hw6_1

March 31, 2023

[27]:
```
import numpy as np
import csv
import matplotlib.pyplot as plt
from sklearn.svm import SVC
from plotSVMBoundaries import plotSVMBoundaries
```

[9]:
```
def getdata(fname):
    data = np.empty([0,2])
    label = []
    with open(fname, mode ='r')as file:
        # reading the CSV file
        csvFile = csv.reader(file)

        # displaying the contents of the CSV file
        for lines in csvFile:
            data = np.row_stack((data,[float(lines[0]), float(lines[1])]))
            if(float(lines[2]) == 1):
                label.append(1.)
            else:
                label.append(-1.)
    label = np.array(label)
    return (data, label)
```

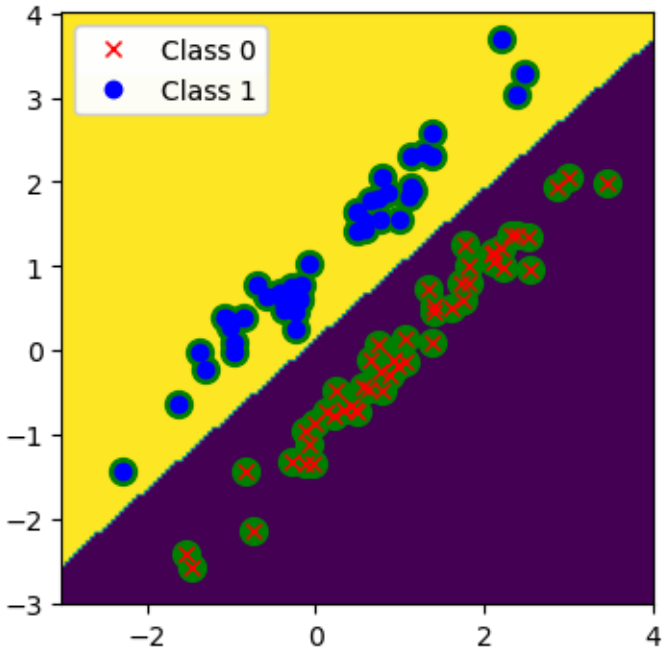
[10]:
```
xdata1_train, ydata1_train = getdata("dataset1_train.csv")
xdata1_test, ydata1_test = getdata("dataset1_test.csv")

xdata3_train, ydata3_train = getdata("dataset3_train.csv")
xdata3_test, ydata3_test = getdata("dataset3_test.csv")
```

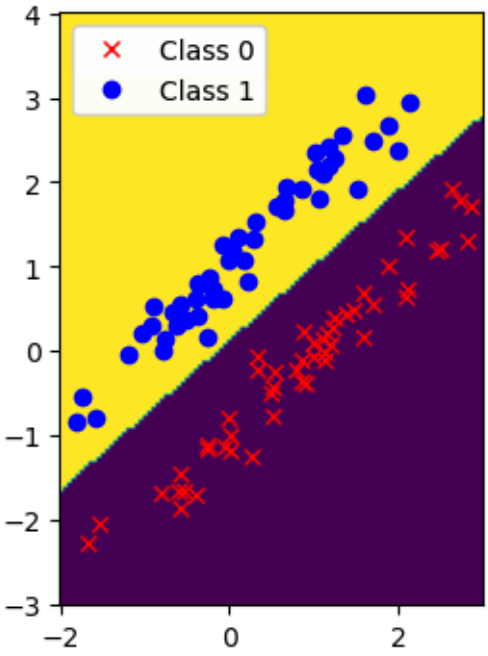
(a) Use the Linear Kernel and try different values of slack variable parameter C. What is the meaning of parameter C and how will it impact your classification? Set C = 0.01 and C = 1. Report the above items and also provide the support vectors in the plots for each value of C. Discuss your results and explain the performance and the differences for the different values of C.

[56]:
```
model1_a_1 = SVC(C=0.01, kernel='linear')
model1_a_1.fit(xdata1_train, ydata1_train)
```

```
train1_acc = model1_a_1.score(xdata1_train, ydata1_train)
test1_acc = model1_a_1.score(xdata1_test, ydata1_test)
print('train acc = {}, test acc = {}'.format(train1_acc, test1_acc))
print('weight vector w = ', model1_a_1.coef_)
print('offset w0 =', model1_a_1.intercept_)

plotSVMBoundaries(xdata1_train, ydata1_train, model1_a_1, support_vectors =␣
 ↪model1_a_1.support_vectors_)
plotSVMBoundaries(xdata1_test, ydata1_test, model1_a_1)
```

