**模式识别报告L6，7**

自卓2201 杨欣怡 U202215067

1，利用二次规划函数，分别编程实现原问题求解的支撑向量机算法（Primal-SVM）、对偶的支撑向量机算法（Dual-SVM）、和核函数的支撑向量机算法（Kernel-SVM）。

Primal-SVM：

1. %% Primal-SVM
2. z=zeros(1,2);
3. Q=[0 z;z.' eye(2)];
4. p=zeros(1,2+1);
5. A1=zeros(3,(n+n)\*0.8);   %(n+n)\*0.8为训练样本个数
6. for i=1:3
7. A1(i,:)=trainData(i,:).\*trainLabel;
8. end
9. A=-1\*A1';     % 二次规划函数约束条件是<=
10. c=(zeros(1,2\*n\*0.8)-1).';
11. u=quadprog(Q,p,A,c);
12. figure(1);
13. plot(X1(2,:),X1(3,:),'rx',X2(2,:),X2(3,:),'b\*');
14. X=-5:0.2:5;
15. Y=-1\*(u(2,:)\*X+u(1,:))/u(3,:);   %画分类线
16. hold on;
17. plot(X,Y,'m-','LineWidth',2);
18. title('Primal-SVM');
19. legend('“+1”类','“-1”类','分类面');
20. %分类正确率
21. num=0;
22. train\_num=size(trainData,2);
23. for i=1:train\_num
24. if(sign(u'\*trainData(:,i)) ~= trainLabel(:,i))
25. num=num+1;
26. end
27. end
28. fprintf("Primal-SVM训练集正确率为%f\n",1-num/train\_num);
29. num=0;
30. test\_num=size(testData,2);
31. for i=1:test\_num
32. if(sign(u'\*testData(:,i)) ~= testLabel(:,i))
33. num=num+1;
34. end
35. end
36. fprintf("Primal-SVM测试集正确率为%f\n",1-num/test\_num);

Dual-SVM:

1. %% Primal-SVM
2. z=zeros(1,2);
3. Q=[0 z;z.' eye(2)];
4. p=zeros(1,2+1);
5. A1=zeros(3,(n+n)\*0.8);   %(n+n)\*0.8为训练样本个数
6. for i=1:3
7. A1(i,:)=trainData(i,:).\*trainLabel;
8. end
9. A=-1\*A1';     % 二次规划函数约束条件是<=
10. c=(zeros(1,2\*n\*0.8)-1).';
11. u=quadprog(Q,p,A,c);
12. figure(1);
13. plot(X1(2,:),X1(3,:),'rx',X2(2,:),X2(3,:),'b\*');
14. X=-5:0.2:5;
15. Y=-1\*(u(2,:)\*X+u(1,:))/u(3,:);   %画分类线
16. hold on;
17. plot(X,Y,'m-','LineWidth',2);
18. title('Primal-SVM');
19. legend('“+1”类','“-1”类','分类面');
20. %分类正确率
21. num=0;
22. train\_num=size(trainData,2);
23. for i=1:train\_num
24. if(sign(u'\*trainData(:,i)) ~= trainLabel(:,i))
25. num=num+1;
26. end
27. end
28. fprintf("Primal-SVM训练集正确率为%f\n",1-num/train\_num);
29. num=0;
30. test\_num=size(testData,2);
31. for i=1:test\_num
32. if(sign(u'\*testData(:,i)) ~= testLabel(:,i))
33. num=num+1;
34. end
35. end
36. fprintf("Primal-SVM测试集正确率为%f\n",1-num/test\_num);

Kernel-SVM:

1. %% Kernel-SVM（高斯核函数）
2. N=size(trainData\_K,2);
3. Q=zeros(N,N);
4. for i=1:N
5. for j=1:N
6. Q(i,j)=trainLabel\_K(1,i)\*trainLabel\_K(1,j)\*exp(-sum((trainData\_K(:,i)-trainData\_K(:,j)).^2,1)).';
7. end
8. end
9. p=zeros(1,N)-1;
10. a1=eye(N);
11. a2=trainLabel\_K;
12. a3=trainLabel\_K\*-1;
13. A=[a1;a2;a3];
14. A=-1\*A;
15. c=zeros(N+2,1);
16. u=quadprog(Q,p,A,c);
17. index=find(u > 1e-4);
18. svLabel=trainLabel\_K(1,index)';
19. svData=trainData\_K(:,index);
20. xx = linspace(min(trainData\_K(1,:)),max(trainData\_K(1,:)),50);
21. yy = linspace(min(trainData\_K(2,:)),max(trainData\_K(2,:)),50);
22. [X,Y] = meshgrid(xx,yy);   %生成网格
23. Z=zeros(50,50);
24. alpha=u(index);   %支撑向量的系数
25. k=sum((svData-svData(:,1)).^2,1);
26. b=svLabel(1)-(alpha.\*svLabel)'\*exp(-sum((svData-svData(:,1)).^2,1)).';
27. for i=1:50
28. for j=1:50
29. Z(i,j)=(alpha.\*svLabel).'\*exp(-sum((svData-[X(i,j);Y(i,j)]).^2,1)).'+b;
30. end
31. end
32. figure(4);
33. plot(X1(2,:),X1(3,:),'rx',X2(2,:),X2(3,:),'b\*');
34. hold on;
35. plot(svData(1,:),svData(2,:),'go','LineWidth', 2,'markersize', 10);   % 支撑向量
36. hold on;
37. contour(X,Y,Z,'LineWidth', 1);
38. title('Kernel-SVM（高斯核函数）');
39. legend("+1”类","“-1”类","支撑向量","分类面");
40. %分类正确率
41. num=0;
42. train\_num=size(trainData\_K,2);
43. for i=1:train\_num
44. if(sign((alpha.\*svLabel)'\*(0.5+0.5\*svData'\*trainData\_K(:,i)).^4+b) ~= trainLabel\_K(:,i))
45. num=num+1;
46. end
47. end
48. fprintf("Kernel-SVM（高斯核函数）训练集正确率为%f\n",1-num/train\_num);
49. num=0;
50. test\_num=size(testData\_K,2);
51. for i=1:test\_num
52. if(sign((alpha.\*svLabel)'\*(0.5+0.5\*svData'\*testData\_K(:,i)).^4+b) ~= testLabel\_K(:,i))
53. num=num+1;
54. end
55. end
56. fprintf("Kernel-SVM（高斯核函数）测试集正确率为%f\n",1-num/test\_num);

