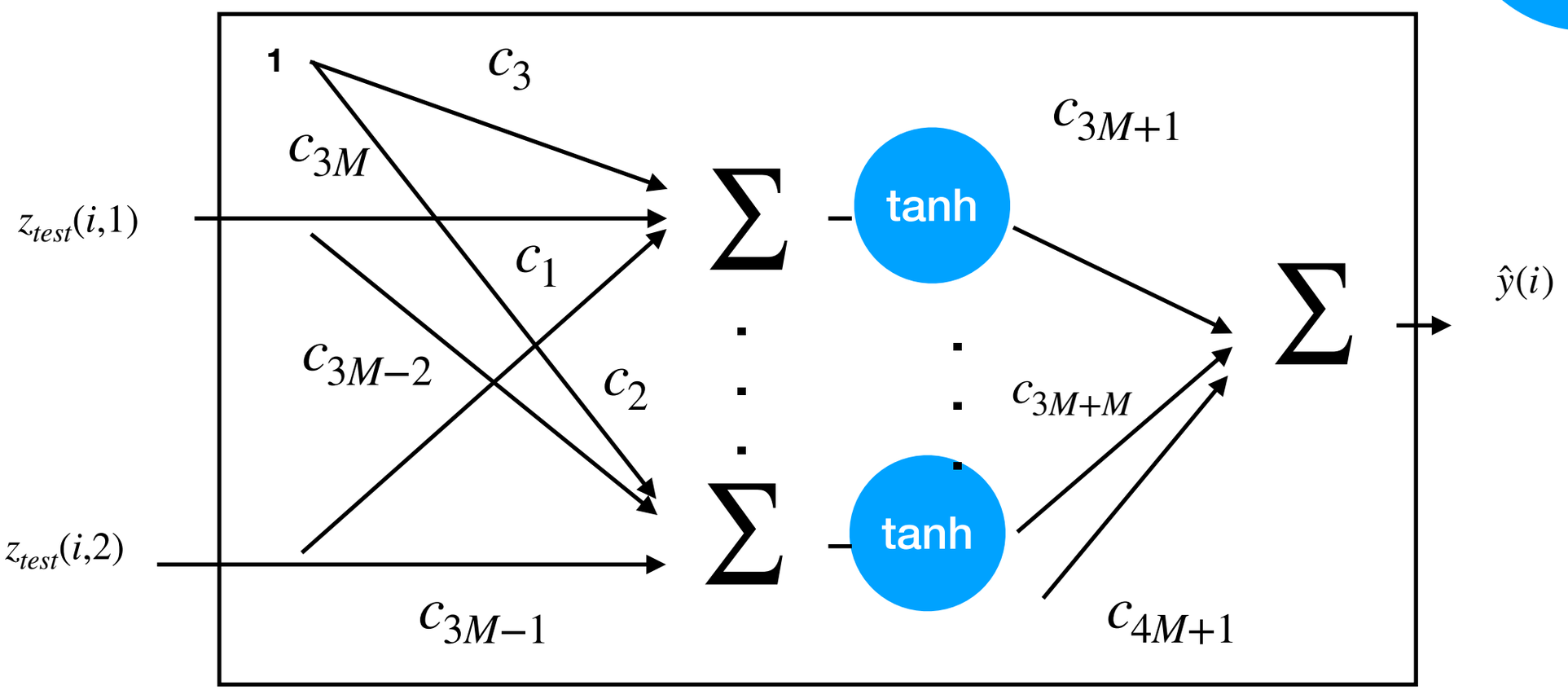
```
z_test=rand(50,2)*2*pi-pi;
y_test=g2(z_test(:,1),z_test(:,2));
```

- Two steps
  - Generate another uniform sample from the domain
  - Substitute z_test(:,1) and z_test(:,2) to g to generate y_test
  - Substitute z_test(:,1) and z_test(:,2) to g_hat to generate y_hat for approximating y_test



Error

$y(i) - \hat{y}(i)$



goal of learning

Minimizing not only the training mean square error but also the testing mean square error

MLP(multilayer perceptrons) learning and testing

1. MLP learning

   - Train testing_mlp(x1, x2,c) to optimize c subject to data z, y

2. Let c_hat denote the learning result

3. Test testing_mlp(x1,x2,c_hat) by z_test and y_test

/ ▸ Users ▸ apple ▸ Desktop ▸ Jiann-Ming Wu ▸ 2023-I NA數值分析 ▸ codes ▸ demo_mlp_learning_M_units

Current Folder

Name ▲

- demo_mlp_learning_M...
- demo_plot2d.m
- g.m
- generate_data.m
- learning_mlp.m
- testing_mlp.m

generate_data.m (S...

