A simple CNN

Levenberg-Marquardt learning
Regressions

Applicability of Levenberg-Marquardt method for learning CNN

- LM method is verified for learning CNN

- A simple CNN is composed of K=2 filters in the first hidden layer. The convolution results are transferred by the tanh function and added to form the final output.

- The input is a 3-ary matrix. The matrix size is 28x28x100. There are 100 28x28 input patterns.

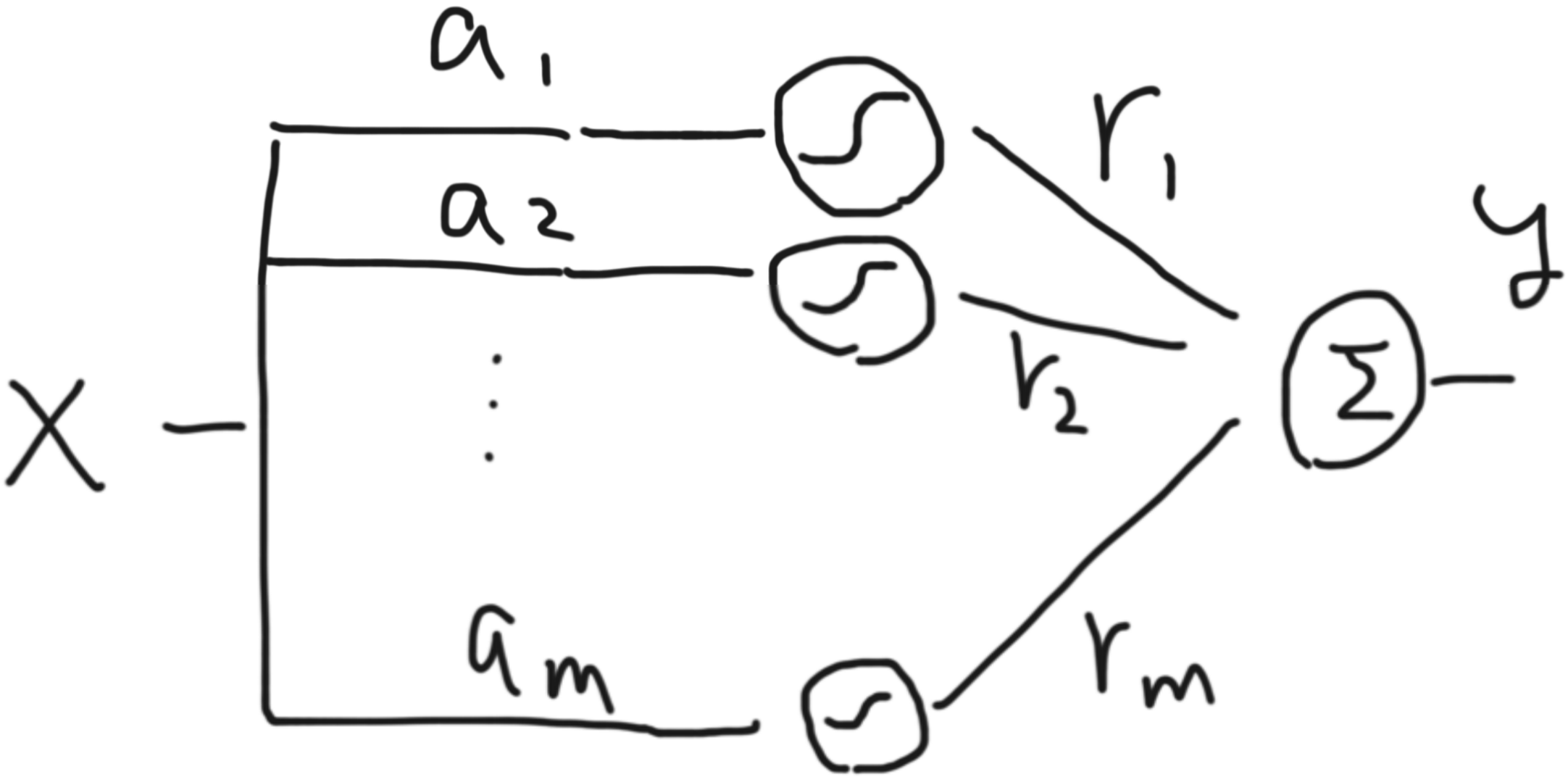
28
28
9
5
5
5
5
5
5
24
24
24
24
24
24
CNN
regressions
M
Y
W