2，（a）产生两个都具有200个二维向量的数据集和。数据集的样本来自均值向量协方差矩阵的正态分布，属于“+1”类，数据集的样本来自均值向量、协方差矩阵的正态分布，属于“-1”类，其中****是一个2\*2的单位矩阵。产生的数据中80%用于训练，20%用于测试。

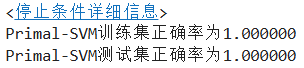
（b）在上述数据集上分别运用Primal-SVM、Dual-SVM和Kernel-SVM算法，利用产生的训练样本集得到分类面，其中，Kernel-SVM中的核函数分别采用四次多项式和高斯核函数，算法中用到的各类超参数自定。

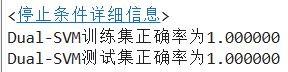
（c）分别在训练集和测试集上统计分类正确率。

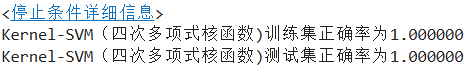
（d）对于Dual-SVM和Kernel-SVM算法，指出哪些样本是支撑向量

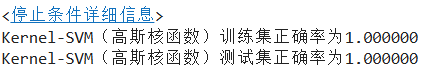
（e）画出数据集和分类面、间隔面，并标注出哪些样本是支撑向量，观察是否有边界上的向量不是支撑向量的现象。

（c）

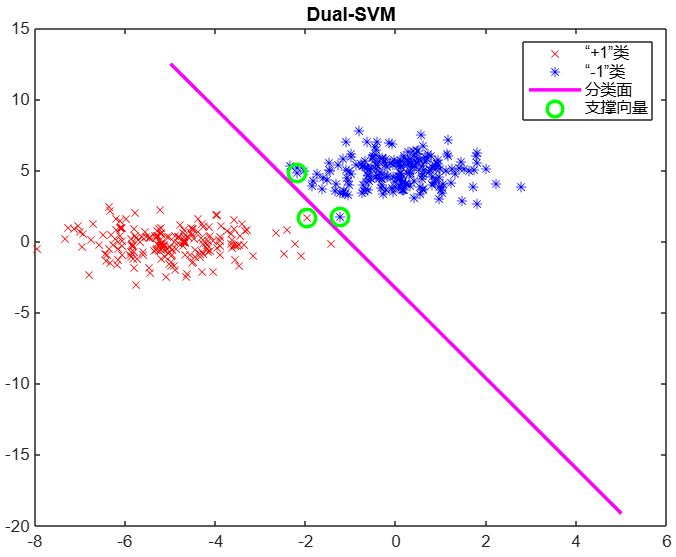
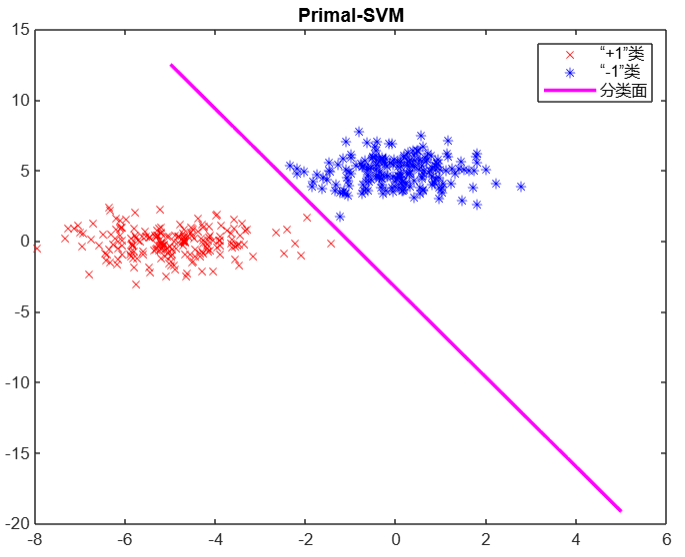


