Problem1

a. iris=load('data/iris.txt');
y=iris(:,end);
X=iris(:,1:end-1);
whos

EDU>> mean(X)

ans =

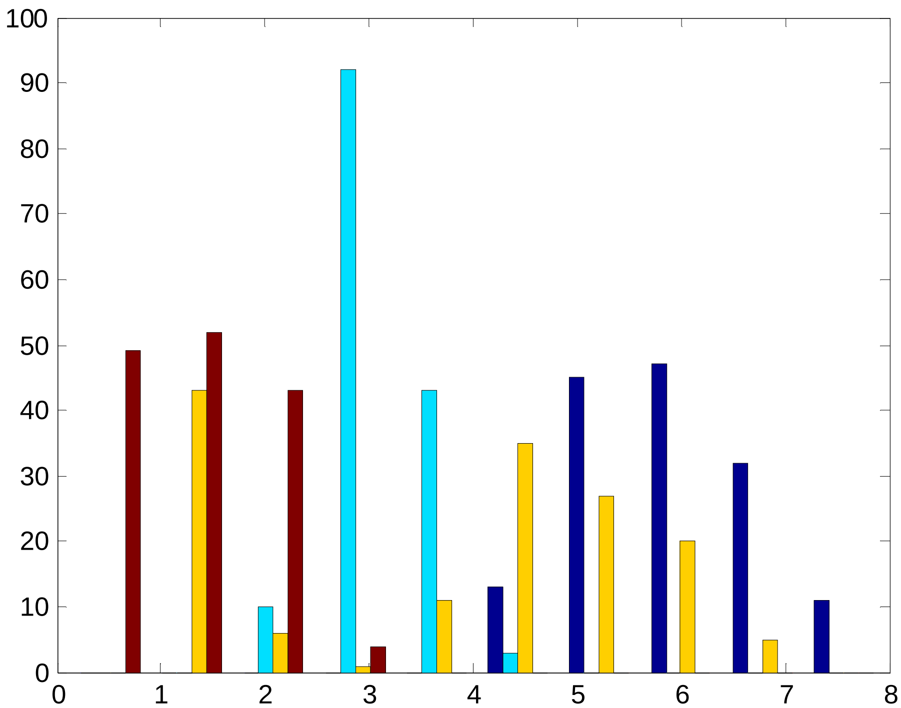
5.9001 3.0989 3.8196 1.2526

EDU>> var(X)

ans =

0.6993 0.1916 3.0976 0.5797

b. EDU>> hist(X)



c. EDU>> unique(y)

ans =

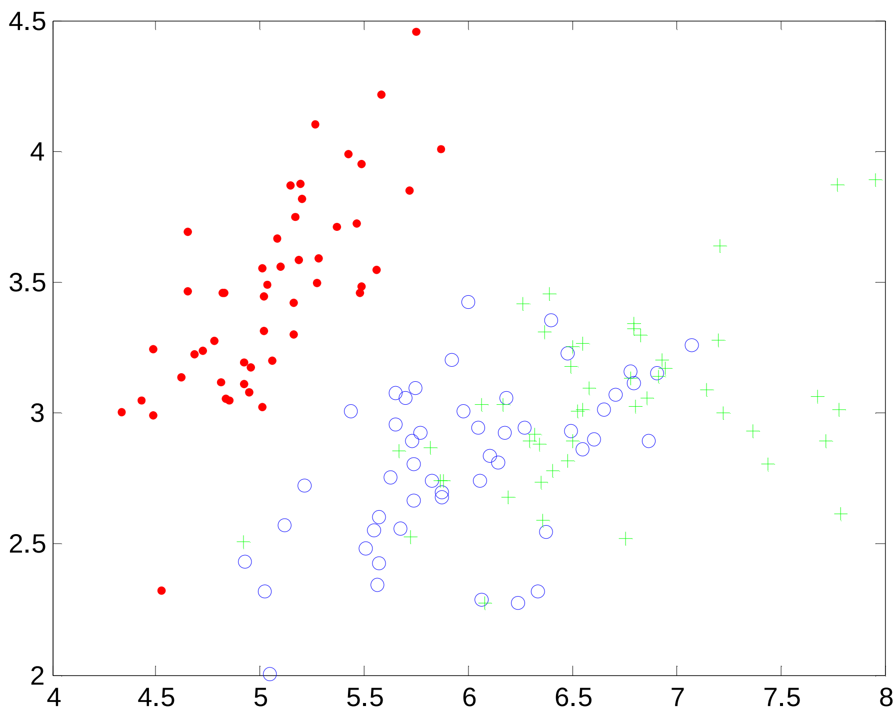
0

1

2

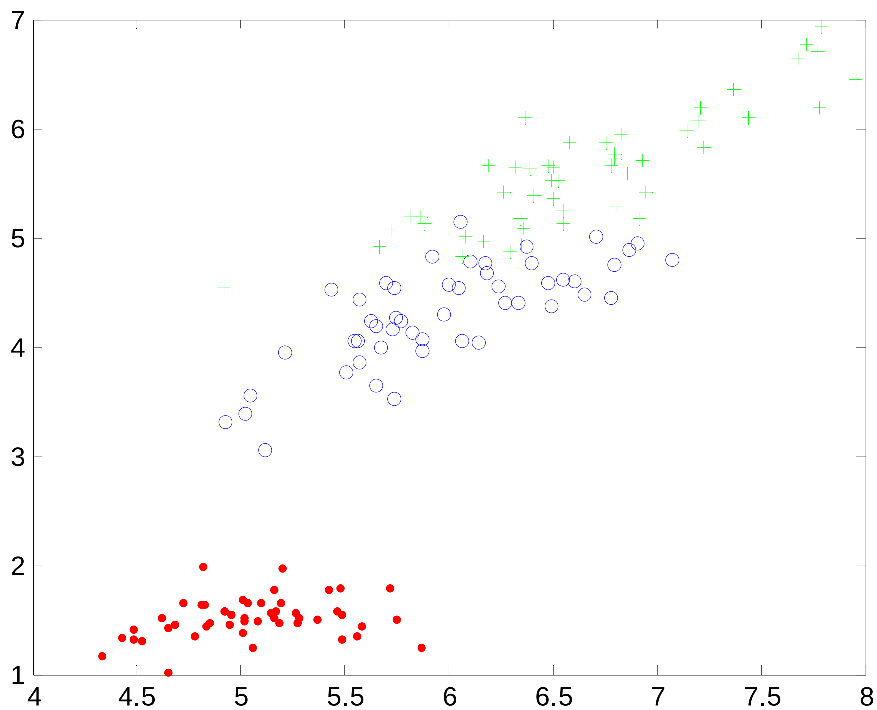
For the pair of features (1,2),

```
plot(X(find(y==0),1),X(find(y==0),2),'.r');
hold on;
plot(X(find(y==1),1),X(find(y==1),2),'ob');
hold on;
plot(X(find(y==2),1),X(find(y==2),2),'+g');
hold off;
```



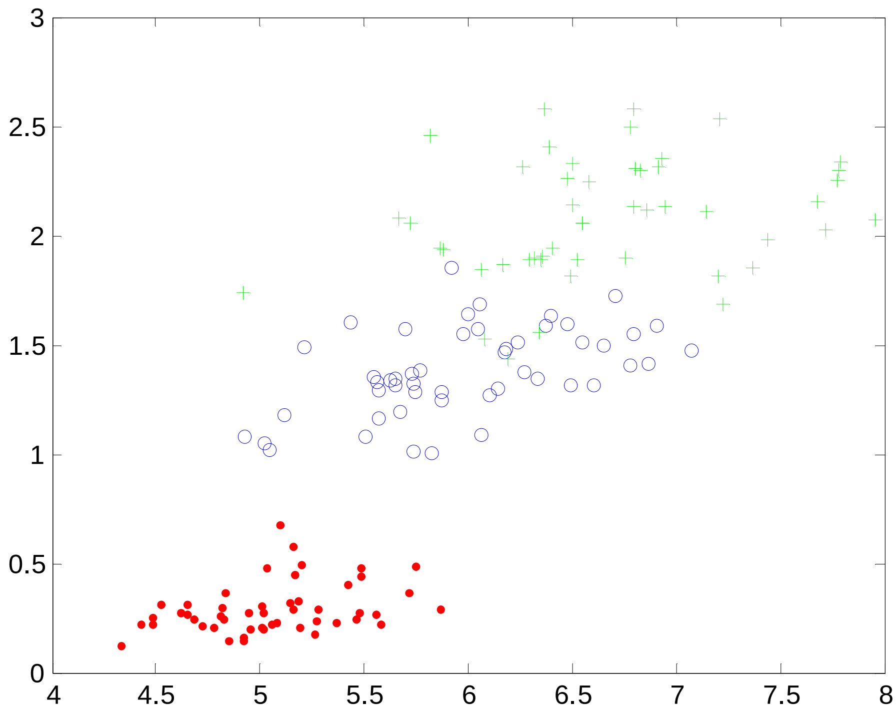
For the pair of features (1,3),

```
plot(X(find(y==0),1),X(find(y==0),3),'.r');
hold on;
plot(X(find(y==1),1),X(find(y==1),3),'ob');
hold on;
plot(X(find(y==2),1),X(find(y==2),3),'+g');
hold off;
```



