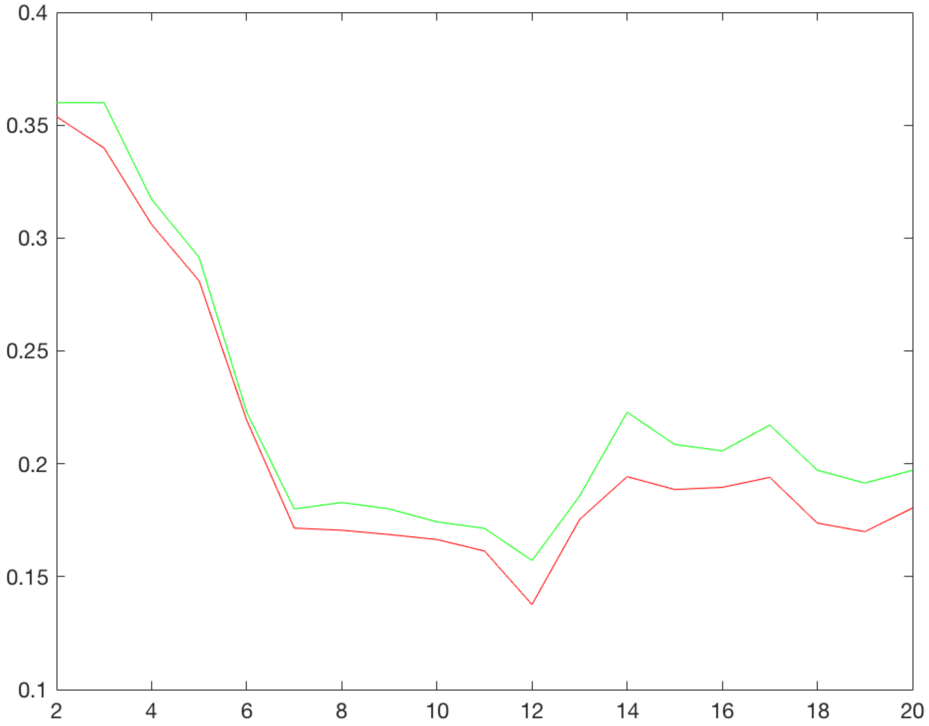
Rongrong Su

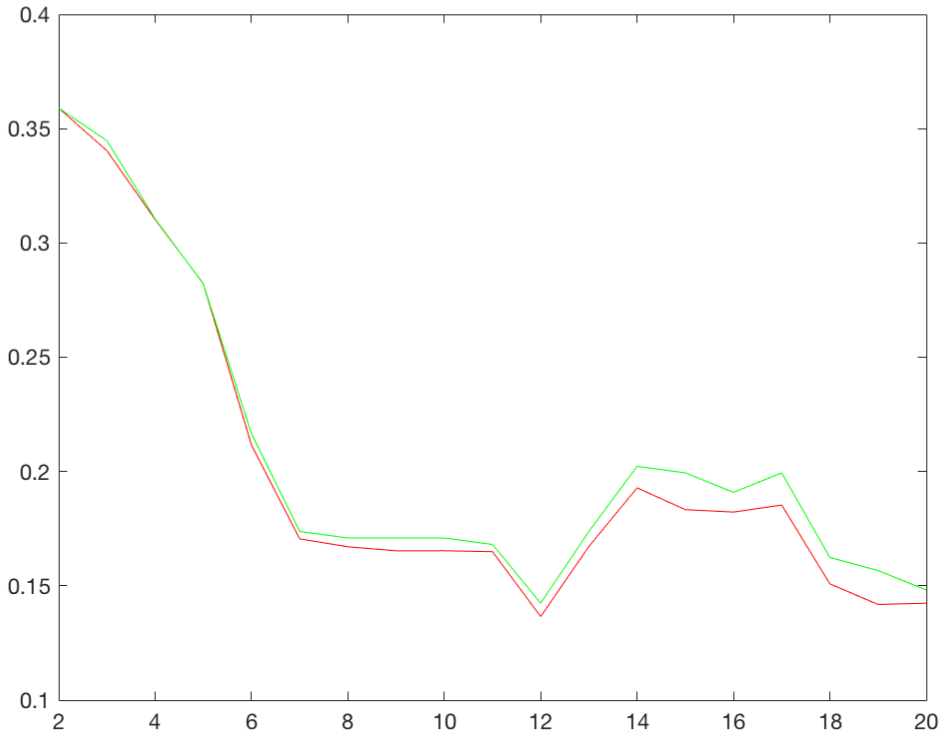
20751802

1.For cross validation:



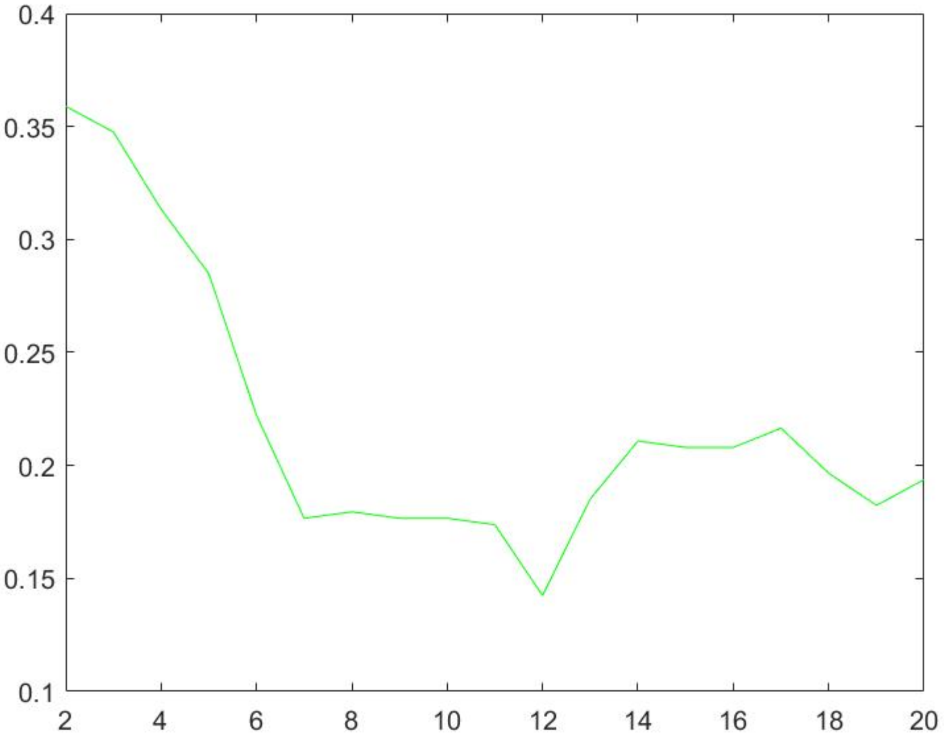
It appears that the optimum number of basis functions is 12.

For loo:



It appears that the optimum number of basis functions is 12.

For cloo:

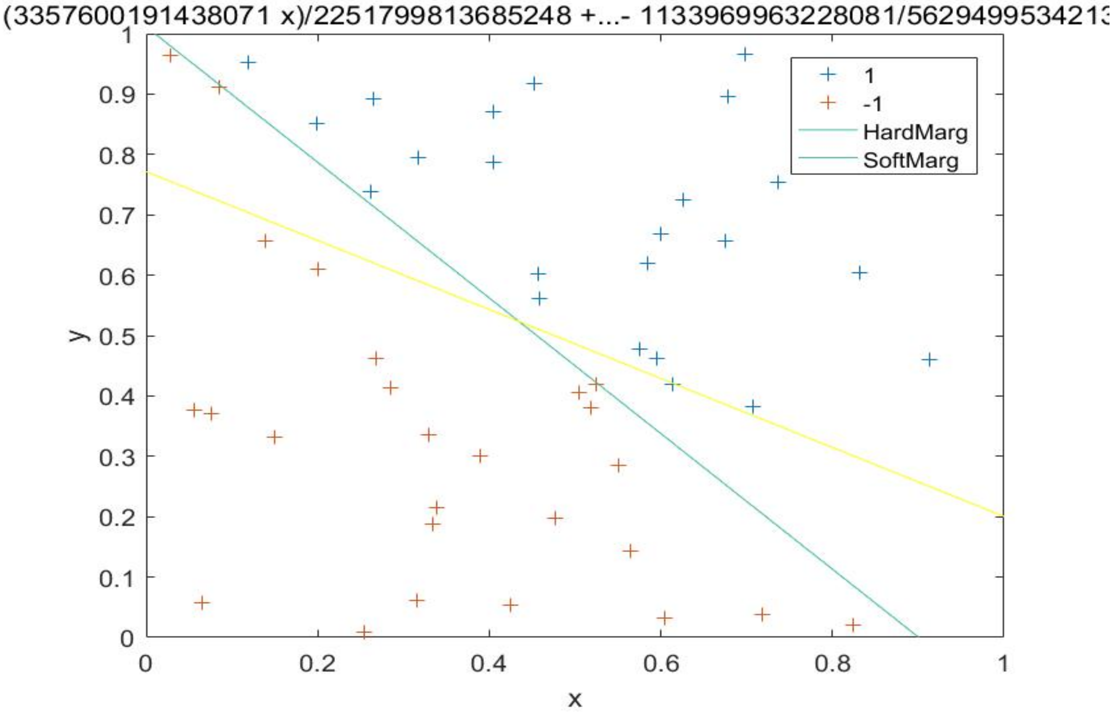


It appears that the optimum number of basis functions is 12.

	Training error	Test error	Number of basis
10-fold-CV	0.1377	0.1571	12
LOO	0.1366	0.1425	12
CLOO		0.1425	12

2.

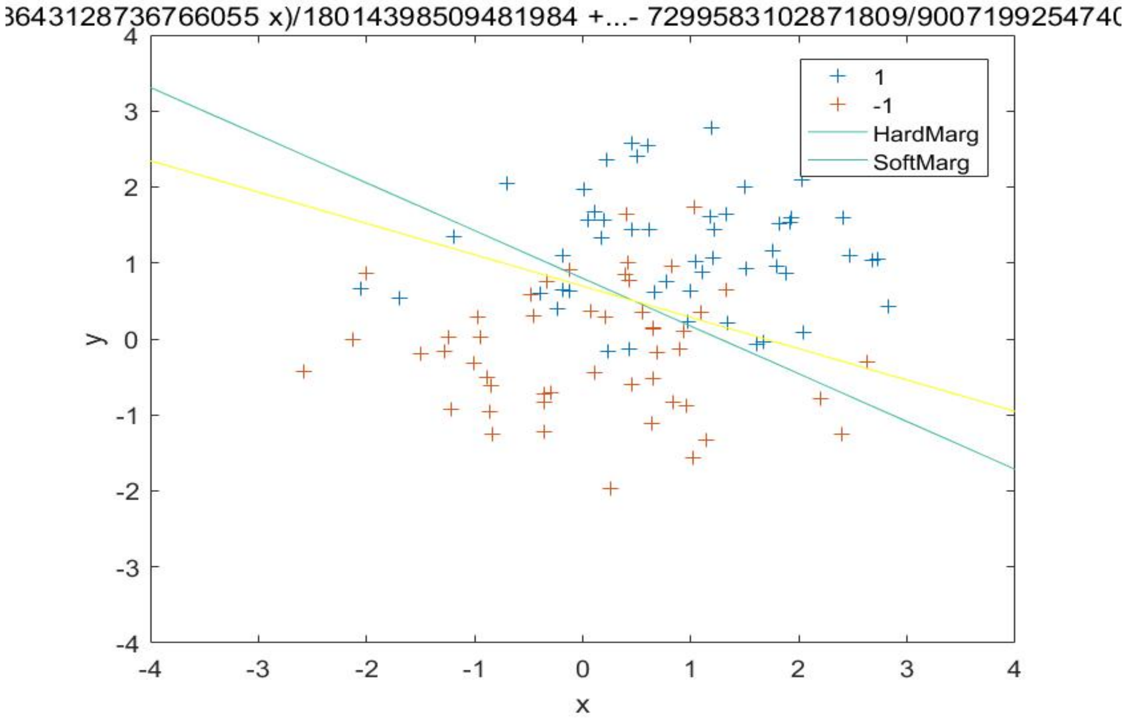
Linear:



tables:

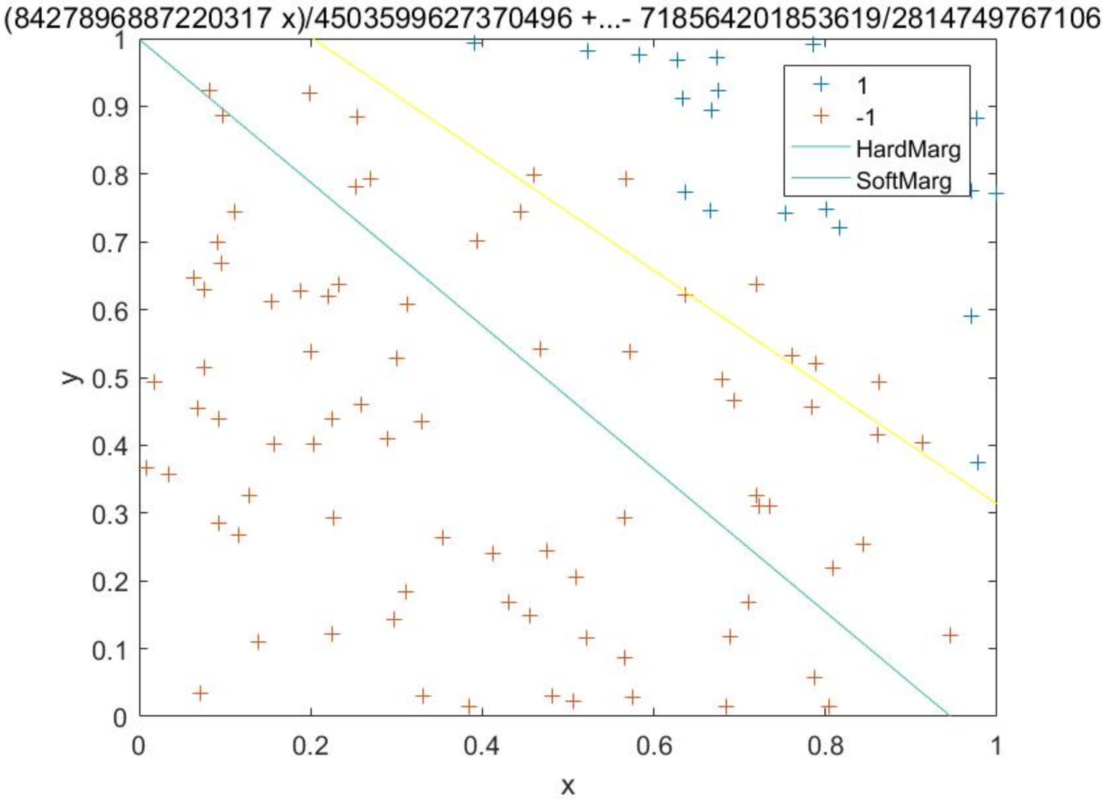
	train_error	test_error
Hard margin	0	0.0379
Soft margin	0.0588	0.1022

Noisy linear:



	train_error	test_error
Hard margin	0.1881	0.2492
Soft margin	0.2079	0.2659

Quadratic:



	train_error	test_error
Hard margin	0.2673	0.3037
Soft margin	0.0693	0.0723

Code:

Q1

(a)

```
rng(1);
num=5;
[idx, mu]=kmeans(X', num);
mu=mu';
sigma=zeros(num, 1);
phi=zeros(num,size(X,2));
for i=1:size(X,2)
sigma(idx(i), 1)=sigma(idx(i), 1)+norm(X(:,i)-
mu(:,idx(i)))^2;
end

for i=1:num
   sigma(i,1)=sigma(i,1)/sum(idx==i);
end

for i=1:num
   for j=1:size(X,2)
      phi(i,j)=exp(-norm(X(:,j)-
mu(:,i))^2/(sigma(i,1)+0.000001));
   end
end



W=inv(phi*phi')*phi*y;
y_est=W'*phi;
for i=1:length(y_est)
   if y_est(i)>=0.5
      y_est(i)=1;
   else
      y_est(i)=0;
```

```
    end
end

error_rate = 1-sum(y_est'==y)/length(y);
```