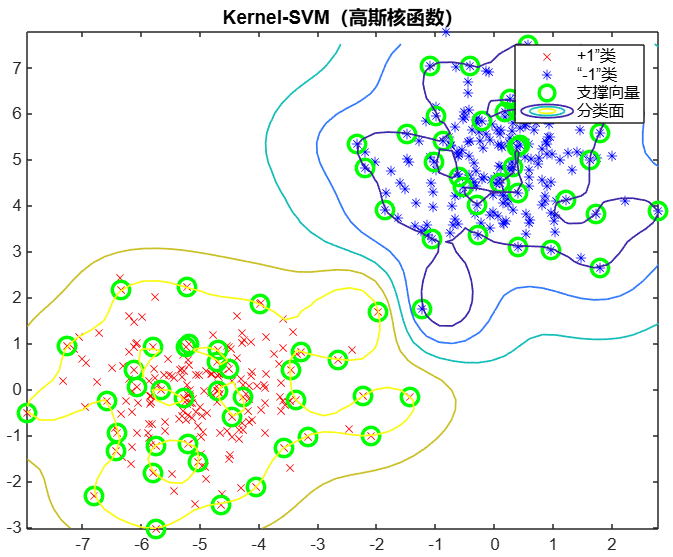
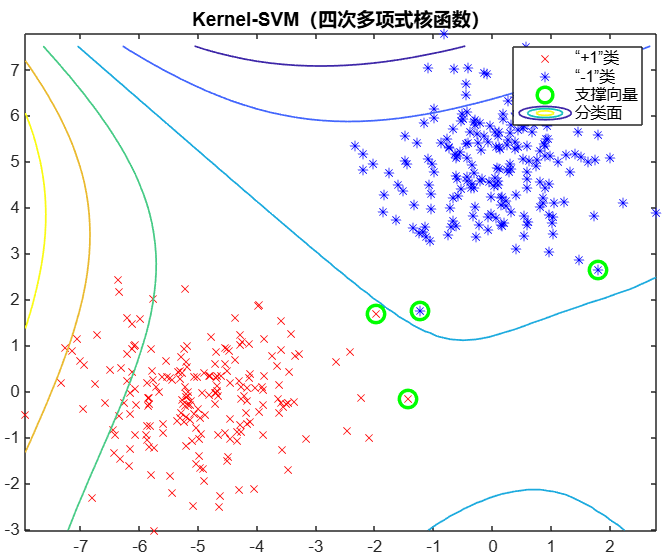






（e）





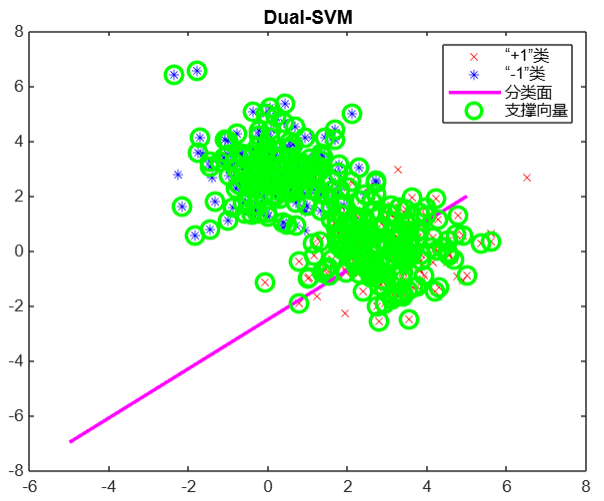
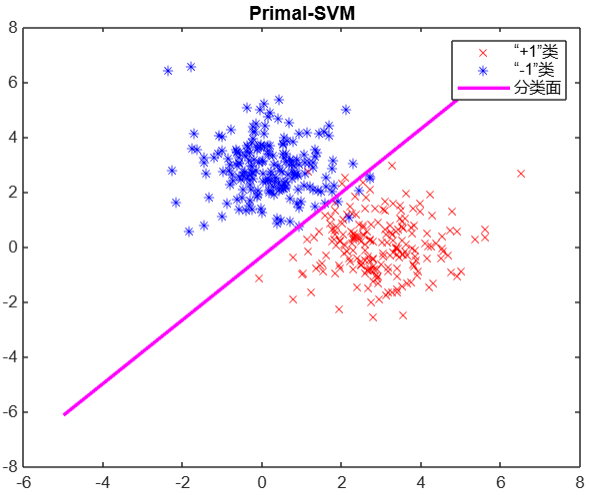
3，重复第2题的内容，但数据集和数据集的均值向量分别改为和，其他不变。