For the pair of features (1,4),

```
plot(X(find(y==0),1),X(find(y==0),4),'.r');
hold on;
plot(X(find(y==1),1),X(find(y==1),4),'ob');
hold on;
plot(X(find(y==2),1),X(find(y==2),4),'+g');
hold off;
```

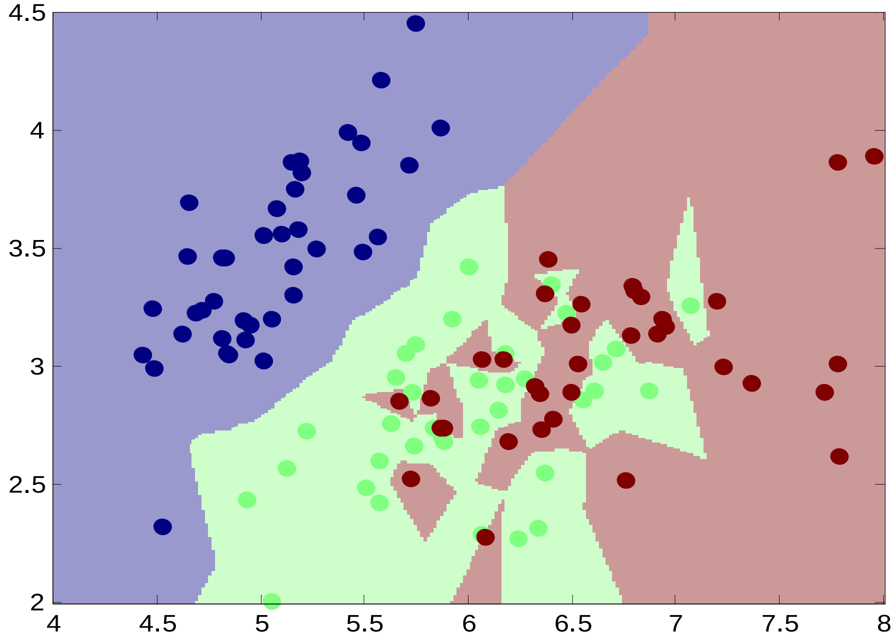


Problem2

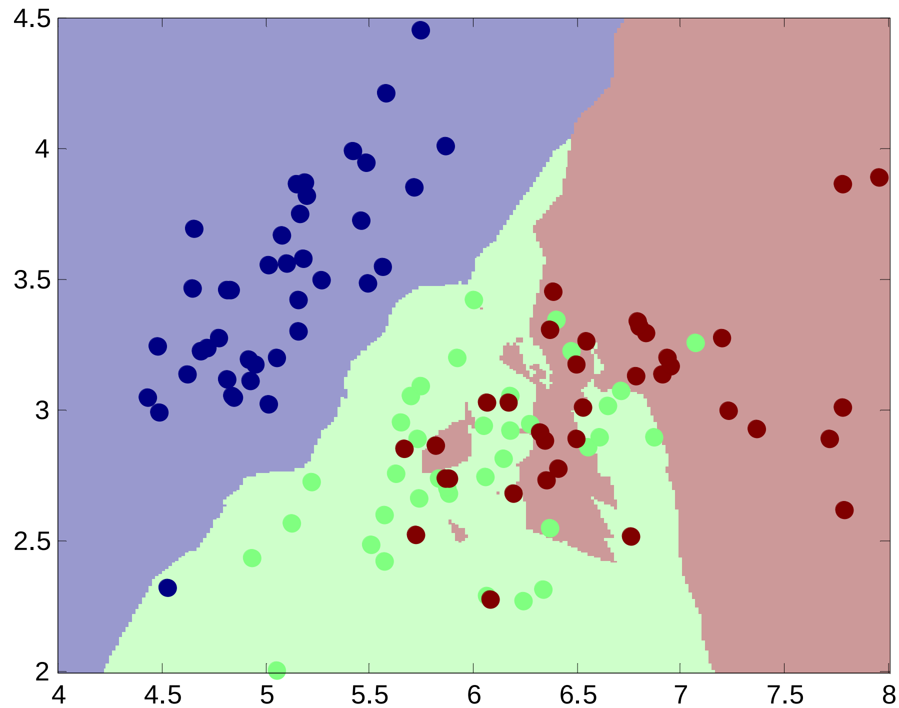
```
iris=load('data/iris.txt');
y=iris(:,end);
X=iris(:,1:end-1);
[X y] = reorderData(X,y);
[Xtr Xte Ytr Yte] = splitData(X,y,.75);
```

a. K =1,

```
knn = knnClassify(Xtr(:,1:2),Ytr,1);
plotClassify2D(knn,Xtr(:,1:2),Ytr);
```

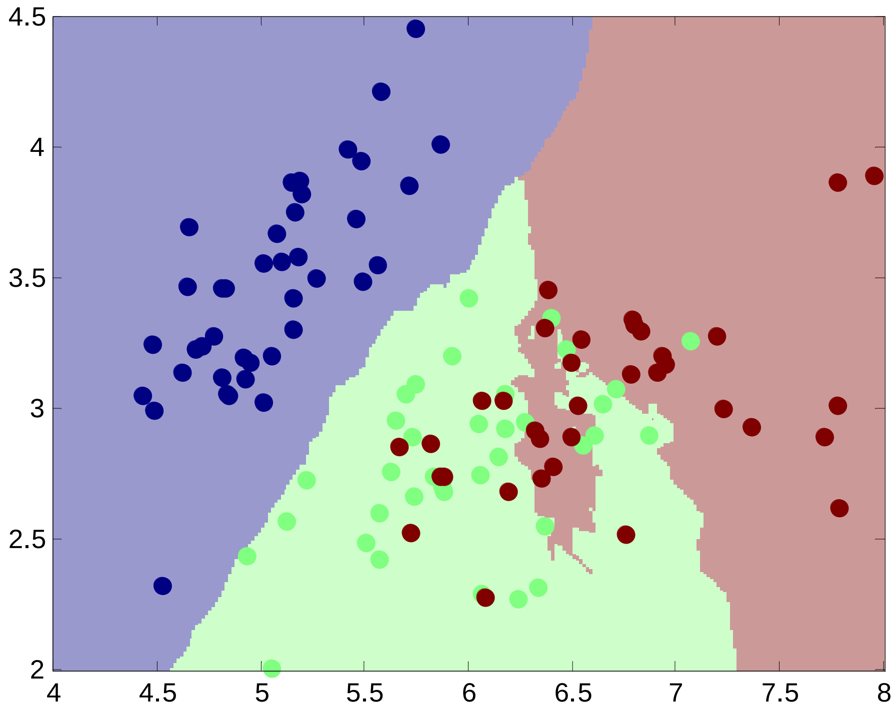


```
K = 5,
knn = knnClassify(Xtr(:,1:2),Ytr,5);
plotClassify2D(knn,Xtr(:,1:2),Ytr);
```

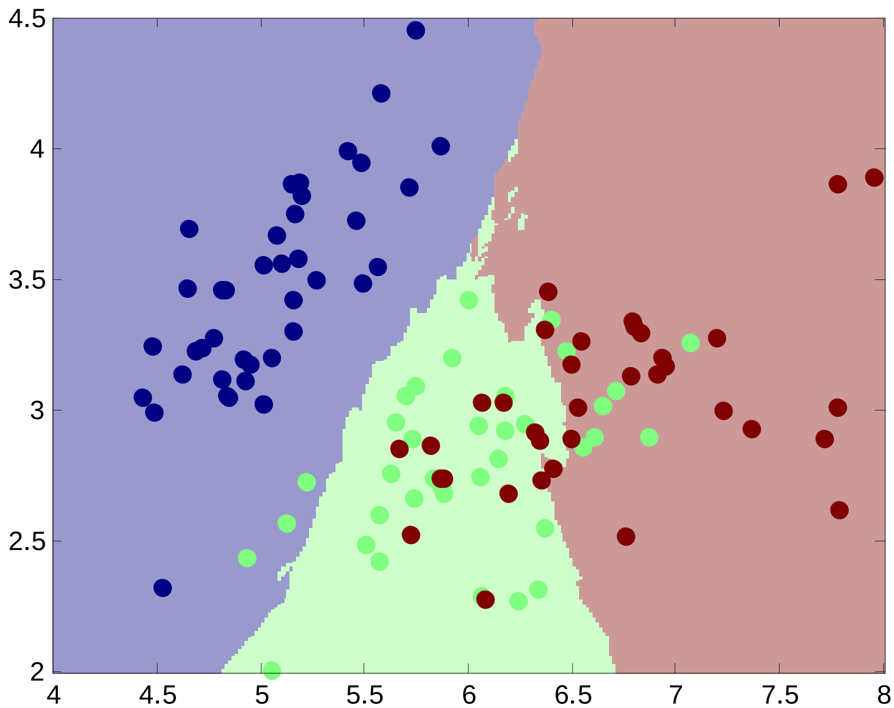


K =10,

```
knn = knnClassify(Xtr(:,1:2),Ytr,10);
plotClassify2D(knn,Xtr(:,1:2),Ytr);
```