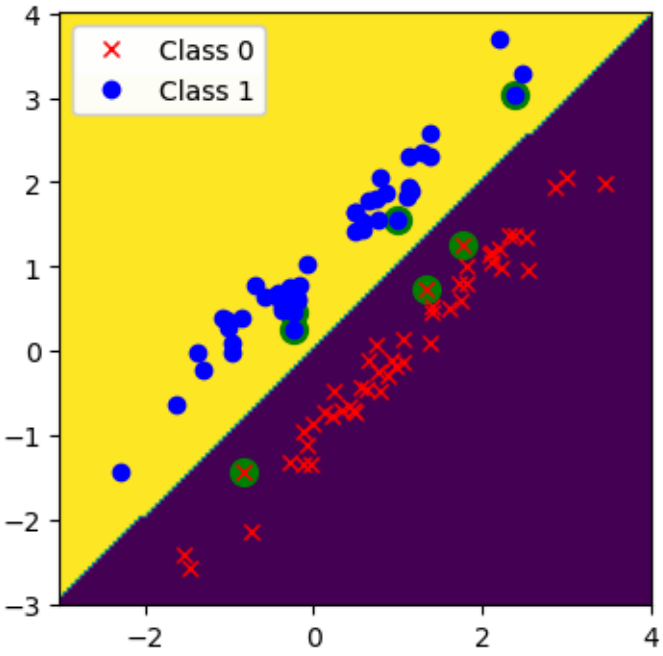
```
train acc = 1.0, test acc = 1.0
weight vector w =  [[-0.47617753  0.53480959]]
offset w0 = [-0.06369157]
```

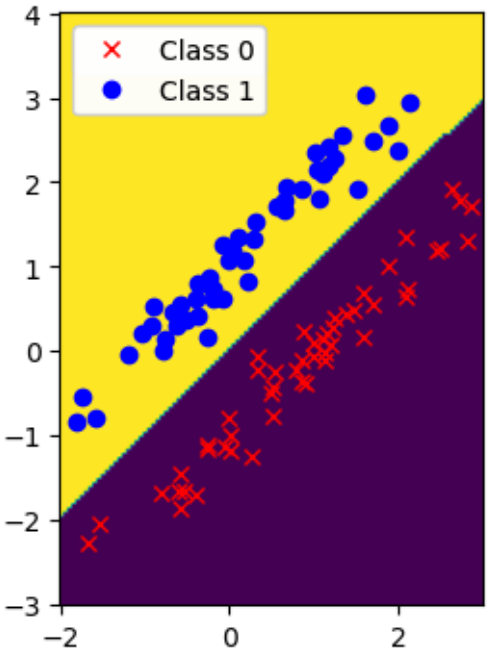




[57]:
```
model1_a_2 = SVC(C=1.0, kernel='linear')
model1_a_2.fit(xdata1_train, ydata1_train)

train1_acc = model1_a_2.score(xdata1_train, ydata1_train)
test1_acc = model1_a_2.score(xdata1_test, ydata1_test)

print('train acc = {}, test acc = {}'.format(train1_acc, test1_acc))
print('weight vector w = ', model1_a_2.coef_)
print('offset w0 =', model1_a_2.intercept_)

plotSVMBoundaries(xdata1_train, ydata1_train, model1_a_2, support_vectors =␣
↪model1_a_2.support_vectors_)
plotSVMBoundaries(xdata1_test, ydata1_test, model1_a_2)
```

```
train acc = 1.0, test acc = 1.0
weight vector w =  [[-1.5291679   1.54596349]]
offset w0 = [-0.04317366]
```



