第八章作业

> 专业：计算机科学与技术
>
> 学号：17341178
>
> 姓名：薛伟豪

8-1

> **采用模糊RBF网络、CMAC网络逼近非线性对象 $y(k)=(u(k-1)-0.9y(k-1))/(1+y(k-1)^2)$，分别进行Matlab仿真。**

解：根据要求，有：

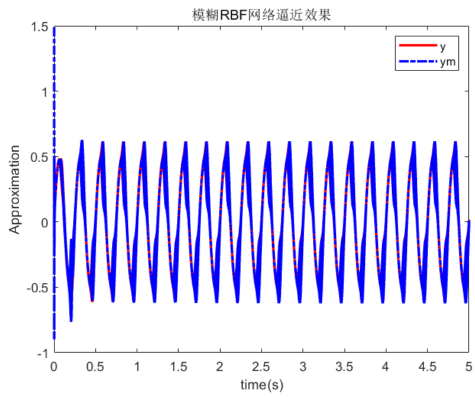
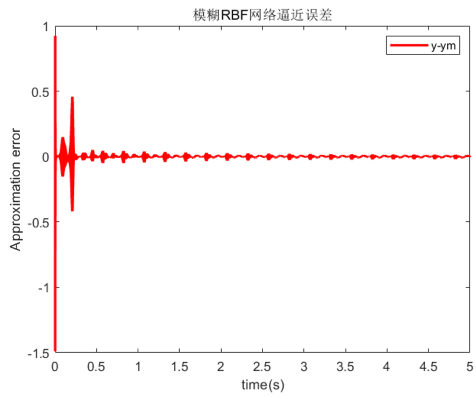
- **采用模糊RBF网络逼近对象**

采样时间取1ms。输入信号为$u(t)=sin(8\pi t)$，神经网络权值W的初始值取$[-1,+1]$之间的随机值，高斯基函数的参数取值为$c=[c_{ij}]=\begin{bmatrix}-1 & -0.5 & 0 & 0.5 & 1\\ -1.5 & -1 & 0 & 1 & 1.5\end{bmatrix}^T$，$b_j=1.0$，$i=1,2$，$j=1,2,3,4,5$。网络的学习参数取$\eta=0.50$，$\alpha=0.05$。模糊RBF网络Matlab仿真代码如下所示：

```
clc, clear;

eta = 0.50;
alpha = 0.05;
bj = 1.0;
c = [-1 -0.5 0 0.5 1;
   -1.5 -1 0 1 1.5];
w = rands(25,1);
w_1 = w;
w_2 = w_1;
u_1 = 0.0;
y_1 = 0.0;

ts = 0.001;
for k = 1:1:50000
    time(k) = k*ts;
    u(k) = sin(0.1*k*ts);
    y(k) = (u_1-0.9*y_1)/(1+y_1^2);
    x = [u(k),y(k)]';                    % Layer1:input
    f1 = x;
    for i = 1:1:2                        % Layer2:fuzzation
        for j = 1:1:5
            net2(i,j) = -(f1(i)-c(i,j))^2/bj^2;
        end
    end
    for i = 1:1:2
        for j = 1:1:5
            f2(i,j) = exp(net2(i,j));
        end
```

```
    end

    for j = 1:1:5                         % Layer3:fuzzy inference(49 rules)
        m1(j) = f2(1,j);
        m2(j) = f2(2,j);
    end

    for i = 1:1:5
        for j = 1:1:5
            ff3(i,j) = m2(i)*m1(j);
        end
    end
    f3 = [ff3(1,:),ff3(2,:),ff3(3,:),ff3(4,:),ff3(5,:)];

    f4 = w_1'*f3';                        % Layer4:output
    ym(k) = f4;
    e(k) = y(k)-ym(k);
    d_w = 0*w_1;
    for j = 1:1:25
        d_w(j) = eta*e(k)*f3(j);
    end
    w = w_1+d_w+alpha*(w_1-w_2);
    u_1 = u(k);
    y_1 = y(k);
    w_2 = w_1;
    w_1 = w;
end

figure(1);
plot(time,y,'r',time,ym,'-.b','linewidth',2);
xlabel('time(s)');
ylabel('Approximation');
title('模糊RBF网络逼近效果');
legend('y','ym');
figure(2);
plot(time,y-ym,'r','linewidth',2);
xlabel('time(s)');
ylabel('Approximation error');
title('模糊RBF网络逼近误差');
legend('y-ym');
```