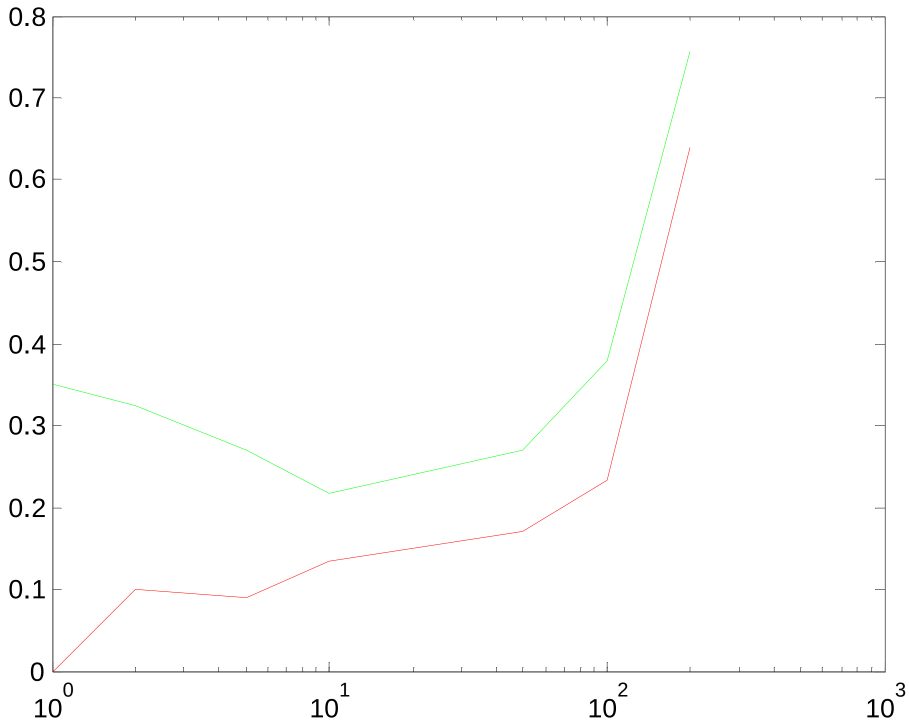


```
K = 50,
knn = knnClassify(Xtr(:,1:2),Ytr,50);
plotClassify2D(knn,Xtr(:,1:2),Ytr);
```



b.

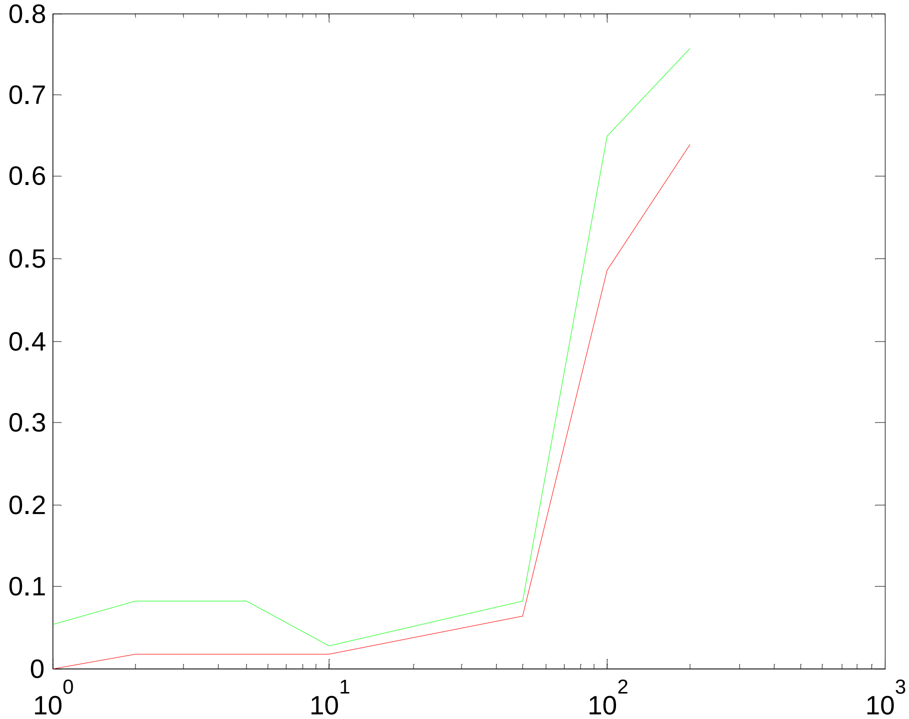
```
K=[1,2,5,10,50,100,200];
for k=1:length(K)
   learner = knnClassify(Xtr(:,1:2),Ytr,K(k));
   ete(k)=err(learner,Xte(:,1:2),Yte);
   etr(k)=err(learner,Xtr(:,1:2),Ytr);
end;
figure; semilogx(K,ete,'-g');
hold on;
semilogx(K,etr,'-r');
hold off;
```



Based on these plots, K = 10 is the best value.

c.

```
K=[1,2,5,10,50,100,200];
for k=1:length(K)
   learner = knnClassify(Xt,Ytr,K(k));
   ete(k)=err(learner,Xte,Yte);
   etr(k)=err(learner,Xtr,Ytr);
end;
figure; semilogx(K,ete,'-g');
hold on;
semilogx(K,etr,'-r');
hold off;
```

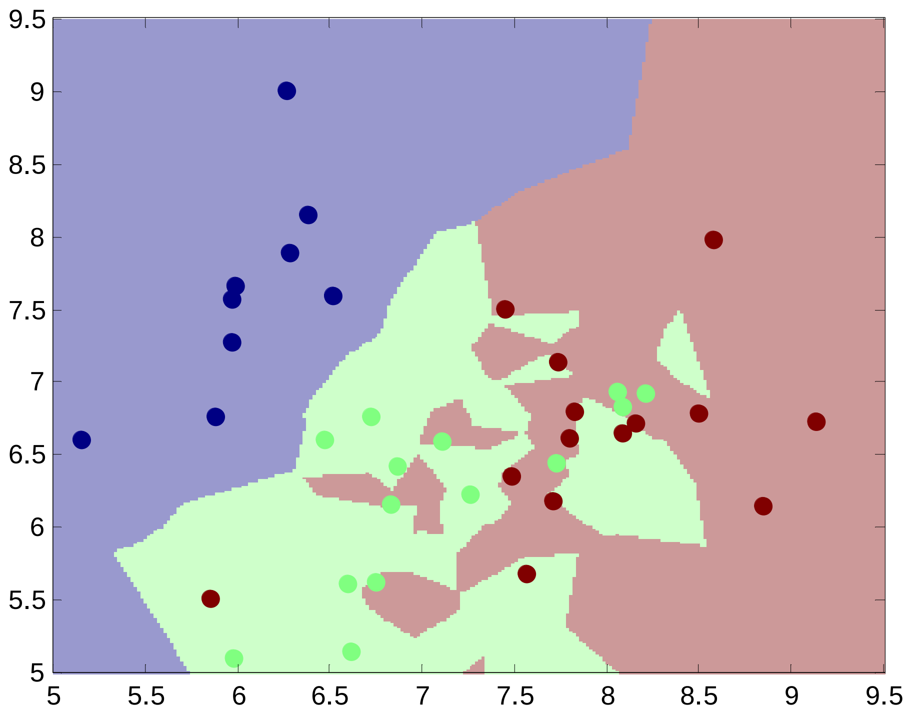


By comparing the figure in problem b and the figure in problem c, it shows that more features can improve accuracy.

d. (1) rescaling the data to unit variant.