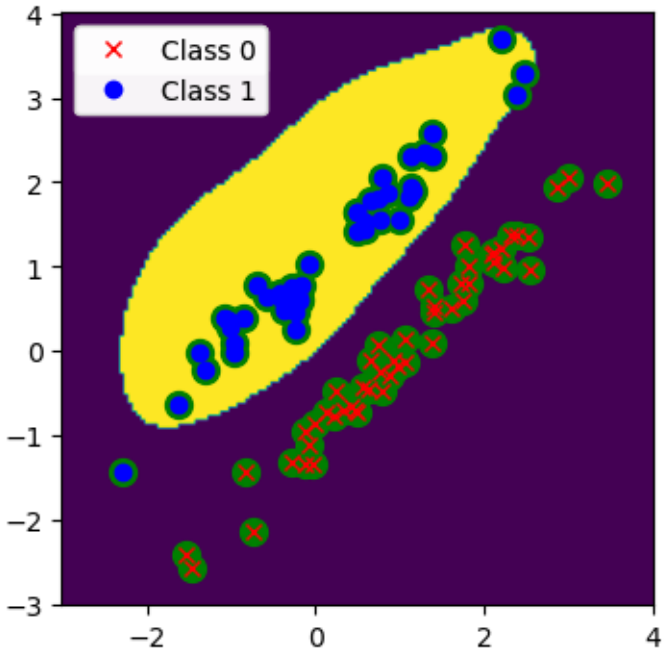


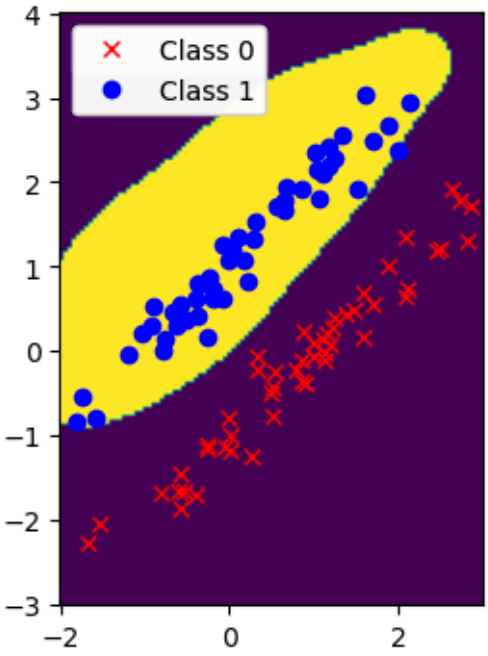
(b) Use a Gaussian (RBF) Kernel with C parameter set to C = 0.01. Set = 1,3,10,50. Report the above items and also show the support vectors in the training-data plots for each value of . Explain the linearity or nonlinearity of the decision boundary, and explain the difference in

decision regions for the various values of . State where (if anywhere) you observe underfitting or overfitting.

[58]:
```
model1_b_1 = SVC(C=0.01, kernel='rbf',gamma = 1.0)
model1_b_1.fit(xdata1_train, ydata1_train)

train1_acc = model1_b_1.score(xdata1_train, ydata1_train)
test1_acc = model1_b_1.score(xdata1_test, ydata1_test)
print('train acc = {}, test acc = {}'.format(train1_acc, test1_acc))

plotSVMBoundaries(xdata1_train, ydata1_train, model1_b_1, support_vectors =␣
↪model1_b_1.support_vectors_)
plotSVMBoundaries(xdata1_test, ydata1_test, model1_b_1)
```

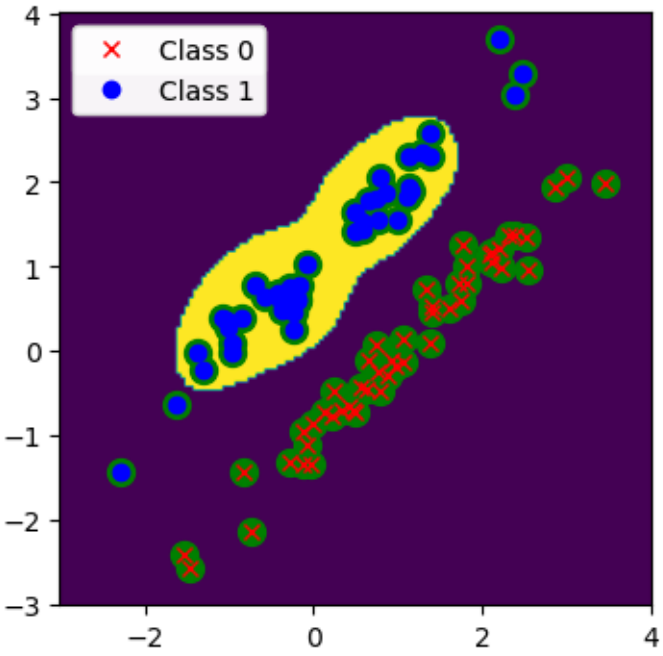
```
train acc = 0.99, test acc = 0.99
```