Workspace

Name ▲	Value
y	300x1 do
y_test	300x1 do

Editor - /Users/apple/Desktop/Jiann-Ming Wu/2023-I NA數值分析/codes/demo_mlp_learning_M_units/lea

+2 | demo_plot2d.m | generate_data.m | g.m | demo_mlp_learning_M_units.m | learning_mlp.m

```
1  function F = learning_mlp(c,x1,x2,y,M)
2      A = [x1 x2 ones(length(x1),1)];
3      F = c(end) - y;
4      for i = 1:M
5          F = F + c(3*M + i) * tanh(A*c((i-1)*3+1:i*3)'); %acti
6      end
7  end
```

A nonlinear system

```
function F = learning_mlp(c,x1,x2,y)
     A = [x1 x2 ones(length(x1),1)];
    .
    .
    .
end
```

- Create a nonlinear function
  - F = learning_mlp(c,x1,x2,y)

```
function F = learning_mlp(c,x1,x2,y)
    A = [x1 x2 ones(length(x1),1)];
    .
    .
    .
end
```

- function learning_mlp is a nonlinear system
- adaptable parameters c are unknown
- Set x1 to z(:,1), x2 to z(:,2) and y
  - Let c_zero denote a zero where F is close to zero
  - The output of g_hat(x1, x2,c_zero) well approximates target y

5a

/ ▸ Users ▸ apple ▸ Desktop ▸ Jiann-Ming Wu ▸ 2023-I NA數值分析 ▸ codes ▸ demo_mlp_learning_M_units

Editor - /Users/apple/Desktop/Jiann-Ming Wu/2023-I NA數值分析/codes/demo_mlp_learning_M_units/demo_mlp_learning_...

+2 | testing_mlp.m | demo_plot2d.m | generate_data.m | g.m | demo_mlp_learning_M_units.m | +

Current ...

Name

demo_ml...
demo_plo...
g.m
generate_...
learning_...
testing_...

generat...

Workspace

Name

y
y_test
z
z_test

```
1  function demo_mlp_learning_M_units()
2      syms c
3      z=rand(300,2)*4*pi-2*pi;
4      y=g(z(:,1),z(:,2));
5      plot3(z(:,1),z(:,2),y,'.');
6      M = 40;
7      f = @(c)learning_mlp(c,z(:,1),z(:,2),y,M);
8      c0=rand(1,M*3+M+1)*2-1;
9      options = optimoptions('fsolve','Algorithm', 'levenberg-marquardt')
10     c_zero = fsolve(f,c0,options);
11     mae = mean(abs(learning_mlp(c_zero,z(:,1),z(:,2),y,M)));
12     fprintf('** mean absolute error for ap func g: %f\n',mae);
```

5b

```
c0=rand(1,M*3+M+1)*2-1;
options = optimoptions('fsolve','Algorithm', 'levenberg-marquardt')
c_zero = fsolve(f,c0,options);
mae = mean(abs(learning_mlp(c_zero,z(:,1),z(:,2),y,M)));
fprintf('** training mean absolute error for ap func g: %f\n',mae);

z_test=rand(300,2)*4*pi-2*pi;
y_test=g(z_test(:,1),z_test(:,2));
mae = mean(abs(learning_mlp(c_zero,z_test(:,1),z_test(:,2),y_test,M)));
fprintf('** testing mean absolute error for ap func g: %f\n',mae);
```

Command Window

```
** training mean absolute error for ap func g: 0.009393
** testing mean absolute error for ap func g: 0.029912
```

Matlab Programming

```
%% Solving a nonlinear system for MLP learning, created by Jiann-Ming Wu
%    Department of Applied Mathematics, National Dong Hwa University
%
%  Using Matlab (R)
%  2018 dec. 3
%  MLP learning is resolved by the Levenberg-Marquardt method
%
```

```
function demo_mlp_learning()
syms c   % adaptable parameters in an MLP network

% initialization
c0= ... ;

% preparation of training data
% uniform sampling
% Substitute to the target function g
z=rand(100,2)*2*pi-pi;
y=g2(z(:,1),z(:,2));
plot3(z(:,1),z(:,2),y,'.');
```

```
% Levenberg-Marquardt method
options = optimoptions('fsolve','Algorithm', ‘levenberg-marquardt')

% specification of a nonlinear system for MLP_learning
f = @(c)learning_mlp(c,z(:,1),z(:,2),y);
```