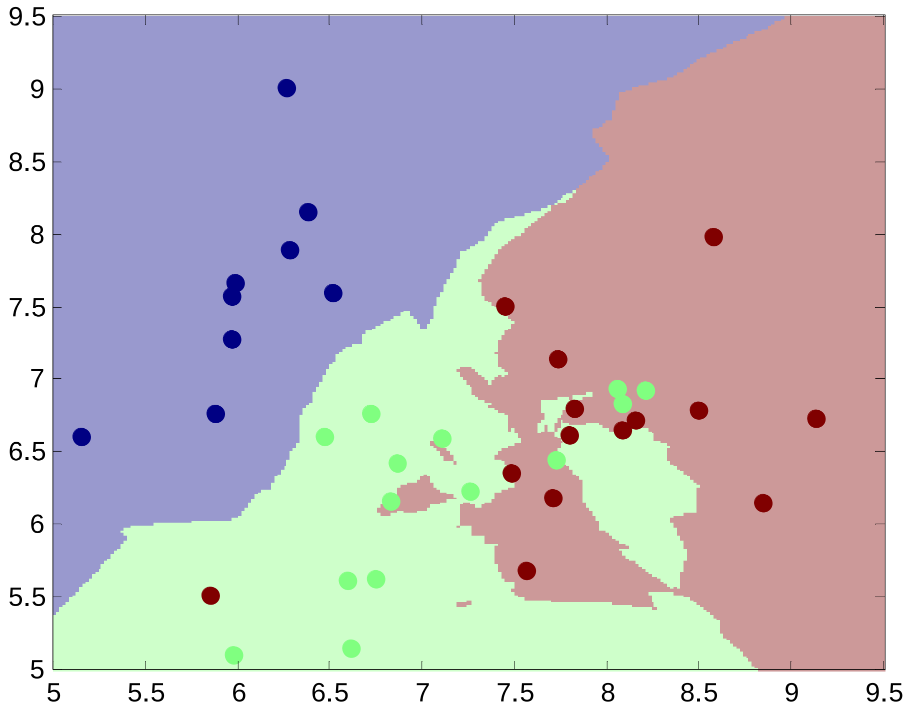
K=1,

```
[Xtr,T] = rescale(Xtr);
Xte=rescale(Xte,T);
knn = knnClassify(Xtr(:,1:2),Ytr,1);
plotClassify2D(knn,Xte(:,1:2),Yte);
```



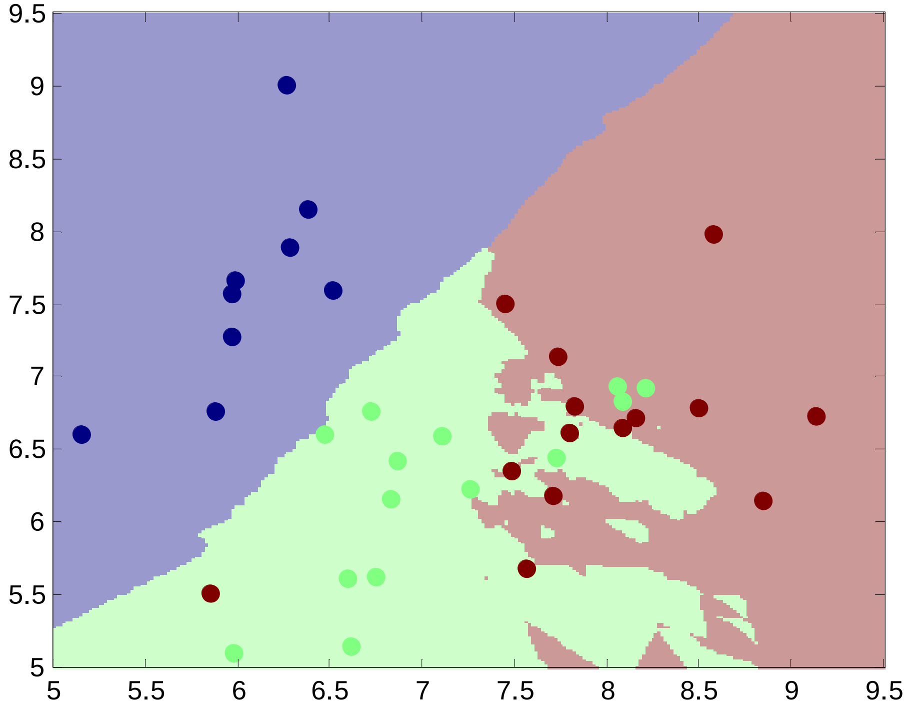
K=5,

```
[Xtr,T] = rescale(Xtr);
Xte=rescale(Xte,T);
knn = knnClassify(Xtr(:,1:2),Ytr,5);
plotClassify2D(knn,Xte(:,1:2),Yte);
```



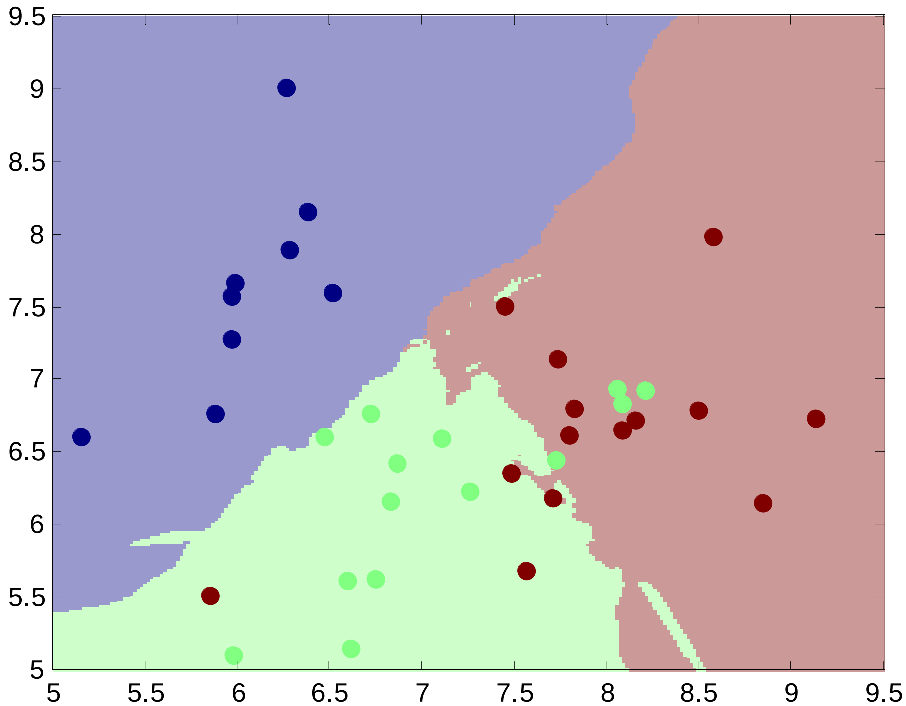
K=10,

```
[Xtr,T] = rescale(Xtr);
Xte=rescale(Xte,T);
knn = knnClassify(Xtr(:,1:2),Ytr,10);
plotClassify2D(knn,Xte(:,1:2),Yte);
```



K = 50,

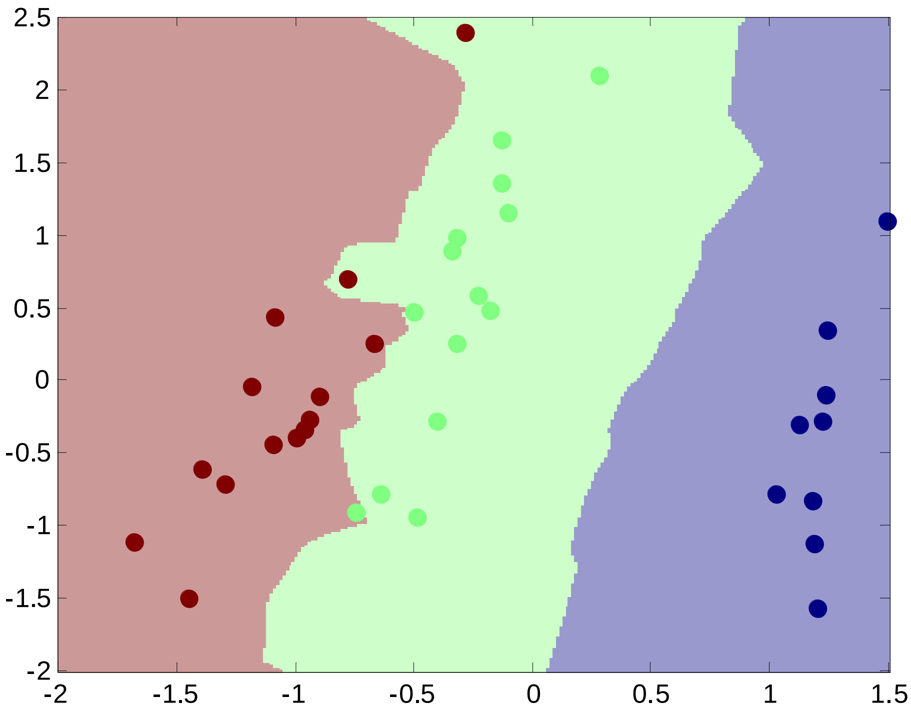
```
[Xtr,T] = rescale(Xtr);
Xte=rescale(Xte,T);
knn = knnClassify(Xtr(:,1:2),Ytr,50);
plotClassify2D(knn,Xte(:,1:2),Yte);
```



Comment: Rescale makes data become more center.