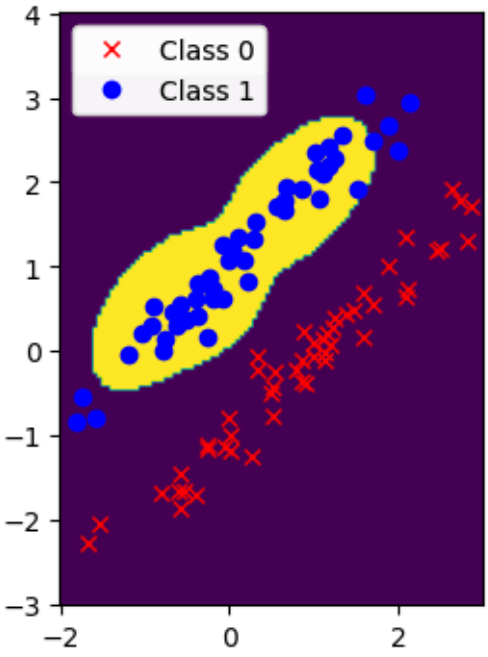




[59]:
```
model1_b_2 = SVC(C=0.01, kernel='rbf',gamma = 3.0)
model1_b_2.fit(xdata1_train, ydata1_train)

train1_acc = model1_b_2.score(xdata1_train, ydata1_train)
test1_acc = model1_b_2.score(xdata1_test, ydata1_test)
print('train acc = {}, test acc = {}'.format(train1_acc, test1_acc))

plotSVMBoundaries(xdata1_train, ydata1_train, model1_b_2, support_vectors =␣
↪model1_b_2.support_vectors_)
plotSVMBoundaries(xdata1_test, ydata1_test, model1_b_2)
```

```
train acc = 0.95, test acc = 0.92
```

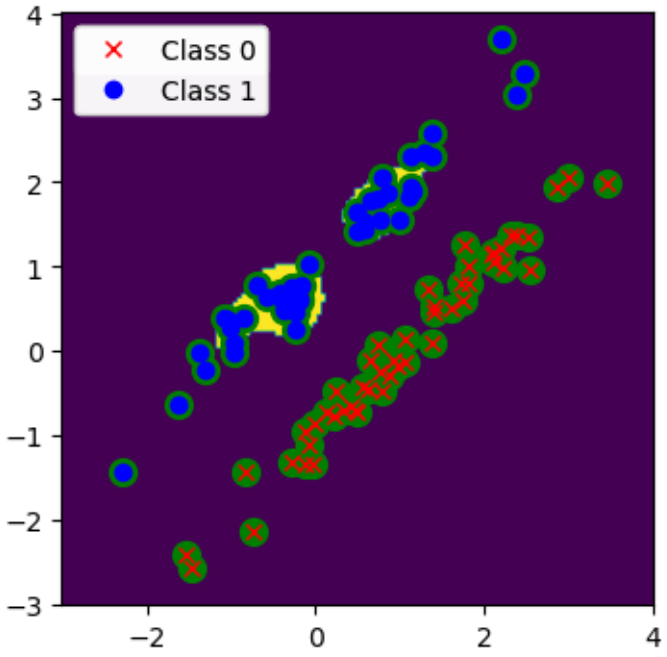


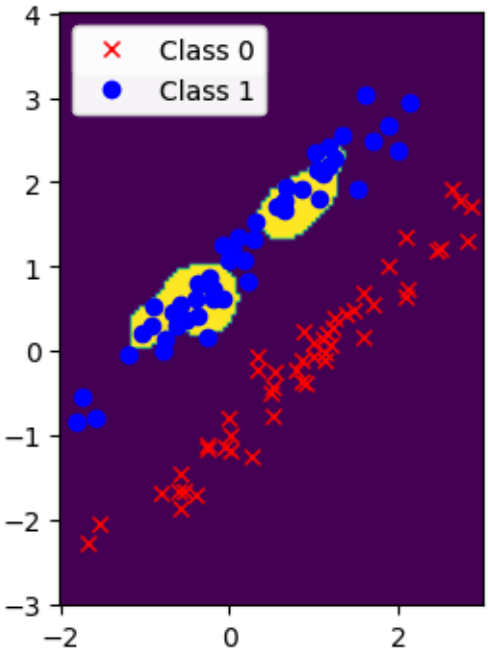


[60]:
```
model1_b_3 = SVC(C=0.01, kernel='rbf',gamma = 10.0)
model1_b_3.fit(xdata1_train, ydata1_train)
```

```
train1_acc = model1_b_3.score(xdata1_train, ydata1_train)
test1_acc = model1_b_3.score(xdata1_test, ydata1_test)
print('train acc = {}, test acc = {}'.format(train1_acc, test1_acc))

plotSVMBoundaries(xdata1_train, ydata1_train, model1_b_3, support_vectors =␣
 ↪model1_b_3.support_vectors_)
plotSVMBoundaries(xdata1_test, ydata1_test, model1_b_3)
```