$$f(x;\theta)$$

$$= \sum_{i=1}^{M} r_i \tanh(a_i^T x + b_i)$$

adaptive parameters

$$\theta = \{r_i \in R\} \cup \{a_i \in R^d\} \cup \{b_i\}$$



multilayer perceptions

MLP

```
function h = g_hat2(x1,x2,c,M)
A = [x1 x2 ones(length(x1),1)];
d=2;W=[];
%W=reshape(c(1:M*(d+1)),d+1,M);
for j=1:M
   W=[W c((j-1)*(d+1)+1:j*(d+1))'];
end
v =tanh(A*W);
h = v*c(M*(d+1)+1:end-1)';
h=h+ones(length(x1),1)*c(end);
end
```

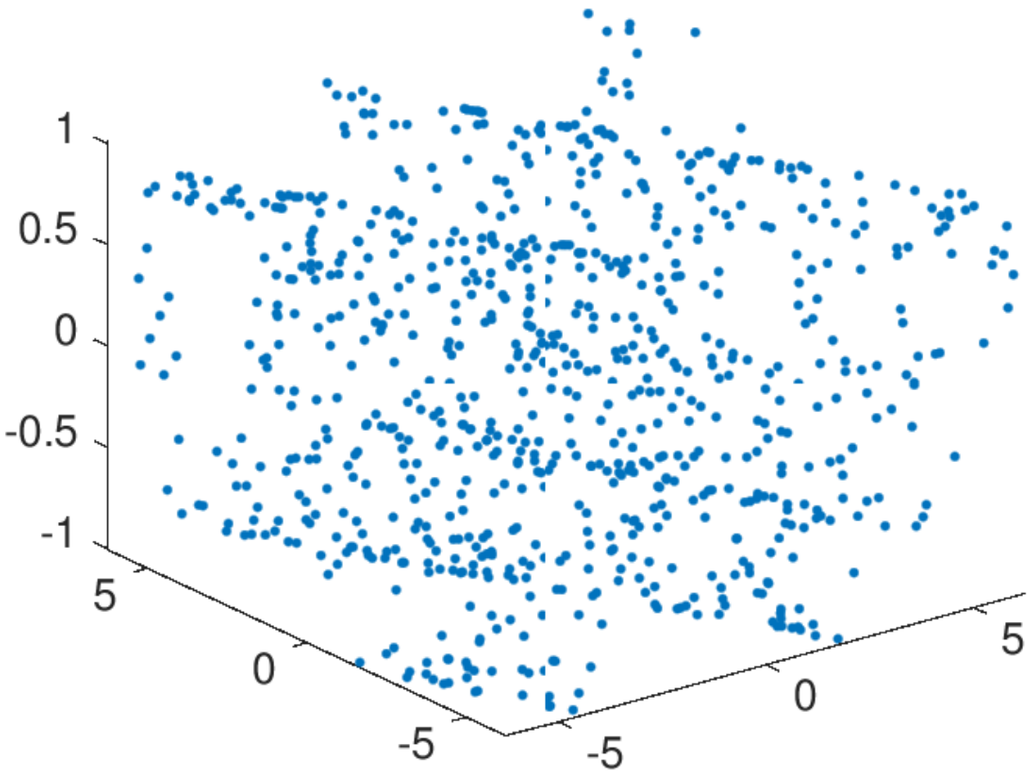
$$f(x;\theta) = \sum_{i=1}^{M} r_i \tanh(a_i^T x + b_i)$$