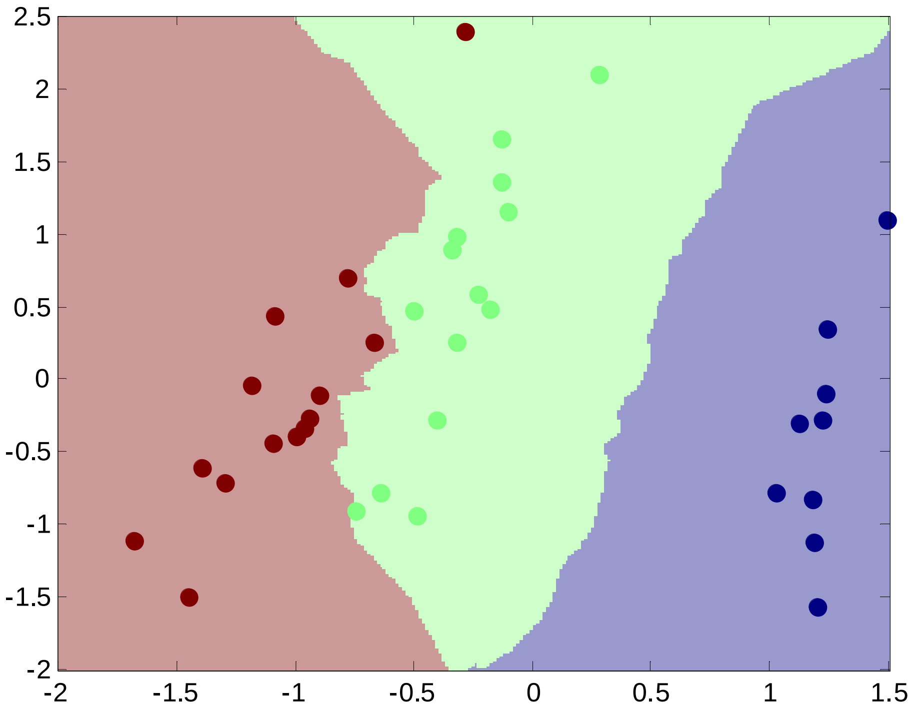
(2) whiten
K = 1,

```
[Xtr,m,s] = whiten(Xtr);
Xte=whiten(Xte,m,s);
knn = knnClassify(Xtr(:,1:2),Ytr,1);
plotClassify2D(knn,Xte(:,1:2),Yte);
```



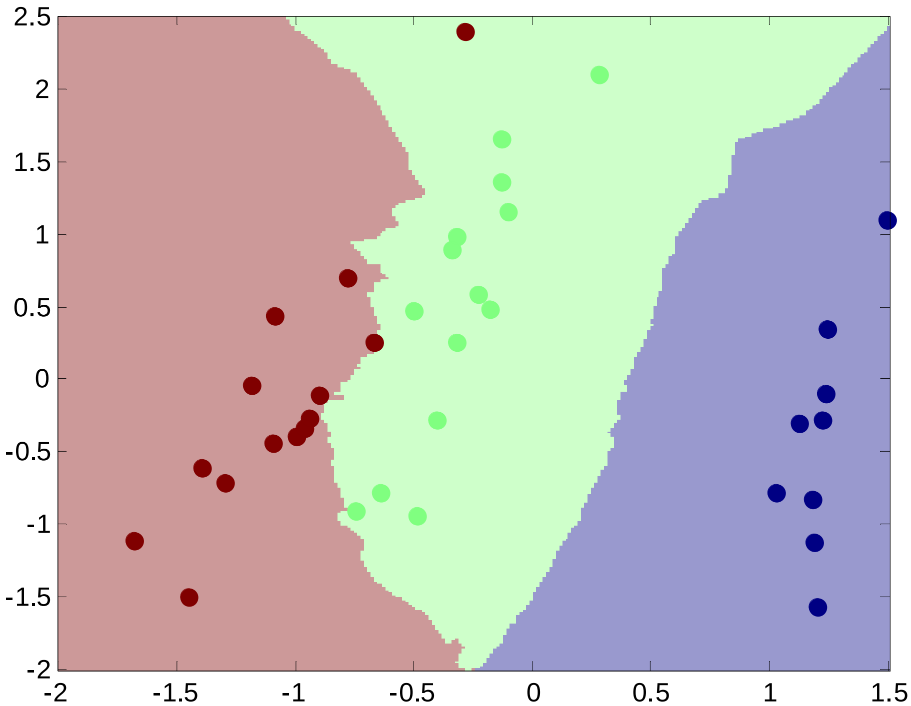
K = 5,

```
[Xtr,m,s] = whiten(Xtr);
Xte=whiten(Xte,m,s);
knn = knnClassify(Xtr(:,1:2),Ytr,5);
plotClassify2D(knn,Xte(:,1:2),Yte);
```



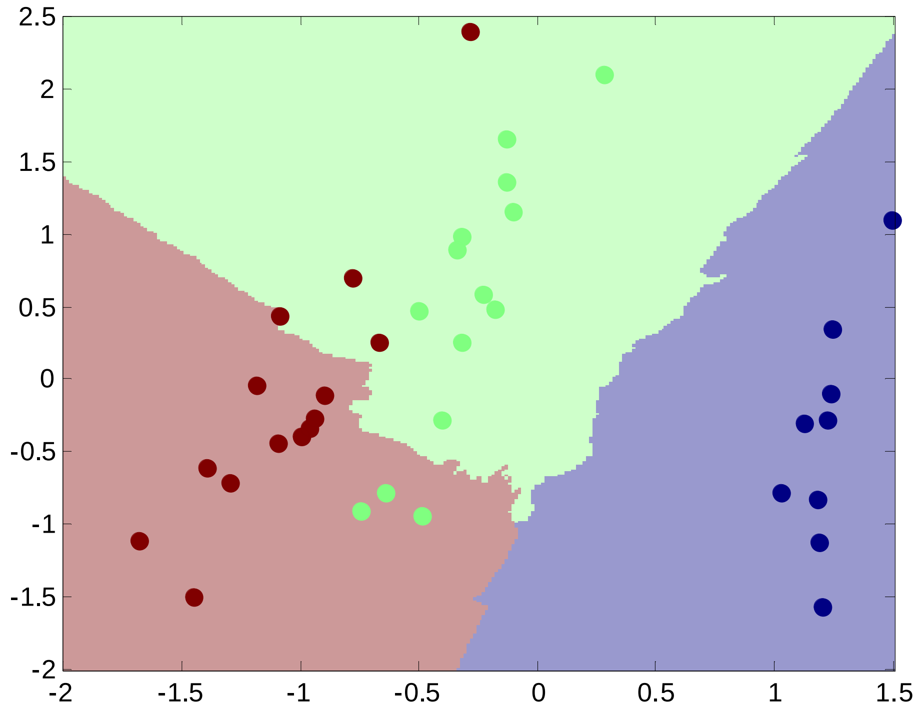
K = 10,

```
[Xtr,m,s] = whiten(Xtr);
Xte=whiten(Xte,m,s);
knn = knnClassify(Xtr(:,1:2),Ytr,10);
plotClassify2D(knn,Xte(:,1:2),Yte);
```



K = 50,

```
[Xtr,m,s] = whiten(Xtr);
Xte=whiten(Xte,m,s);
knn = knnClassify(Xtr(:,1:2),Ytr,50);
plotClassify2D(knn,Xte(:,1:2),Yte);
```

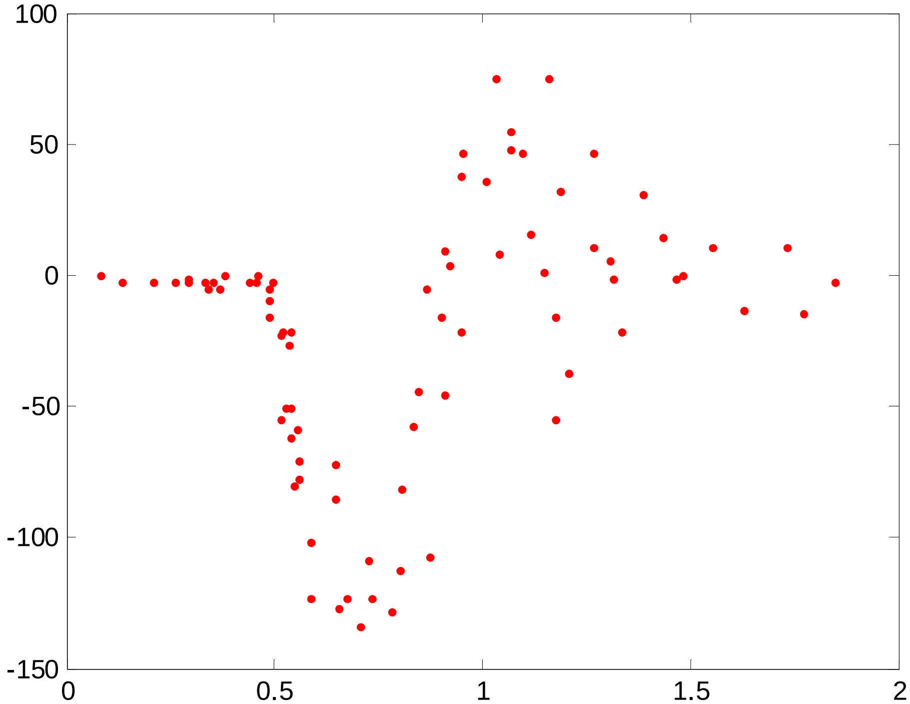


Comment: whiten makes data distribute around 0, more center, uncorrelated with each other.