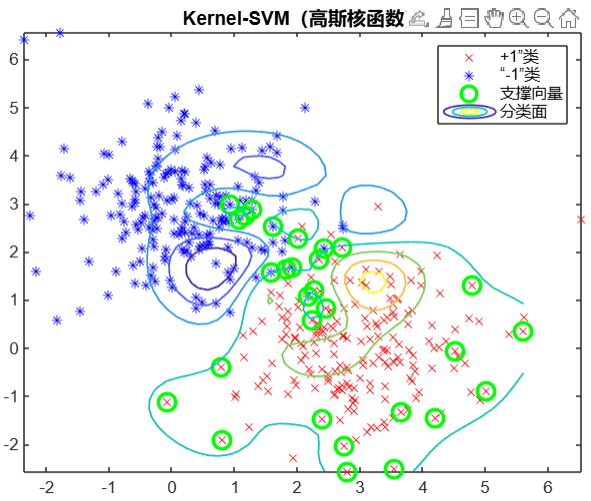
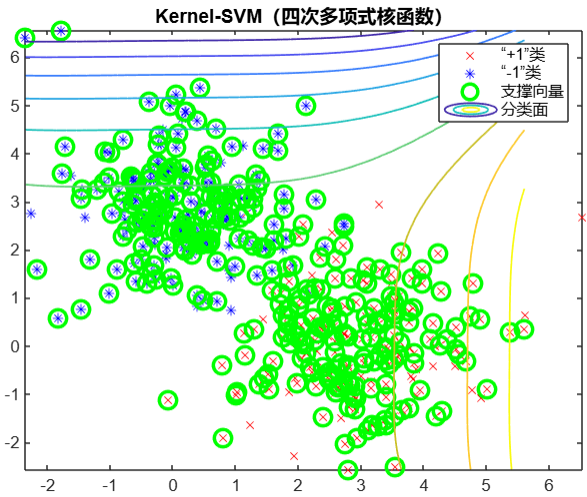










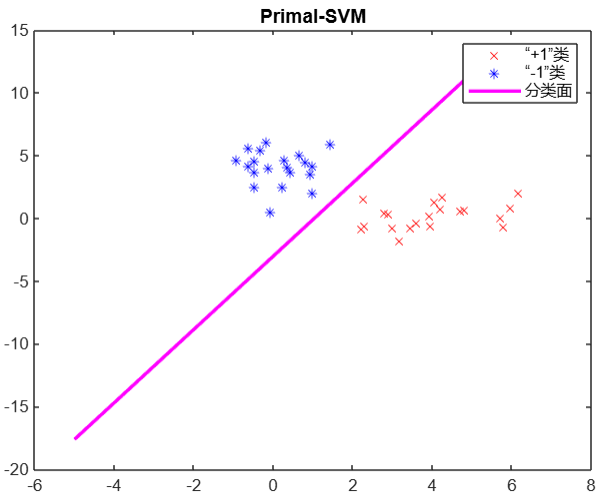
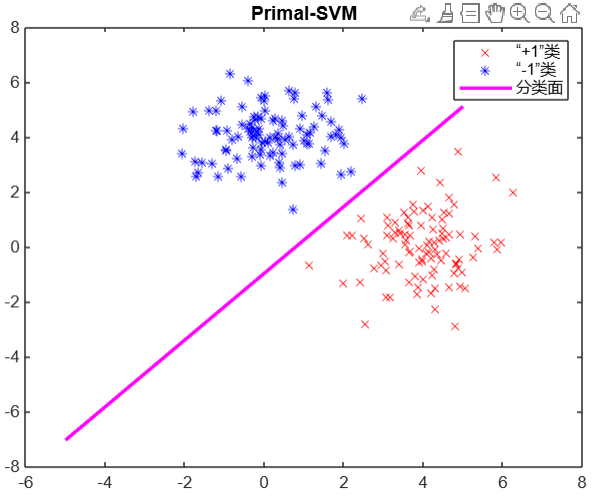
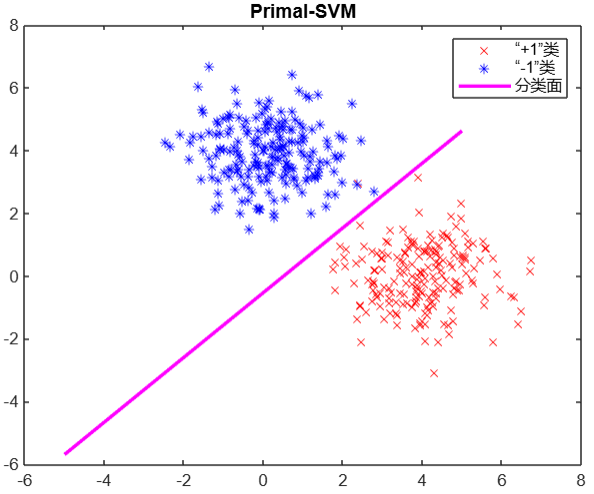