(b)

```
%STANDARD CROSS VALIDATION
num=20; %cluster 1-20 centers
cv_num=10; %divide raw date into 13 groups
a=35;
error_train=zeros(num,1);
error_test=zeros(num,1);
for n=2:num
    rng(150);
    [idx, mu]=kmeans(X', n);
    mu=mu';
    sigma=zeros(n, 1);
    train_error=zeros(cv_num,1);
    test_error=zeros(cv_num,1);


    %calculate sigma
    for l=1:size(X,2)
        sigma(idx(l), 1)=sigma(idx(l), 1)+norm(X(:,l)-
mu(:,idx(l)))^2;
    end
    for i=1:n
        sigma(i,1)=sigma(i,1)/sum(idx==i);
    end

    %cross validation
    for i=1:cv_num

        %split data
```

```
       train=[X(:,1:a*(i-1)),X(:,(a*(i)+1):size(X,2))];
       test=X(:,(a*(i-1)+1):a*(i));
       y_train=[y(1:a*(i-1),1);y((a*(i)+1):size(X,2),1)];
       y_test=y((a*(i-1)+1):a*i);


       %calculate phi&W&train-error for train data
       phi=zeros(n,size(train,2));
       for k=1:n
          for j=1:size(train,2)
             phi(k,j)=exp(-norm(train(:,j)-
mu(:,k))^2/(sigma(k,1)+0.000001));
          end
       end
       W=pinv(phi*phi')*(phi*y_train);
       train_est=phi'*W;
       for t=1:length(train_est)
          if train_est(t)>=0.5
             train_est(t)=1;
          else
             train_est(t)=0;
          end
       end
       train_error(i,1)=1-
sum(train_est==y_train)/size(y_train,1);

       %calculate phi&W&test-error for test data
       phi=zeros(n,a);
       for k=1:n
          for j=1:a
             phi(k,j)=exp(-norm(test(:,j)-
mu(:,k))^2/(sigma(k,1)+0.000001));
          end
```

```
        end

        test_est=phi'*W;
        for w=1:length(test_est)
            if test_est(w)>=0.5
                test_est(w)=1;
            else
                test_est(w)=0;
            end
        end
        test_error(i,1)=1-
sum(test_est==y_test)/size(y_test,1);
    end
    error_train(n,1)=sum(train_error)/cv_num;
    error_test(n,1)=sum(test_error)/cv_num;
end
plot(2:num,error_train(2:num,1),'r');
hold on;
plot(2:num,error_test(2:num,1),'g');

%STANDARD LOO
num=20; %cluster 1-20 centers
a=1;%size
error_train=zeros(num,1);
error_test=zeros(num,1);
for n=2:num
    rng(150);
    [idx, mu]=kmeans(X', n);
    mu=mu';
    sigma=zeros(n, 1);
    train_error=zeros(cv_num,1);
    test_error=zeros(cv_num,1);
    %calculate sigma
    for l=1:size(X,2)
        sigma(idx(l), 1)=sigma(idx(l), 1)+norm(X(:,l)-
mu(:,idx(l)))^2;
    end
    for i=1:n
        sigma(i,1)=sigma(i,1)/sum(idx==i);
    end

    %cross validation
    for i=1:cv_num

        %split data
        train=[X(:,1:a*(i-1)),X(:,(a*(i)+1):size(X,2))];
        test=X(:,(a*(i-1)+1):a*(i));
        y_train=[y(1:a*(i-1),1);y((a*(i)+1):size(X,2),1)];
        y_test=y((a*(i-1)+1):a*i);



        %calculate phi&W&train-error for train data
        phi=zeros(n,size(train,2));
        for k=1:n
            for j=1:size(train,2)
                phi(k,j)=exp(-norm(train(:,j)-
mu(:,k))^2/(sigma(k,1)+0.000001));
            end
        end
        W=pinv(phi*phi')*(phi*y_train);
        train_est=phi'*W;
        for t=1:length(train_est)
            if train_est(t)>=0.5
                train_est(t)=1;
            else
                train_est(t)=0;
            end
        end
        train_error(i,1)=1-
sum(train_est==y_train)/size(y_train,1);

        %calculate phi&W&test-error for test data
        phi=zeros(n,a);
        for k=1:n
            for j=1:a
                phi(k,j)=exp(-norm(test(:,j)-
mu(:,k))^2/(sigma(k,1)+0.000001));
            end
        end
        test_est=phi'*W;
        for w=1:length(test_est)
            if test_est(w)>=0.5
                test_est(w)=1;
            else
                test_est(w)=0;
            end
        end
        test_error(i,1)=1-
sum(test_est==y_test)/size(y_test,1);
    end
    error_train(n,1)=sum(train_error)/cv_num;
    error_test(n,1)=sum(test_error)/cv_num;
end
plot(2:num,error_train(2:num,1),'r');
hold on;
plot(2:num,error_test(2:num,1),'g');
```

```
%STANDARD CROSS VALIDATION without iteration
num=20; %cluster 1-20 centers
rng(150);
y_error=zeros(num/2,1);
for n=2:num
    %cluster data using Kmeans
    [idx,mu]=kmeans(X',n);
    mu=mu';
    sigma=zeros(n,1);
    for j=1:size(X,2)
        sigma(idx(j),1)=sigma(idx(j),1)+norm(X(:,j)-
mu(:,idx(j)))^2;
    end
    for i=1:n
        sigma(i,1)=sigma(i,1)/sum(idx==i);
    end

    %calculate phi&W&train-error for train data
    phi=zeros(n,size(X,2));
    for i=1:n
        for j=1:size(X,2)
            phi(i,j)=exp(-norm(X(:,j)-
mu(:,i))^2/(sigma(i,1)+0.0001));
        end
    end
    w=pinv(phi*phi')*(phi*y);
    y_est=phi'*w;
    H=phi'*pinv(phi*phi')*phi;
    for t=1:length(y_est)
        if y_est(t)>=0.5
            y_est(t)=1;
        else
            y_est(t)=0;
        end
    end
    y_error(i,1)=1-sum(y_est==y)/size(y,1);
end

plot(2:num,y_error(2:num,1),'g');

y_error(13,1)
```