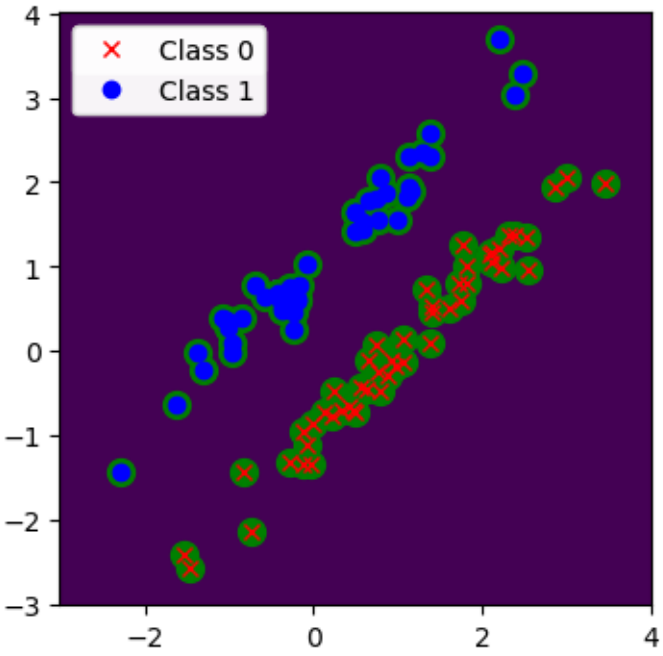
```
train acc = 0.88, test acc = 0.77
```

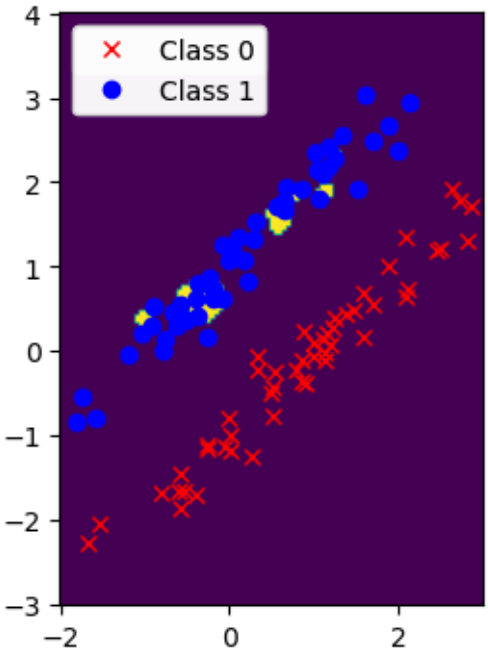




[61]:
```
model1_b_4 = SVC(C=0.01, kernel='rbf',gamma = 50.0)
model1_b_4.fit(xdata1_train, ydata1_train)

train1_acc = model1_b_4.score(xdata1_train, ydata1_train)
test1_acc = model1_b_4.score(xdata1_test, ydata1_test)
print('train acc = {}, test acc = {}'.format(train1_acc, test1_acc))

plotSVMBoundaries(xdata1_train, ydata1_train, model1_b_4, support_vectors =␣
↪model1_b_4.support_vectors_)
plotSVMBoundaries(xdata1_test, ydata1_test, model1_b_4)
```

```
train acc = 0.79, test acc = 0.6
```