4，改变算法中的超参数、样本数量、样本分布等，讨论实验结果。

(1)改变样本数量

样本分布为[4,0],[0,4]。样本数量为200,100,20。

1. Primal-SVM



200：



100：

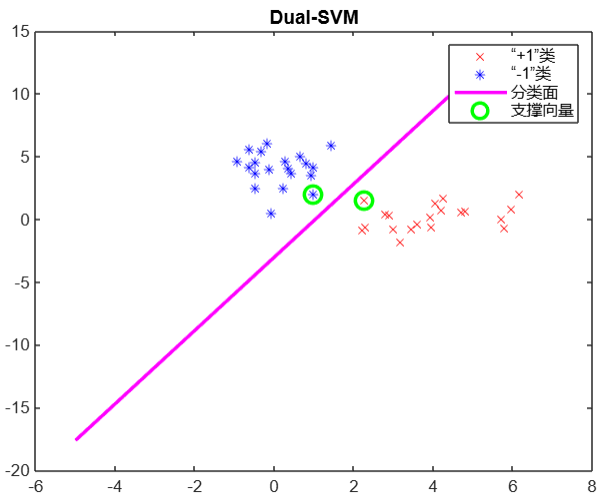
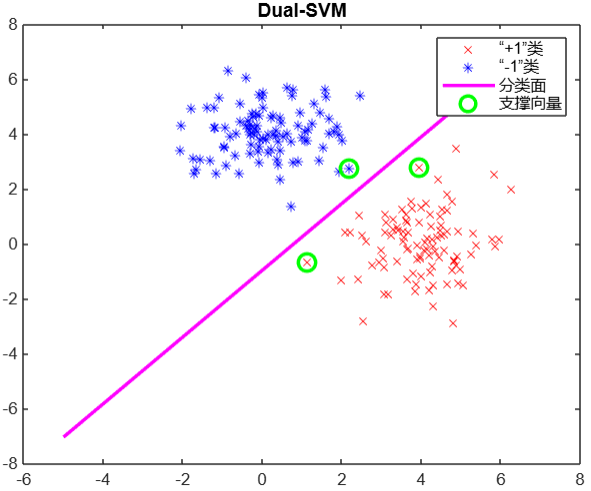
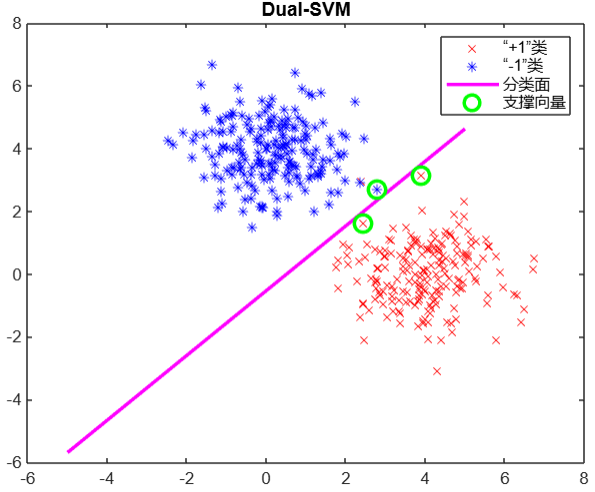


20：



可以发现：发现随着样本数量的减少间隔面的宽度越来越大,分类正确率也有所提高。

1. Dual-SVM



200：



100：

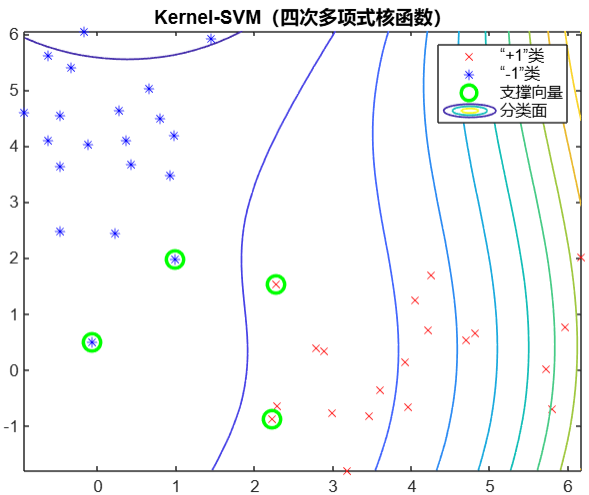
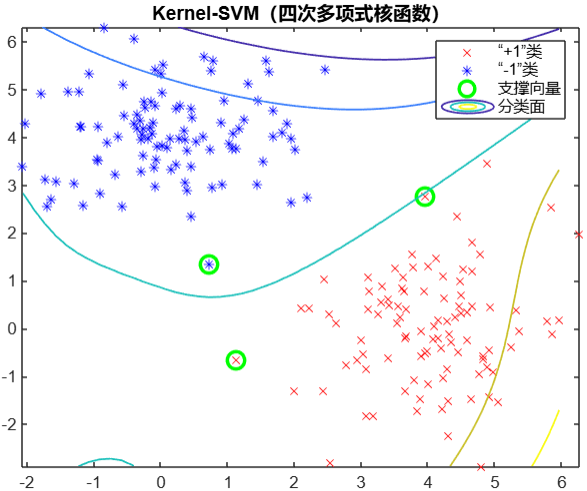
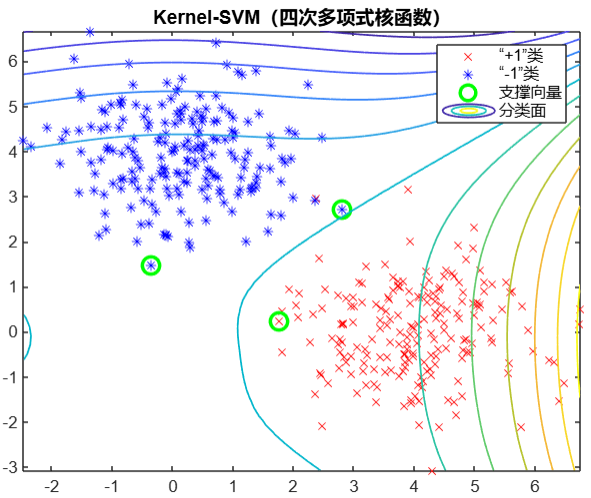