Q2

```
%harmargin
function [b,b0]=hardmarg(X,y)
[d,n]=size(X);
Y=diag(y)
H=(X*Y)'*(X*Y);
f=-1*ones(n,1);
Aeq=Y';
beq=0;
lb=zeros(n,1);
alpha=quadprog(H,f,[],[],Aeq,beq,lb);
b=X*Y*alpha;
[argVal,argMax]=max(alpha);
b0=1/y(argMax,1) - b'* X(:, argMax);
end

%softmargin
function [b,b0] = SoftMarg(X,y,gamma)
[d,n]=size(X);
Y=diag(y)
H=(X*Y)'*(X*Y);
f=-1*ones(n,1);
Aeq=Y';
beq=0;
```

```
lb=zeros(n,1);
ub=gamma*ones(n,1);
alpha = quadprog(H, f, [], [], A, bb, lb, ub);
b=X*Y*alpha;
while alpha(arg,1) < 0.01 || alpha(arg,1) > gamma
arg = arg+1;
end
b0s = 1/y(arg,1) - b'* X(:, arg);
end

%classifer
function[yhat]=classify(Xtest,b,b0)
[d,n] = size(X);
yhat=(b'*Xtest+b0)';
for i=1:n
    if yhat(i)>=0
        yhat(i)=1;
    else
        yhat(i)=-1;
    end
end
end

%error
count = 0;
for i = 1:size(Xtest, 2)
if yhat(i,1) ~= ytest(i,1)
count = count+1;
end
end
test_error = count/size(X, 2);
```