adaptive parameters

$$\theta = \{r_i \in R\} \cup \{a_i \in R^d\} \cup \{b_i\}$$

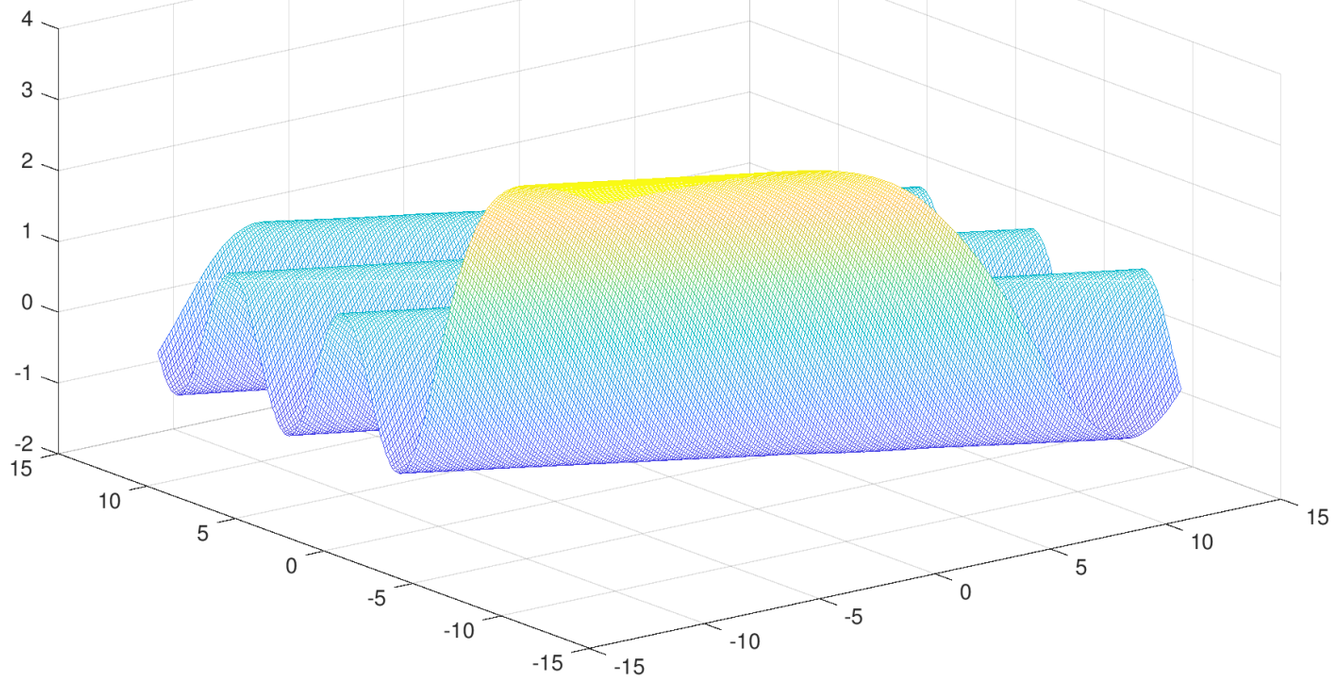
Projection Tanh

DemoLMlearning



DemoLMlear
ning

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



RBF Network function

$$y(t\,|\,\theta) = G(\mathbf{x}[t]\,|\,\theta)$$

$$= w_0 + \sum_{m=1}^{M} w_m \exp\left(-\frac{\left\|\mathbf{x}[t] - \boldsymbol{\mu}_m\right\|^2}{2\sigma_m^2}\right)$$