20：



可以发现：Dual-SVM同样也呈现出间隔面越来越大的趋势

1. Kernel-SVM（四次多项式核函数）



200：



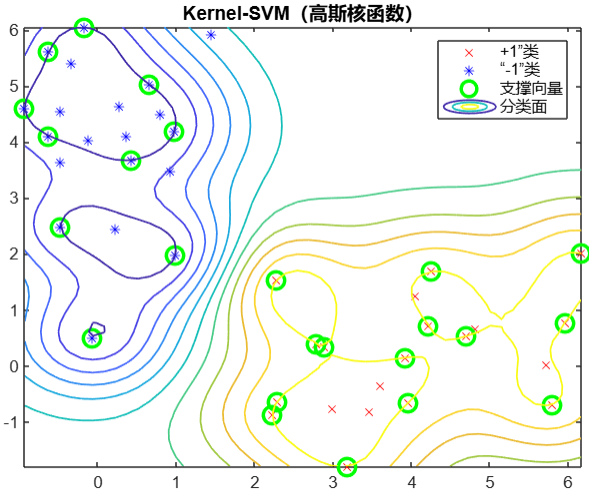
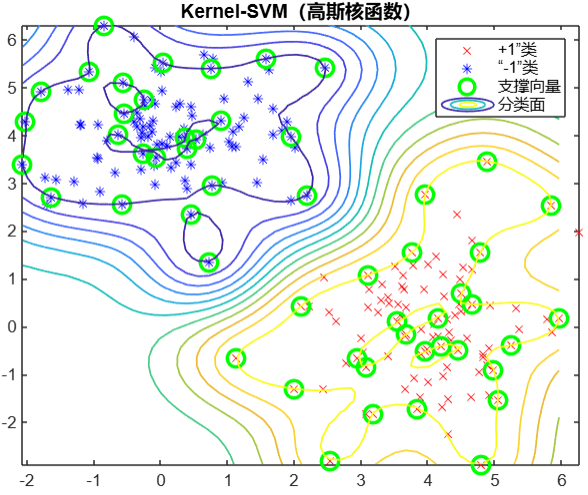
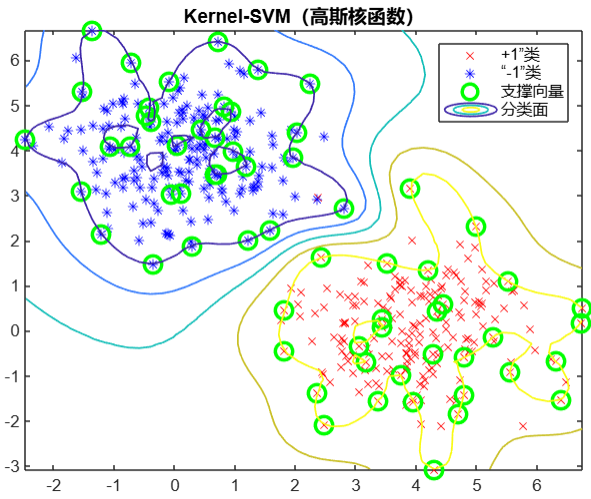
100：



20：



(d) Kernel-SVM（高斯核函数）



200：



100：



20：



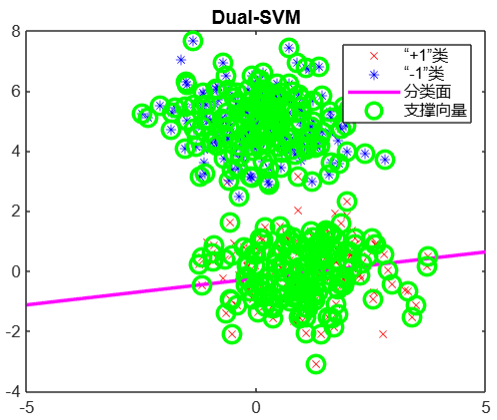
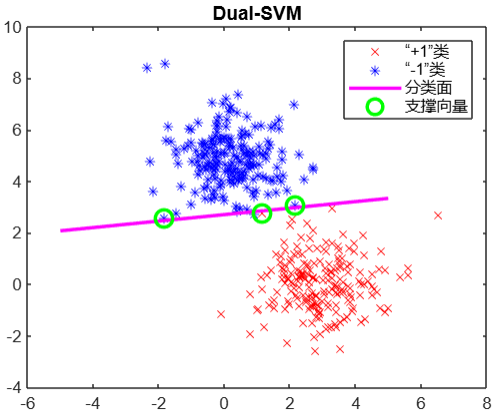
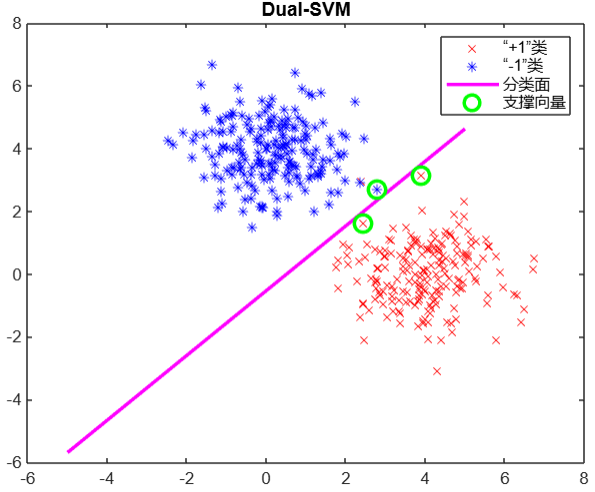
（2）改变样本分布：