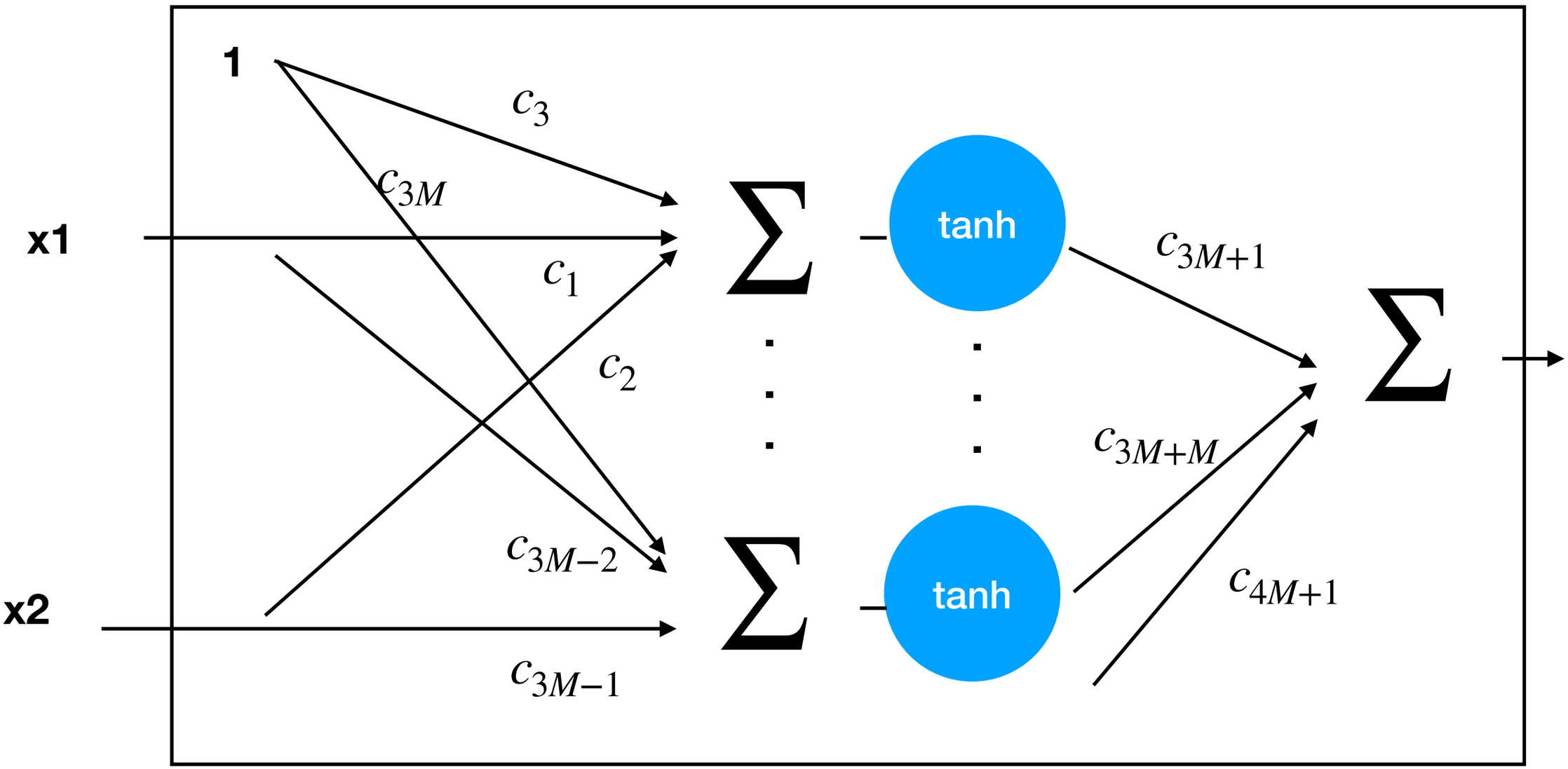
```
% apply fsolve
% Verification of c_zero
c_zero = fsolve(f,c0,options);
sum(abs(learning_mlp(c_zero,z(:,1),z(:,2),y)))
end
```

```
% a target function
function h = g(x1,x2)
    C1=[1 1/2 -1/2]'; %weight
    C2=[1/3 -1 1]'; %weight
    A = [x1 x2 ones(length(x1),1)];
    h = tanh(A*C1)+tanh(A*C2); %activation function
end
```

```
% a target function
function h = g2(x1,x2)
    C1=[1 1/2 -1/2]'; %weight
    C2=[1/3 -1 1]'; %weight
    A = [x1 x2 ones(length(x1),1)];
     .
     .
     .
end
```

```
% a nonlinear system for MLP_learning
function F = learning_mlp(c,x1,x2,y)
        .
        .
        .
end

% An approximating function
function h = g_hat(x1,x2,c)
        .
        .
        .
end
```