Network parameter

$$\theta = \{\mathrm{w}_i\}_i \cup \{\boldsymbol{\mu}_i\}_i \cup \{\sigma_i\}_i$$

$$f(x;\theta)$$

$$= \sum_{i=1}^{M} r_i \tanh(a_i^T x + b_i) + r0$$

adaptive parameters

$$\theta = \{r_i \in R\} \cup \{a_i \in R^d\} \cup \{b_i\}$$

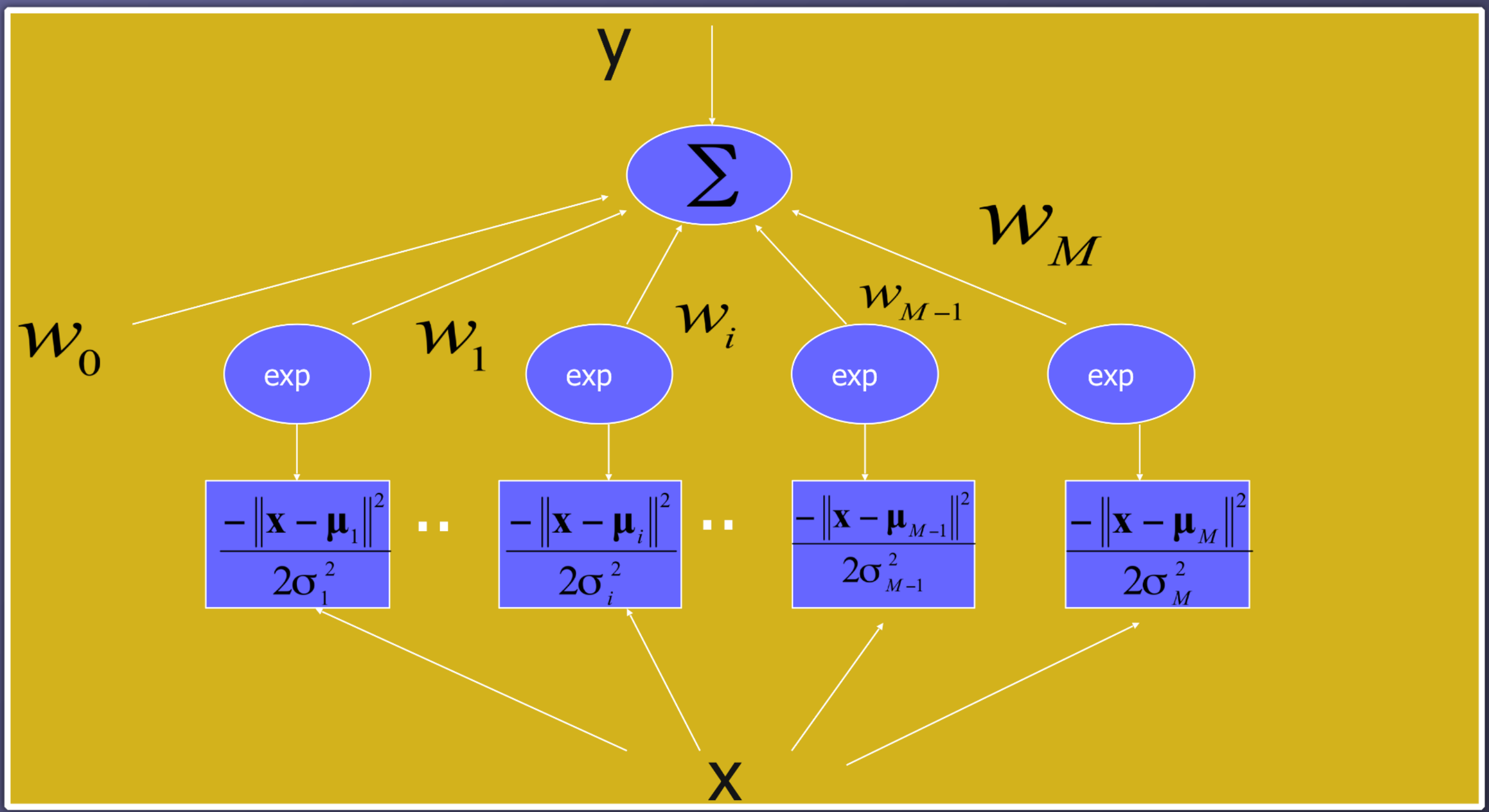
$$y(t\,|\,\theta) = G(\mathbf{x}[t]\,|\,\theta)$$

$$= w_0 + \sum_{m=1}^{M} w_m \exp\left(-\frac{\left\|\mathbf{x}[t] - \boldsymbol{\mu}_m\right\|^2}{2\sigma_m^2}\right)$$