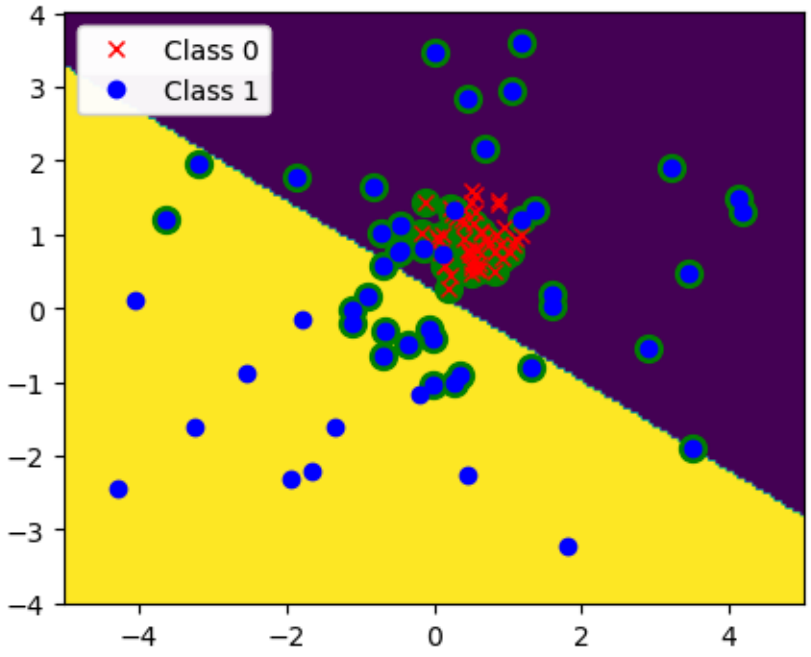


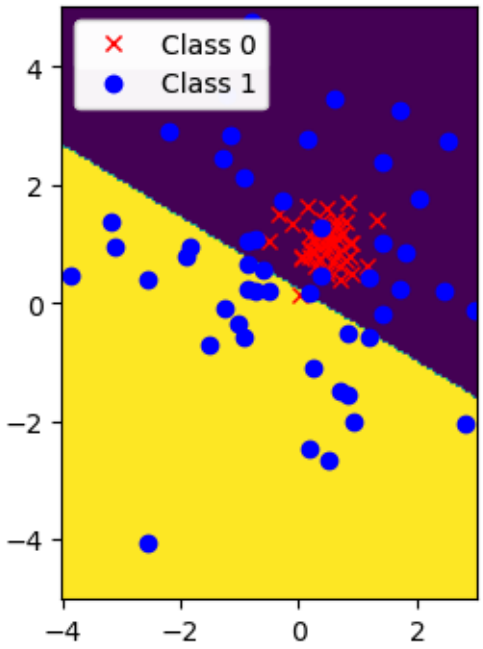


(c) Use the Linear Kernel and try different values of slack variable parameter C. Set C = 1 and C = 100. Report the above items for each value of C. Discuss your results. You will provide 4 plots in total.

[73]:
```
model2_c_1 = SVC(C=1.0, kernel='linear')
model2_c_1.fit(xdata3_train, ydata3_train)

train3_acc = model2_c_1.score(xdata3_train, ydata3_train)
test3_acc = model2_c_1.score(xdata3_test, ydata3_test)
print('train acc = {}, test acc = {}'.format(train3_acc, test3_acc))
print('weight vector w = ', model2_c_1.coef_)
print('offset w0 =', model2_c_1.intercept_)

plotSVMBoundaries(xdata3_train, ydata3_train, model2_c_1, support_vectors =␣
↪model2_c_1.support_vectors_)
plotSVMBoundaries(xdata3_test, ydata3_test, model2_c_1)
```

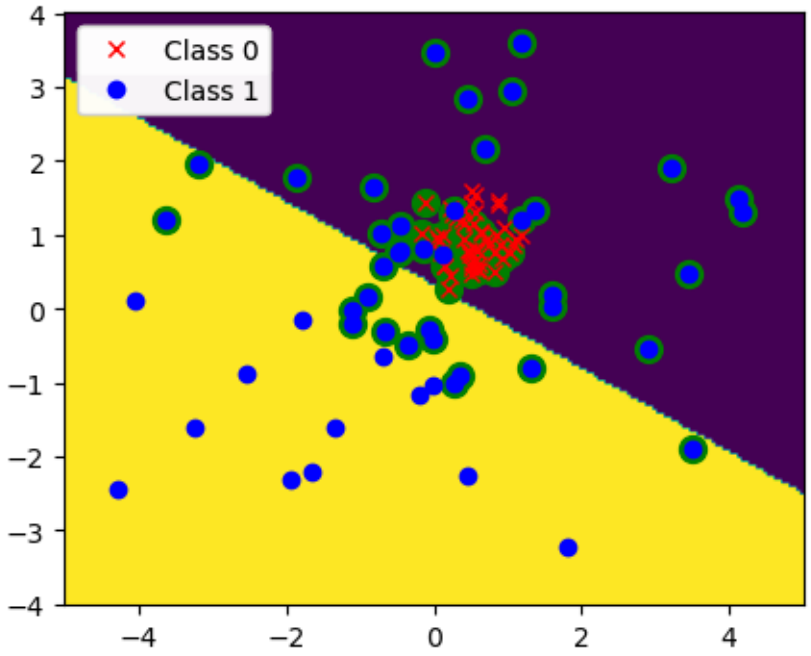
```
train acc = 0.76, test acc = 0.74
weight vector w =  [[-0.48069371 -0.78654222]]
offset w0 = [0.1697672]
```