实验结果如下所示：

模糊RBF网络逼近效果	模糊RBF网络逼近误差
	

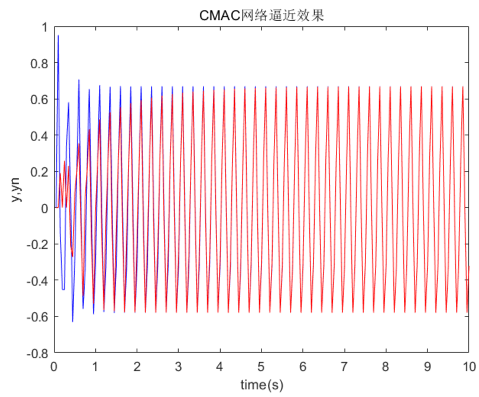
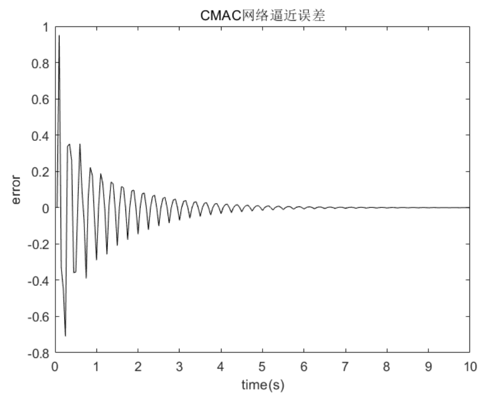
- **采用CMAC网络逼近对象**

在仿真中，网络输入信号为信号$u(t)=sin(8\pi t)$，采样时间取0.05s。网络参数取$M=200$，$N=20$，$c=3$，$\eta=20$，$\alpha=0.05$，可保证$c\ll M$及$c\le N\le M$。CMAC网络Matlab仿真代码如下所示：

```
clc, clear;

eta = 0.20;
alpha = 0.05;
M = 200;
N = 20;
c = 3;
w = zeros(N,1);
w_1 = w;
w_2 = w;
d_w = w;
u_1 = 0;
y_1 = 0;

ts = 0.05;
for k = 1:1:200
    time(k) = k*ts;
    u(k) = sin(8*pi*k*ts);
    xmin = -1.0;
    xmax = 1.0;
    for i = 1:1:c
        s(k,i) = round((u(k)-xmin)*M/(xmax-xmin))+i;   %Quantity:U-->AC
        ad(i) = mod(s(k,i),N)+1;                       %Hash transfer:AC-->AP
    end
    sum = 0;
    for i = 1:1:c
        sum = sum+w(ad(i));
    end
    yn(k) = sum;
    y(k) = (u_1-0.9*y_1)/(1+y_1^2);                      %Nonlinear model
    error(k) = y(k)-yn(k);
    for i = 1:1:c
        ad(i) = mod(s(k,i),N)+1;
        j = ad(i);
        d_w(j) = eta*error(k);
        w(j) = w_1(j)+d_w(j)+alpha*(w_1(j)-w_2(j));
    end
    w_2 = w_1;
    w_1 = w;
    u_1 = u(k);
    y_1 = y(k);
end

figure(1);
plot(time,y,'b',time,yn,'r');
xlabel('time(s)');
ylabel('y,yn');
title('CMAC网络逼近效果');
figure(2);
plot(time,y-yn,'k');
```

```
xlabel('time(s)');
ylabel('error');
title('CMAC网络逼近误差');
```

实验结果如下所示：

CMAC网络逼近效果	CMAC网络逼近误差
	

8-2

> **参照本书10.6节仿真实例，构造30个城市的位置坐标，采用Hopefiled网络，实现30个城市路径的TSP问题优化，并进行Matlab仿真**

解：根据要求，我们构造30个城市的位置坐标如下：

```
0.1 0.1
0.9 0.5
0.9 0.1
0.45 0.9
0.9 0.8
0.7 0.9
0.1 0.45
0.45 0.1
0.75 0.9
0.69 0.12
0.47 0.49
0.36 0.97
0.82 0.14
0.3 0.05
0.18 0.64
0.8 0.5
0.6 0.3
0.3 0.2
0.5 0.9
0.4 0.6
0.86 0.47
0.95 0.68
0.74 0.26
0.86 0.22
0.15 0.2
```

```
0.63 0.12
0.2 0.9
0.37 0.39
0.2 0.5
0.47 0.1
```