Problem3

a.

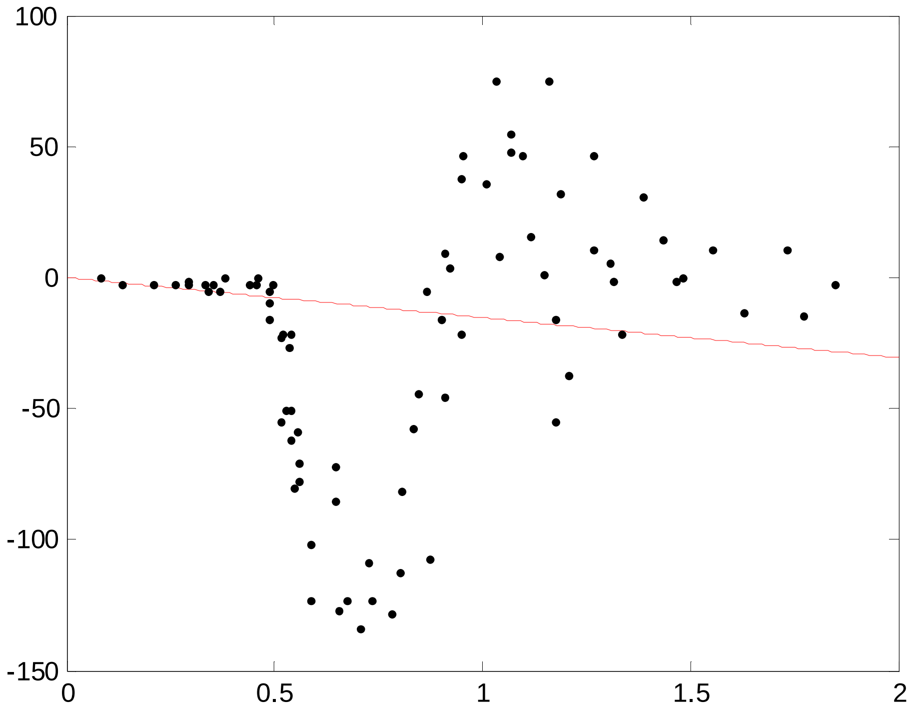
```
Tr=load('data/mcycleTrain.txt');
Te=load('data/mcycleTest.txt');
y = Tr(:,1);
x = Tr(:,2);
plot(x,y,'.r');
```



Comment: It looks like a polynomial function.

b.

```
Xtr = Tr(:,2);
Ytr = Tr(:,1);
lr = linearRegress(Xtr,Ytr);
xs = 0:.01:2;
ys = predict(lr,xs);
plot(xs,ys,'-r');
hold on;
plot(Xtr,Ytr,'.k');
hold off;
```

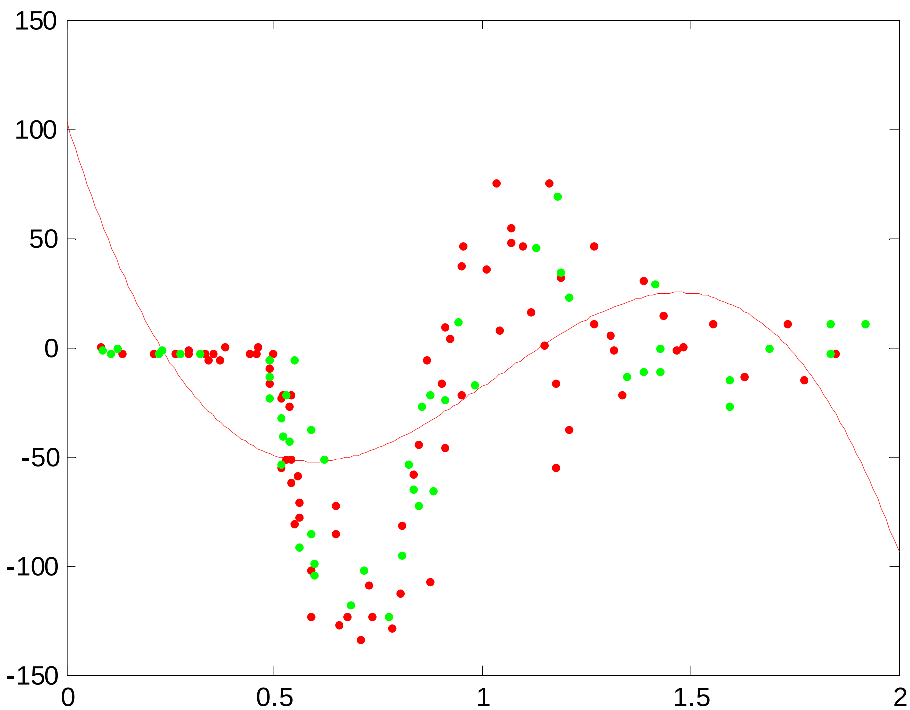



```
Ytr = Tr(:,1); Xtr = Tr(:,2);
lr = linearRegress(Xtr,Ytr);
Jtr = mse(lr,Xtr,Ytr)
Yte = Te(:,1);Xte = Te(:,2);
Jte = mse(lr,Xte,Yte)

Jtr =

   2.7978e+03

Jte =

   2.2504e+03
```

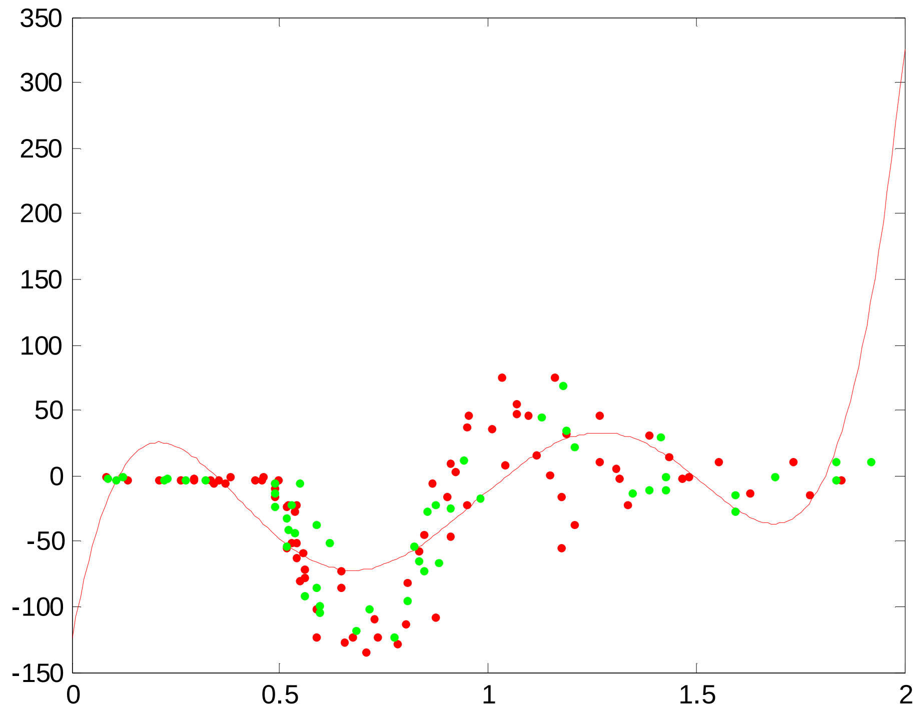
c. (1)

```
function polynomialFunction(degree)
Tr=load('data/mcycleTrain.txt');
Xtr = Tr(:,2);
Ytr = Tr(:,1);
XtrP = fpoly(Xtr,degree);
[XtrP,T] = rescale(XtrP);
lr = linearRegress(XtrP,Ytr);
xs = [0:.01:2]';
xsP = fpoly(xs,degree);
xsP = rescale(xsP,T);
ys = predict(lr,xsP);
plot(xs,ys,'-r');
hold on;
plot(Xtr,Ytr,'.k');
hold off;
```