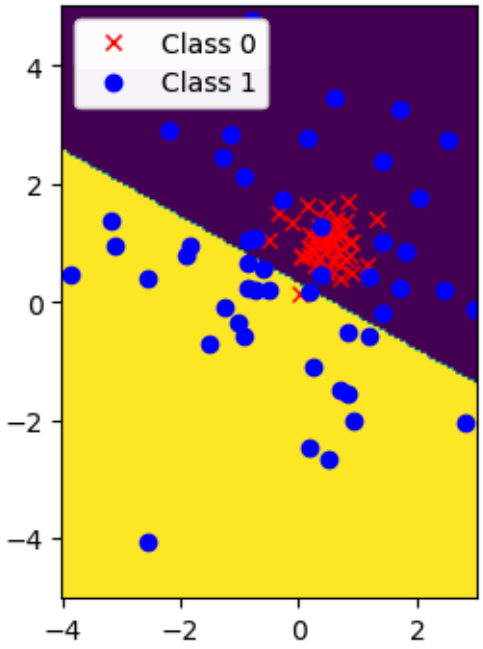




[72]:
```
model2_c_2 = SVC(C=100.0, kernel='linear')
model2_c_2.fit(xdata3_train, ydata3_train)

train3_acc = model2_c_2.score(xdata3_train, ydata3_train)
test3_acc = model2_c_2.score(xdata3_test, ydata3_test)

print('train acc = {}, test acc = {}'.format(train3_acc, test3_acc))
print('weight vector w = ', model2_c_2.coef_)
print('offset w0 =', model2_c_2.intercept_)

plotSVMBoundaries(xdata3_train, ydata3_train, model2_c_2, support_vectors =␣
↪model2_c_2.support_vectors_ )
plotSVMBoundaries(xdata3_test, ydata3_test, model2_c_2)
```

```
train acc = 0.77, test acc = 0.75
weight vector w =  [[-0.49088354 -0.86901288]]
offset w0 = [0.26395834]
```



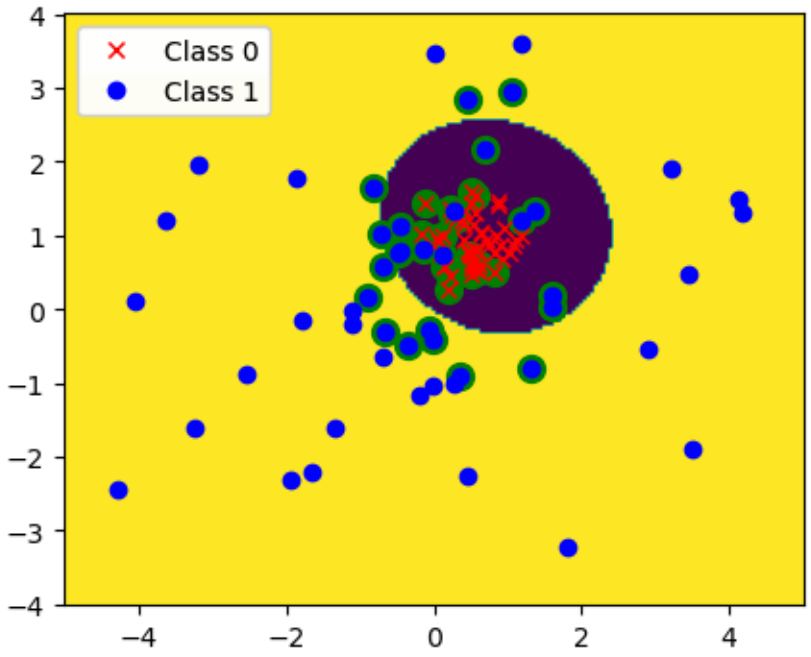


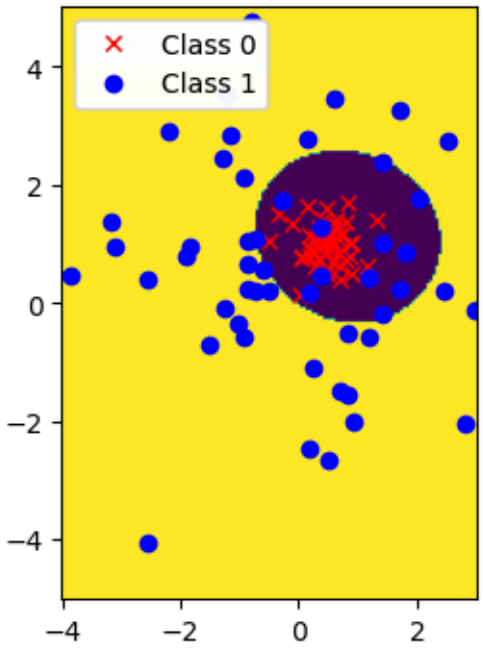
(d) Use a Gaussian (RBF) Kernel with C parameter set to C = 1. Set = 0.1, 10, 200. Report the above items and also show the support vectors in the training-data plots for each value of . Explain the difference in decision regions for the different values of . Tip: you might want to try plots at other values of to help you understand its effects (no need to include

these extra plots in your solution). Do you observe any overfitting or underfitting for any of the given values of ? You will provide 6 plots in total.