Network parameter

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

RBF Network


y
Σ
w_0
w_1
w_i
w_{M-1}
w_M
exp
exp
exp
exp
$\frac{-\|\mathbf{x}-\boldsymbol{\mu}_1\|^2}{2\sigma_1^2}$
..
$\frac{-\|\mathbf{x}-\boldsymbol{\mu}_i\|^2}{2\sigma_i^2}$
..
$\frac{-\|\mathbf{x}-\boldsymbol{\mu}_{M-1}\|^2}{2\sigma_{M-1}^2}$
$\frac{-\|\mathbf{x}-\boldsymbol{\mu}_M\|^2}{2\sigma_M^2}$
x

Network parameter

$$\theta = [\mathbf{u}_1^T\ \mathbf{u}_2^T \cdots \mathbf{u}_M^T\ \sigma_1\ \sigma_2 \cdots \sigma_{\mathrm{M}}\ \mathrm{w}_0\ \mathrm{w}_1\ \mathrm{w}_2 \cdots \mathrm{w}_{\mathrm{M}}]^T$$

$$= [\theta_1, \ldots, \theta_{M*d+2M+1}]$$

$$w_1 \in M_{3\times 1} \qquad c_1 \in R$$

$$\vdots \qquad\qquad \vdots$$

$$w_M \in M_{3\times 1} \qquad c_M \in R$$

$$c_0 \in R$$

$$\theta = \left[w_1^T \;\; w_2^T \;\; \cdots \;\; w_M^T \;\; c_1 \;\; c_2 \;\; \cdots \;\; c_M \;\; c_0 \right]$$

$\mu_1 \in M_{2\times 1}$ $\quad \beta_1 \in R \quad w_1 \in R$

$\vdots \qquad\qquad \vdots \qquad\qquad \vdots$

$\mu_M \in M_{2\times 1} \quad \beta_M \in R \quad w_M \in R$

$w_0 \in R$

$\theta = [\mu_1^T \; \mu_2^T \; \cdots \; \mu_M^T \; \beta_1 \cdots \beta_M \; w_1 \cdots w_M \; w_0]$

```
function h = g_hat2(x1,x2,c,M)
A = [x1 x2 ones(length(x1),1)];
d=2;W=[];
%W=reshape(c(1:M*(d+1)),d+1,M);
for j=1:M
   W=[W c((j-1)*(d+1)+1:j*(d+1))'];
end
% revise
v =tanh(A*W);
h = v*c(M*(d+1)+1:end-1)';
h=h+ones(length(x1),1)*c(end);
end
```

RBF Network function

$$y(t\,|\,\theta) = G(\mathbf{x}[t]\,|\,\theta)$$

$$= w_0 + \sum_{m=1}^{M} w_m \exp\left(-\frac{\left\|\mathbf{x}[t]-\boldsymbol{\mu}_m\right\|^2}{2\sigma_m^2}\right)$$

Network parameter

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

DemoLMlear
ning

28
28
9
5
5
5
5
5
5
24
24
24
24
24
24
CNN
regressions
M
Y
W

```
function ans = g_hat_CNN(A,BB,M,f_size)

B_size=M*f_size*f_size;
B=BB(1:B_size);
WW=BB(B_size+1:end);
%
for i=1:M
   cc(i,:) = B((i-1)*f_size^2+1:i*f_size^2);
   for j=1:f_size
       C(j,:,i)=cc(i,(j-1)*f_size+1:j*f_size);
   end
end
W_size=sqrt(length(WW))
for k=1:W_size
    W(k,:)=WW((k-1)*W_size+1:k*W_size)
end
N=size(A,3);
for n=1:N
    v(:,:,n)=zeros(size(A,2)-f_size+1,size(A,2)-f_size+1);
for i=1:M
    h(:,:,i)=conv(A(:,:,n),C(:,:,i),'valid');
    v(:,:,i)=v(:,:,i)+tanh(h(:,:,i));
end
    y_hat(:,n)=sum(sum(v(:,:,n).*W));
end

end
```