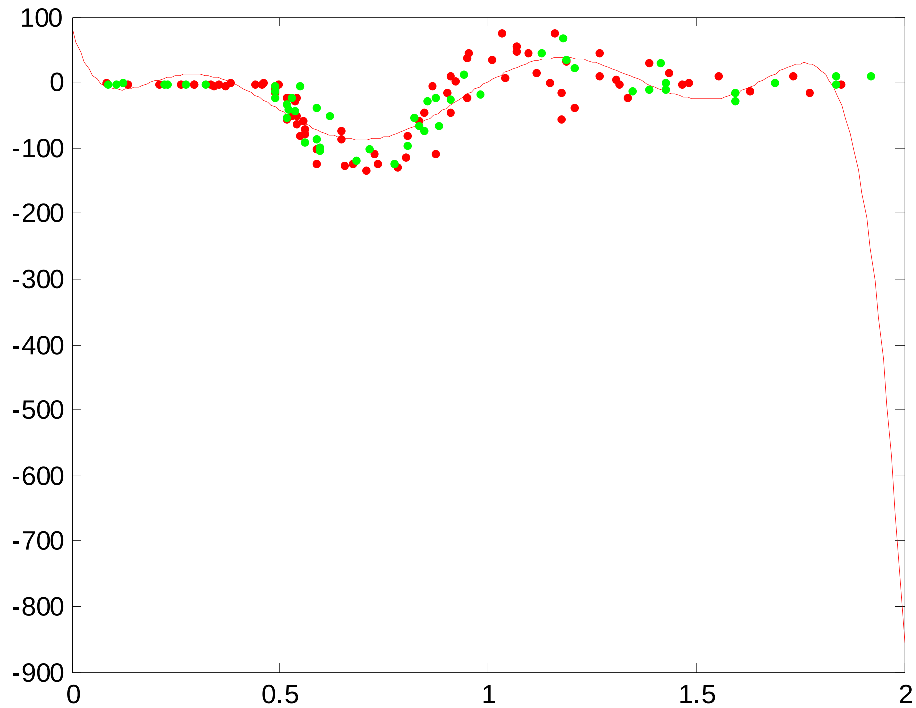
Degree = 3,



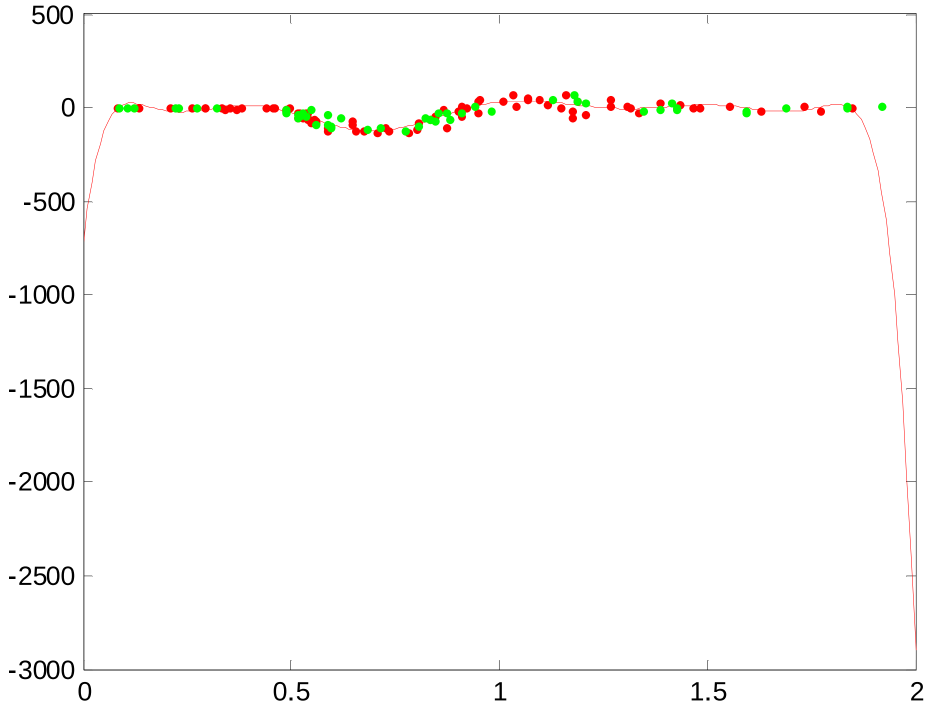
Degree = 5,



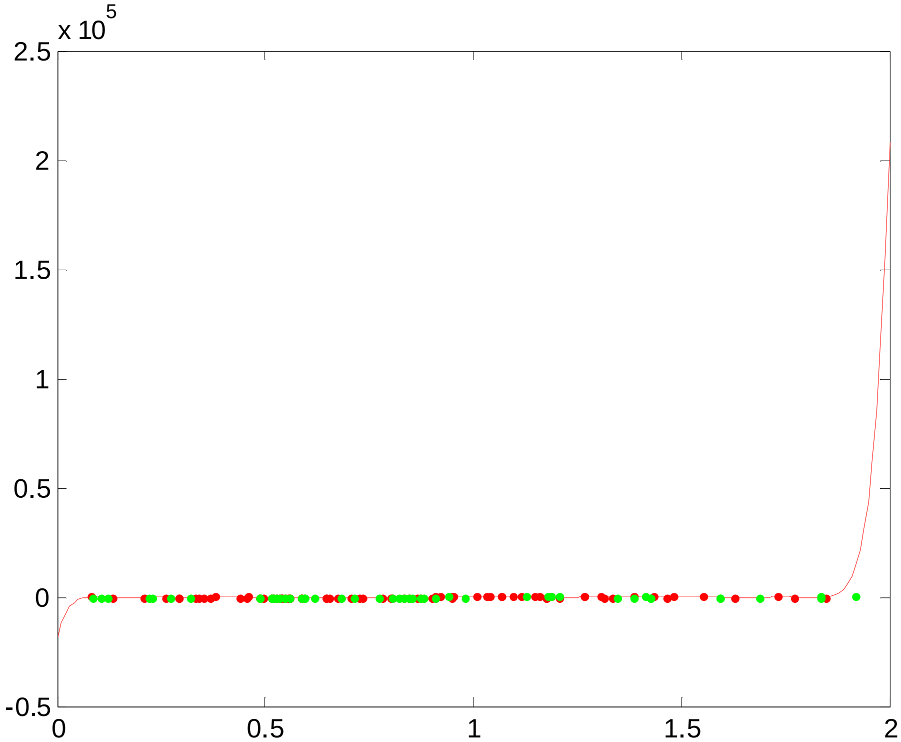
Degree = 7



Degree = 10,

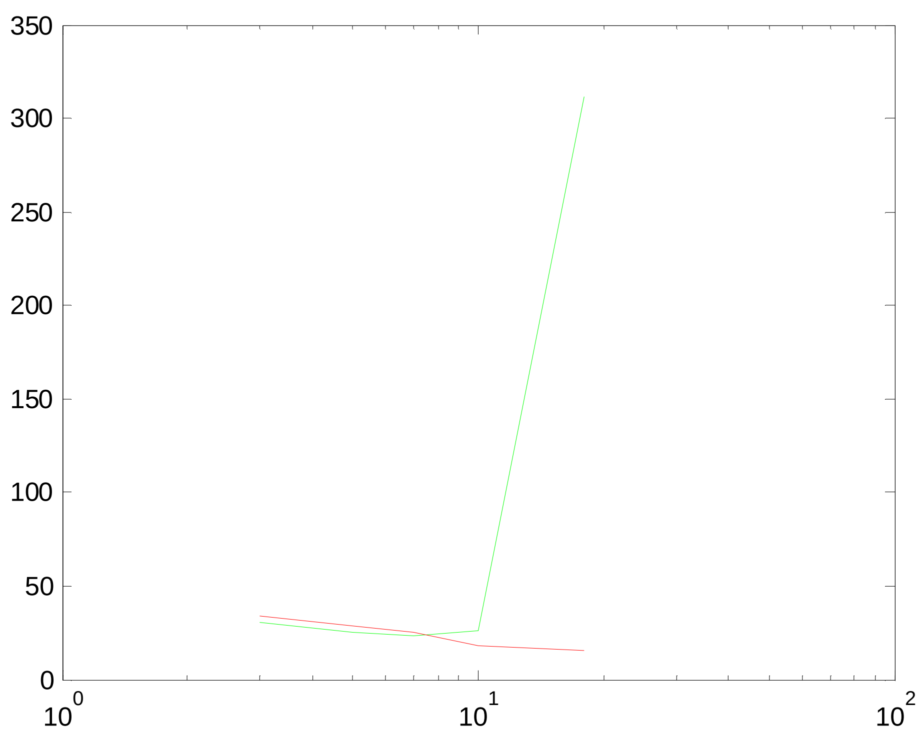


Degree = 18,



(2)