样本数量为200，将样本分布分别设置为[5,0],[3,0],[1,0]

(a) Primal-SVM

由于Primal-SVM可能会出现二次规划无法求解的情况,故不研究

(b)Dual-SVM

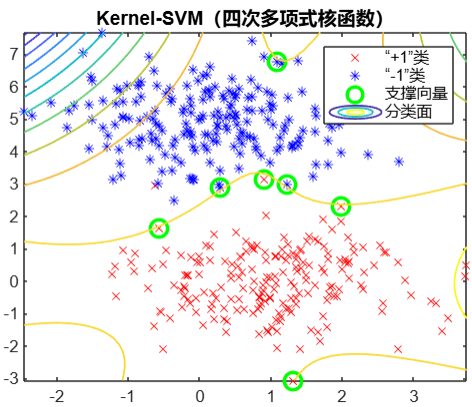
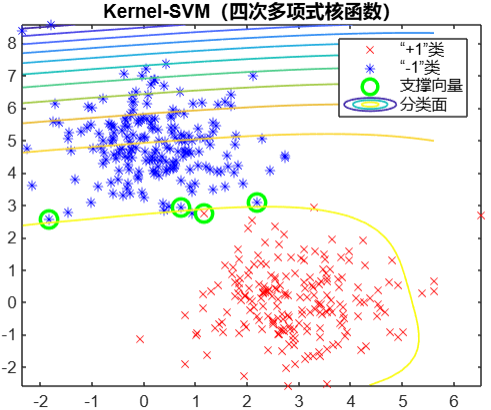
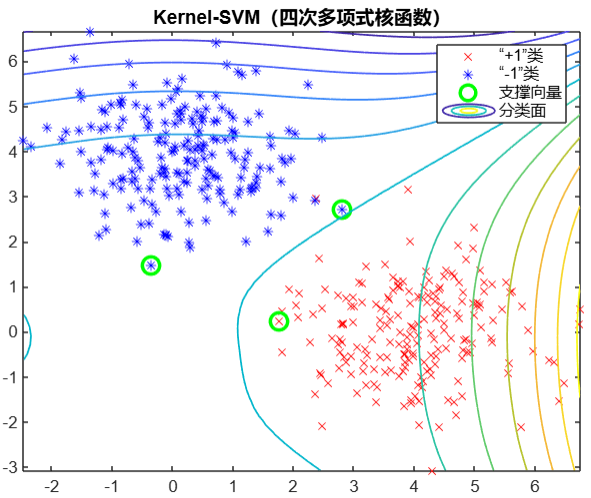


[1,0]



可以发现：在Dual-SVM下仍能找到分类面，但由于无法使样本都被分类正确，无法求出间隔面

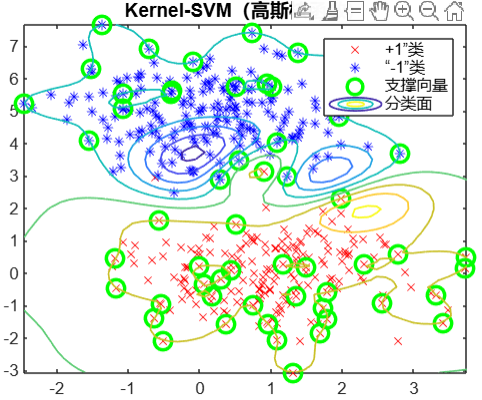
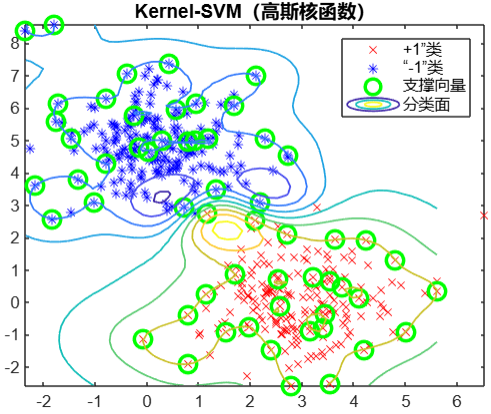
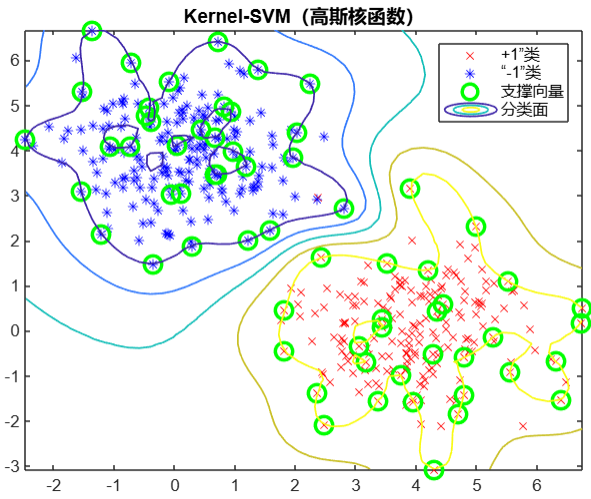
(c)Kernel-SVM（四次多项式核函数）