DemoLMlear
ning

```
%% Solving a nonlinear system for LM learning, created by Jiann-Ming Wu
%    Department of Applied Mathematics, National Dong Hwa University
%
%  Using Matlab (R)
%  2018 dec. 25
%  learning is resolved by the Levenberg-Marquardt method
%

function demo_LM_CNN_regressions()
syms c   % adaptable parameters in an MLP network
M=3;d=28;N=300
f_size=5;
% initialization
c0_size=f_size*f_size*M;
c0_size=c0_size+(d-f_size+1)^2
c0=rand(1,c0_size)*2-1;
size(c0)
% preparation of training data
% uniform sampling
% Substitute to the target function g
z=rand(d,d,N);
A=z>0.5;
y=rand(N,1)*0.3-0.15;
size(y)
y_hat = g_hat_CNN(z,c0,M,f_size);

size(y_hat)
% Levenberg-Marquardt method
options = optimoptions('fsolve','Algorithm', 'levenberg-marquardt')
% specification of a nonlinear system for MLP_learning
f = @(c)learning_CNN(c,A,y,M,f_size);

% apply fsolve
% Verification of c_zero
c_zero = fsolve(f,c0,options);
BB=c_zero;
F = learning_CNN(BB,A,y,M,f_size);
mean((F).^2)
%mean((g_hat2(z(:,1),z(:,2),c_zero,M)-y).^2)
% testing phase
%z_test=rand(100,2)*2*pi-pi;
%y_test=g_sin(z_test(:,1),z_test(:,2));
%mean((g_hat2(z_test(:,1),z_test(:,2),c_zero,M)-y_test).^2)
end
```

```
function F = learning_CNN(BB,A,y,M,f_size)
B_size=M*f_size*f_size;
B=BB(1:B_size);
WW=BB(B_size+1:end);
%
for i=1:M
   cc(i,:) = B((i-1)*f_size^2+1:i*f_size^2);
   for j=1:f_size
      C(j,:,i)=cc(i,(j-1)*f_size+1:j*f_size);
   end
end
W_size=sqrt(length(WW))
for k=1:W_size
    W(k,:)=WW((k-1)*W_size+1:k*W_size);
end
N=size(A,3);
for n=1:N
    v(:,:,n)=zeros(size(A,2)-f_size+1,size(A,2)-f_size+1);
for i=1:M
    h(:,:,i)=convn(A(:,:,n),C(:,:,i),'valid');
    v(:,:,n)=v(:,:,n)+tanh(h(:,:,i));
end
    y_hat(n)=sum(sum(v(:,:,n).*W));
end
F=y_hat;
F =F-y;
end
```

```
function y_hat = g_hat_CNN(A,BB,M,f_size)

B_size=M*f_size*f_size;
B=BB(1:B_size);
WW=BB(B_size+1:end);
%
for i=1:M
   cc(i,:) = B((i-1)*f_size^2+1:i*f_size^2);
   for j=1:f_size
      C(j,:,i)=cc(i,(j-1)*f_size+1:j*f_size);
   end
end
W_size=sqrt(length(WW))
for k=1:W_size
    W(k,:)=WW((k-1)*W_size+1:k*W_size);
end
N=size(A,3);
for n=1:N
    v(:,:,n)=zeros(size(A,2)-f_size+1,size(A,2)-f_size+1);
for i=1:M
    h(:,:,i)=convn(A(:,:,n),C(:,:,i),'valid');
    v(:,:,n)=v(:,:,n)+tanh(h(:,:,i));
end
y_hat(n)=sum(sum(v(:,:,n).*W));
end

end
```