Neural Architecture: Perceptrons 認知機



h = g_hat(x1,x2,c)

g_hat is a parametric function
c: adaptable parameters

apple ▸ Desktop ▸ Jiann-Ming Wu ▸ 2022-I NA數值分析 ▸ codes

Editor - /Users/apple/Desktop/Jiann-Ming Wu/2022-I NA數值分析/codes/demo_mlp_learning_M_units.m

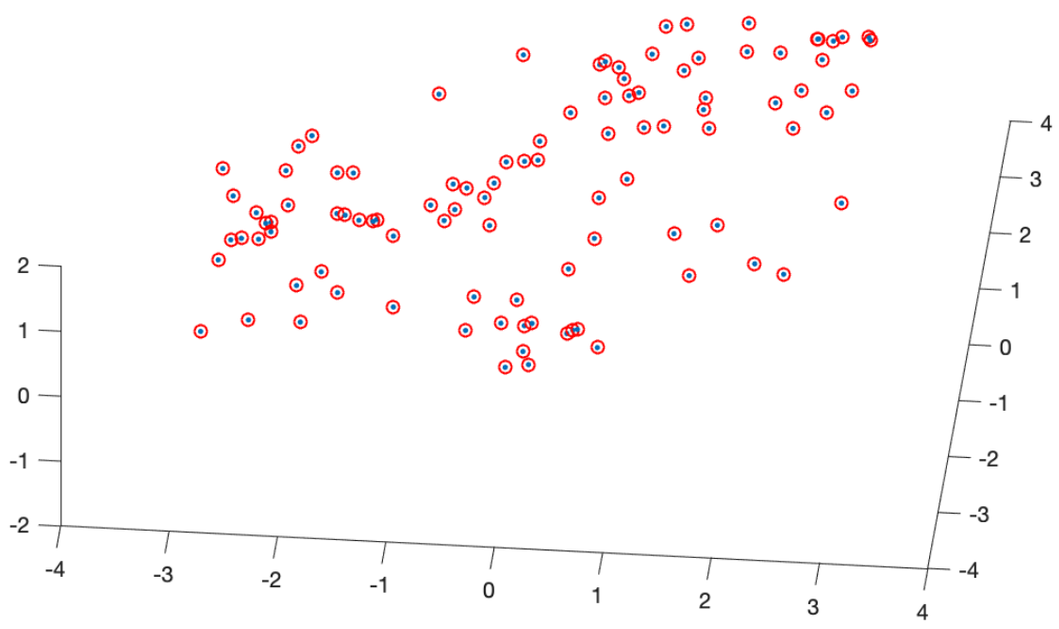
demo_mlp_learning.m | demo_mlp_learning_M_units.m | demo_plot2d.m | +

```
1   %% Solving a nonlinear system for MLP learning, created by Jiann-Ming Wu
2   %    Department of Applied Mathematics, National Dong Hwa University
3   %
4   %  Using Matlab (R)
5   %  2018 dec. 3
6   %  MLP learning is resolved by the Levenberg-Marquardt method
7   %
8   function demo_mlp_learning_M_units()
```

Command Window

```
as measured by the value of the function tolerance, and
the problem appears regular as measured by the gradient.

<stopping criteria details>
^^ mean absolute error for ap func g 0.000000

ans =

   4.4442e-21
```

Mean square error



```
** mean absolute error for ap func g 0.001100

ans =

   1.9939e-06
```

Mean square error