[1,0]



(d) Kernel-SVM（高斯核函数）



[1,0]:

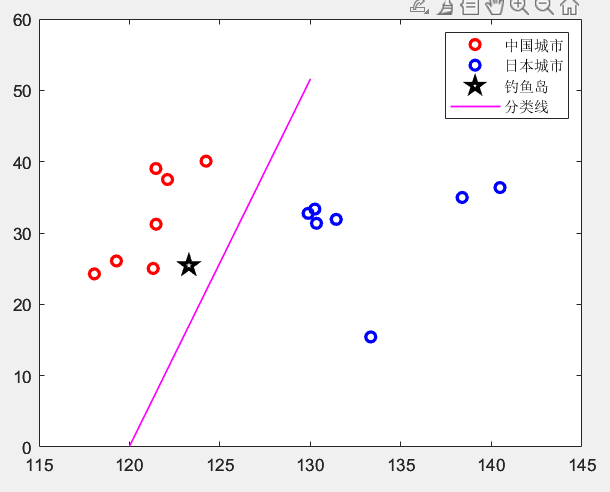


5，训练集: 中国与日本的沿海城市的经纬度坐标向量，中国标签为+1, 日本为标签为-1.

测试集: 钓鱼岛的经纬度坐标向量

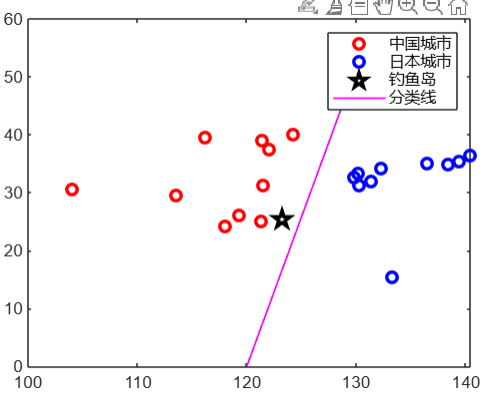
用支撑向量机设计分类器，（1）判断钓鱼岛属于哪一类；（2）增加几个非海边城市的经纬度坐标进行训练，判断这些城市是否影响分类结果，是否为支撑向量。

1. 数据集
2. %% 产生数据集
3. x1=[119.28,26.08;%福州
4. 121.31,25.03;%台北
5. 121.47,31.23;%上海
6. 118.06,24.27;%厦门
7. 121.46,39.04;%大连
8. 122.10,37.50;%威海
9. 124.23,40.07];%丹东
11. x2=[129.87,32.75;%长崎
12. 130.33,31.36;%鹿儿岛
13. 131.42,31.91;%宫崎
14. 130.24,33.35;%福冈
15. 133.33,15.43;%鸟取
16. 138.38,34.98;%静冈
17. 140.47,36.37];%水户
19. % 增加内陆城市
20. % x1=[119.28,26.08;%福州
21. %     121.31,25.03;%台北
22. %     121.47,31.23;%上海
23. %     118.06,24.27;%厦门
24. %     113.53,29.58;%武汉
25. %     104.06,30.67;%成都
26. %     116.25,39.54;%北京
27. %     121.46,39.04;%大连
28. %     122.10,37.50;%威海
29. %     124.23,40.07];%丹东
30. % x2=[129.87,32.75;%长崎
31. %     130.33,31.36;%鹿儿岛
32. %     131.42,31.91;%宫崎
33. %     130.24,33.35;%福冈
34. %     136.54,35.10;%名古屋
35. %     132.27,34.24;%广岛
36. %     139.46,35.42;%东京
37. %     133.33,15.43;%鸟取
38. %     138.38,34.98;%静冈
39. %     140.47,36.37];%水户
40. 代码
41. %% 分类器
42. z=zeros(1,2);
43. Q=[0 z;z.' eye(2)];
44. p=zeros(1,2+1);
45. A1=zeros(3,size(x1,1)\*2);
46. for i=1:3
47. A1(i,:)=trainData(i,:).\*trainLabel;
48. end
49. A=-1\*A1';
50. c=(zeros(1,size(x1,1)\*2)-1)';
51. u=quadprog(Q,p,A,c);
52. figure(1);
53. plot(X1(2,:),X1(3,:),'ro',X2(2,:),X2(3,:),'bo','LineWidth', 2);
54. X=120:0.2:130;
55. Y=-1\*(u(2,:)\*X+u(1,:))/u(3,:);
56. hold on;
57. plot(123.28,25.45,'kp','LineWidth', 2, 'markersize', 10);  %钓鱼岛坐标
58. plot(X,Y,'m-','LineWidth',1);
59. legend("中国城市","日本城市","钓鱼岛","分类线");
60. 分类结果



结果：钓鱼岛在经纬度上属于中国。

1. 增加沿海城市：



结果：新加入的城市并没有改变钓鱼岛属于中国的结果，切新城市不作为样本的支撑向量