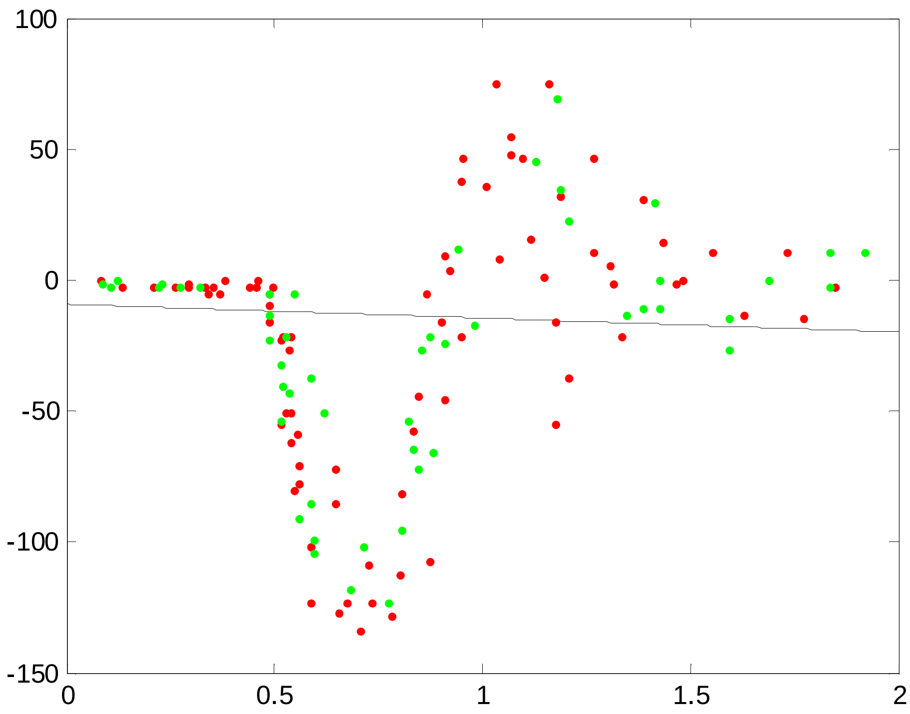
```
Tr=load('data/mcycleTrain.txt'); Te=load('data/mcycleTest.txt');

Xtr = Tr(:,2);
Ytr = Tr(:,1);
Xte = Te(:,2);
Yte = Te(:,1);
D=[3,5,7,10,18];
for d=1:length(D)
   XtrP = fpoly(Xtr,D(d));
   [XtrP,T] = rescale(XtrP);
   learner = linearRegress(XtrP,Ytr);
   XteP = fpoly(Xte,D(d));
   XteP = rescale(XteP,T);
   ete(d)=mae(learner,XteP,Yte);
   etr(d)=mae(learner,XtrP,Ytr);
end;
figure; semilogx(D,ete,'-g');
hold on;
semilogx(D,etr,'-r');
hold off;
```



d.

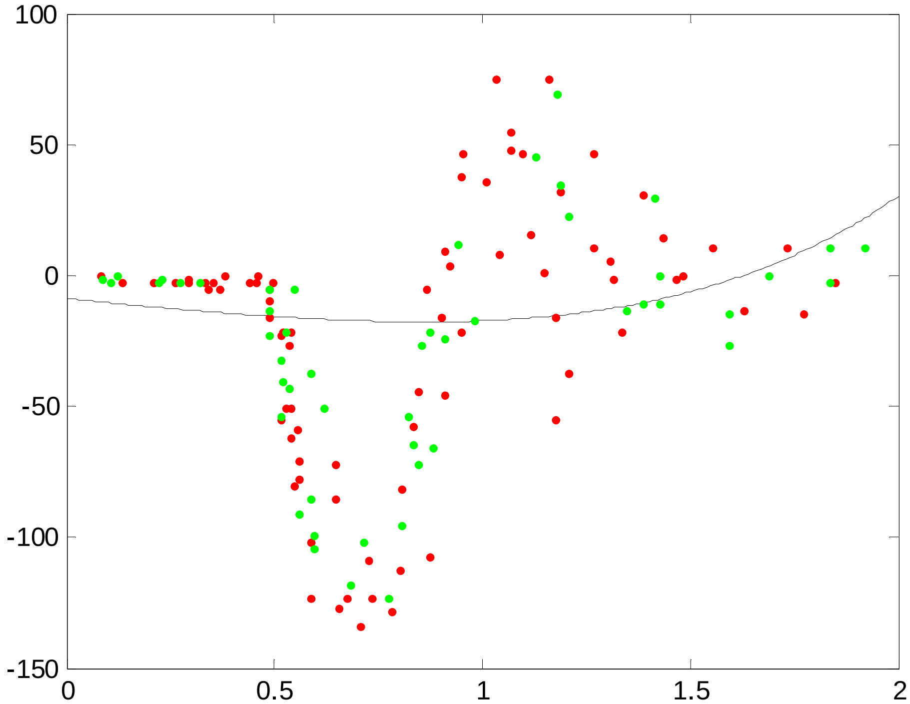
```
function regularization(degree, reg)
Xtr = Tr(:,2);
Ytr = Tr(:,1);
Xte = Te(:,2);
Yte = Te(:,1);
XtrP = fpoly(Xtr,degree); %degree
[XtrP,T] = rescale(XtrP);
lr = linearRegress(XtrP,Ytr,reg);%reg
xs = [0:.01:2]';
xsP =fpoly(xs,degree); %degree
xsP = rescale(xsP,T);
ys = predict(lr,xsP);
plot(xs,ys,'-k');
hold on;
plot(Xtr,Ytr,'.r');
hold on;
plot(Xte,Yte,'.g');
hold off;
```

degree =1, reg =1,

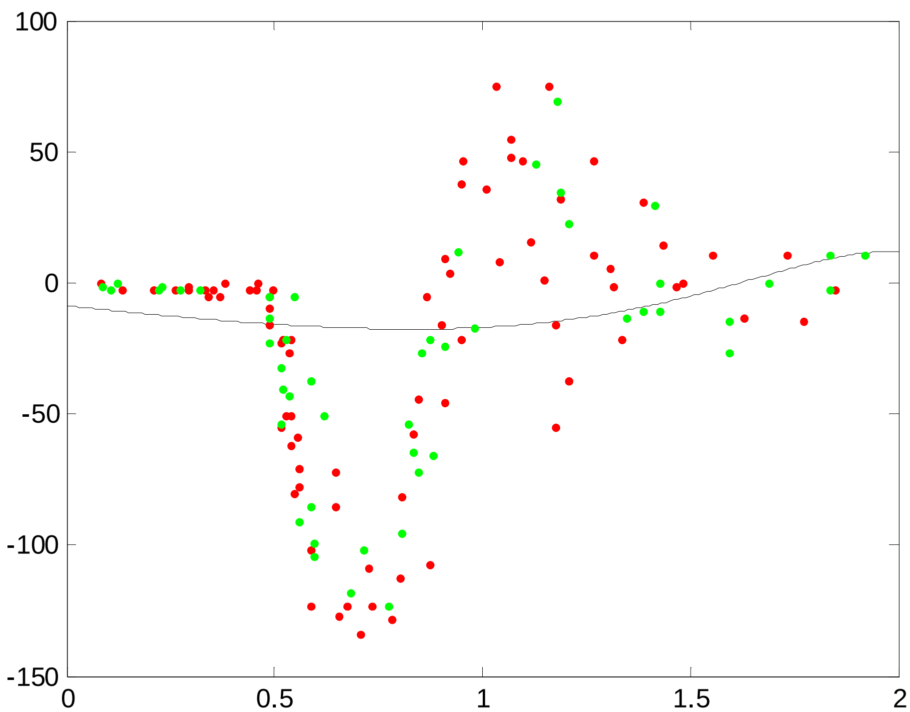


Comment: there is no much difference with the one without regulating.

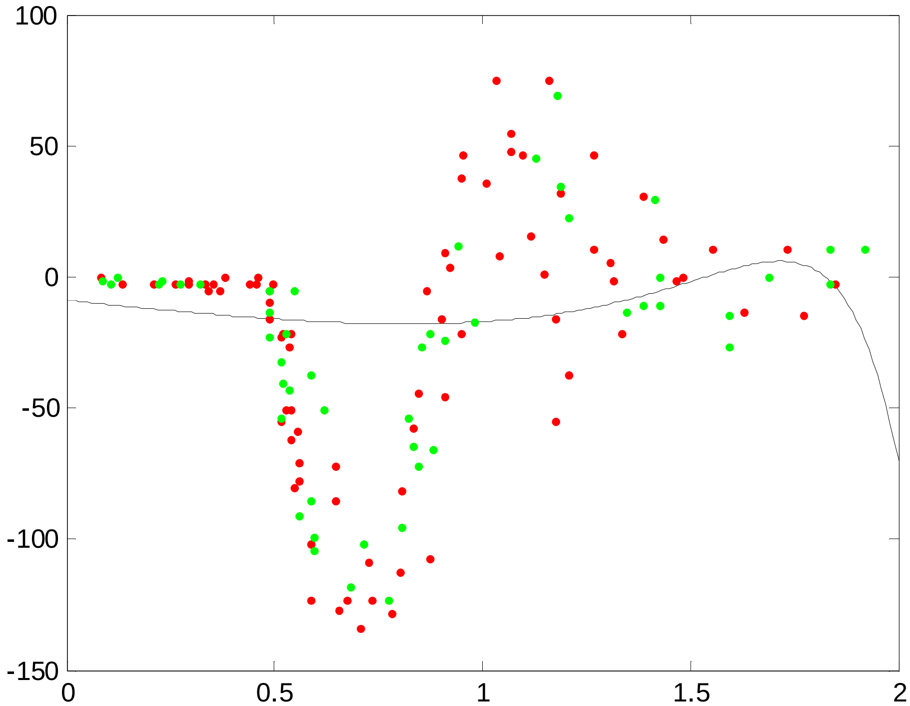
degree = 5, reg =1,



Degree =10, reg = 1,



Degree =18, reg =1,



Comment: After regulating, data is less important than the one without regulating. The function become more smooth and more practical, at least might not be over-fitting as the one without regulating.