采用Hopefiled网络，对着30个城市路径的TSP问题完成优化，Matlab仿真代码如下

```
% TSP Solving by Hopfield Neural Network
function TSP_hopfield()
clc;
clear;
clear all;
close all;25

%Step 1:置初值
A = 1.5;
D = 1;
Mu = 50;
Step = 0.01;

%Step 2: %计算N个城市之间距离,计算初始路径长度
N = 30;
cityfile = fopen( 'city30.txt', 'rt' );
cities = fscanf( cityfile, '%f %f',[ 2,inf] )
fclose(cityfile);
Initial_Length = Initial_RouteLength(cities);  % 计算初始路径长度

DistanceCity = dist(cities',cities);
%Step 3: 神经网络输入的初始化
U = 0.001*rands(N,N);
V = 1./(1+exp(-Mu*U)); % S函数

for k = 1:1:10000  %神经网络优化
times(k) = k;
%Step 4: 计算du/dt
    dU = DeltaU(V,DistanceCity,A,D);
%Step 5: 计算u(t)
    U = U+dU*Step;
%Step 6: 计算网络输出
    V = 1./(1+exp(-Mu*U)); % S函数
%Step 7: 计算能量函数
    E = Energy(V,DistanceCity,A,D);
    Ep(k) = E;
%Step 8: 检查路径合法性
    [V1,CheckR] = RouteCheck(V);
end

%Step 9:显示及作图
if(CheckR == 0)
   Final_E = Energy(V1,DistanceCity,A,D);
   Final_Length = Final_RouteLength(V1,cities); %计算最终路径长度
    disp('迭代次数');k
    disp('寻优路径矩阵');V1
    disp('最优能量函数:');Final_E
    disp('初始路程:');Initial_Length
    disp('最短路程:');Final_Length
```

```
    PlotR(V1,cities);  %寻优路径作图
else
    disp('寻优路径矩阵:');V1
    disp('寻优路径无效,需要重新对神经网络输入进行初始化');
end

figure(2);
plot(times,Ep,'r');
title('Energy Function Change');
xlabel('k');ylabel('E');

%%%%%%%计算能量函数
function E = Energy(V,d,A,D)
[n,n] = size(V);
t1 = sumsqr(sum(V,2)-1);
t2 = sumsqr(sum(V,1)-1);
PermitV = V(:,2:n);
PermitV = [PermitV,V(:,1)];
temp = d*PermitV;
t3 = sum(sum(V.*temp));
E = 0.5*(A*t1+A*t2+D*t3);

%%%%%%%计算du/dt
function du = DeltaU(V,d,A,D)
[n,n] = size(V);
t1 = repmat(sum(V,2)-1,1,n);
t2 = repmat(sum(V,1)-1,n,1);
PermitV = V(:,2:n);
PermitV = [PermitV, V(:,1)];
t3 = d*PermitV;
du = -1*(A*t1+A*t2+D*t3);

%%%%%%标准化路径，并检查路径合法性：要求每行每列只有一个“1”
function [V1,CheckR] = RouteCheck(V)
[rows,cols] = size(V);
V1 = zeros(rows,cols);
[XC,Order] = max(V);
for j = 1:cols
    V1(Order(j),j) = 1;
end
C = sum(V1);
R = sum(V1');
CheckR = sumsqr(C-R);

%%%%%%%%计算初始总路程
function L0 = Initial_RouteLength(cities)
[r,c] = size(cities);
L0 = 0;
for i = 2:c
   L0 = L0+dist(cities(:,i-1)',cities(:,i));
end

%%%%%%%计算最终总路程
function L = Final_RouteLength(V,cities)
[xxx,order] = max(V);
New = cities(:,order);
New = [New New(:,1)]
```

```
[rows,cs] = size(New);

L = 0;
for i = 2:cs
    L = L+dist(New(:,i-1)',New(:,i));
end

%%%%%%路径寻优作图
function PlotR(V,cities)
figure;

cities = [cities cities(:,1)];

[xxx,order] = max(V);
New = cities(:,order);
New = [New New(:,1)];

subplot(1,2,1);
plot( cities(1,1), cities(2,1),'r*' );   %First city
hold on;
plot( cities(1,2), cities(2,2),'+' );    %Second city
hold on;
plot( cities(1,:), cities(2,:),'o-' ), xlabel('X axis'), ylabel('Y axis'), title('Original Route');
axis([0,1,0,1]);

subplot(1,2,2);
plot( New(1,1), New(2,1),'r*' );   %First city
hold on;
plot( New(1,2), New(2,2),'+' );    %Second city
hold on;
plot(New(1,:),New(2,:),'o-');
title('TSP solution');
xlabel('X axis');ylabel('Y axis');
title('New Route');
axis([0,1,0,1]);
axis on
```

仿真结果如下所示：

Energy Function Change	Route	Result