[66]:
```
model2_d_1 = SVC(C=1.0, kernel='rbf', gamma = 0.1)
model2_d_1.fit(xdata3_train, ydata3_train)

train3_acc = model2_d_1.score(xdata3_train, ydata3_train)
test3_acc = model2_d_1.score(xdata3_test, ydata3_test)

print('train acc = {}, test acc = {}'.format(train3_acc, test3_acc))

plotSVMBoundaries(xdata3_train, ydata3_train, model2_d_1, support_vectors =␣
↪model2_d_1.support_vectors_ )
plotSVMBoundaries(xdata3_test, ydata3_test, model2_d_1)
```

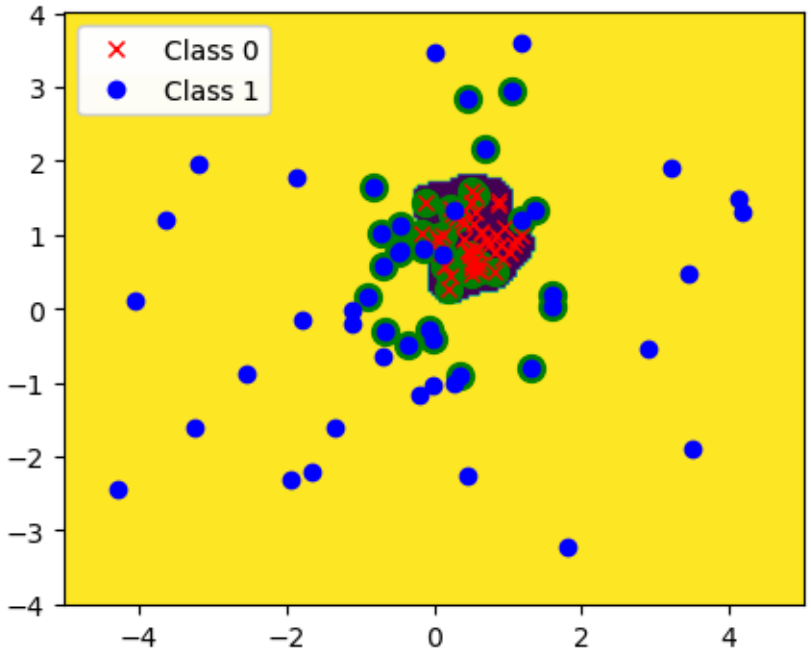
```
train acc = 0.89, test acc = 0.89
```

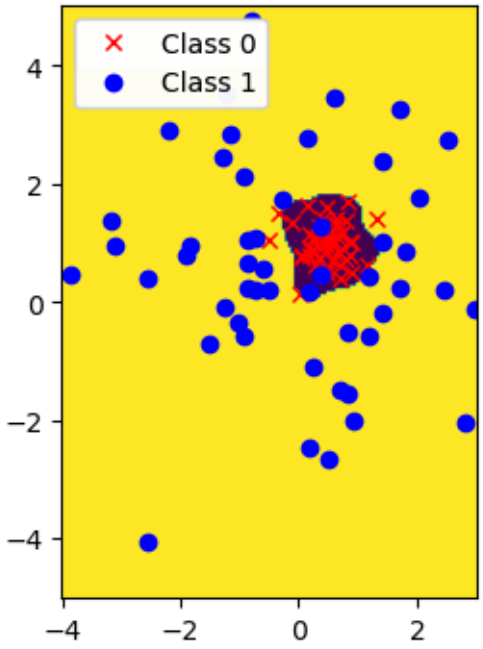




[67]:
```
model2_d_2 = SVC(C=1.0, kernel='rbf', gamma = 10.0)
model2_d_2.fit(xdata3_train, ydata3_train)

train3_acc = model2_d_2.score(xdata3_train, ydata3_train)
test3_acc = model2_d_2.score(xdata3_test, ydata3_test)

print('train acc = {}, test acc = {}'.format(train3_acc, test3_acc))

plotSVMBoundaries(xdata3_train, ydata3_train, model2_d_2, support_vectors =␣
↪model2_d_1.support_vectors_ )
plotSVMBoundaries(xdata3_test, ydata3_test, model2_d_2)
```

```
train acc = 0.98, test acc = 0.94
```

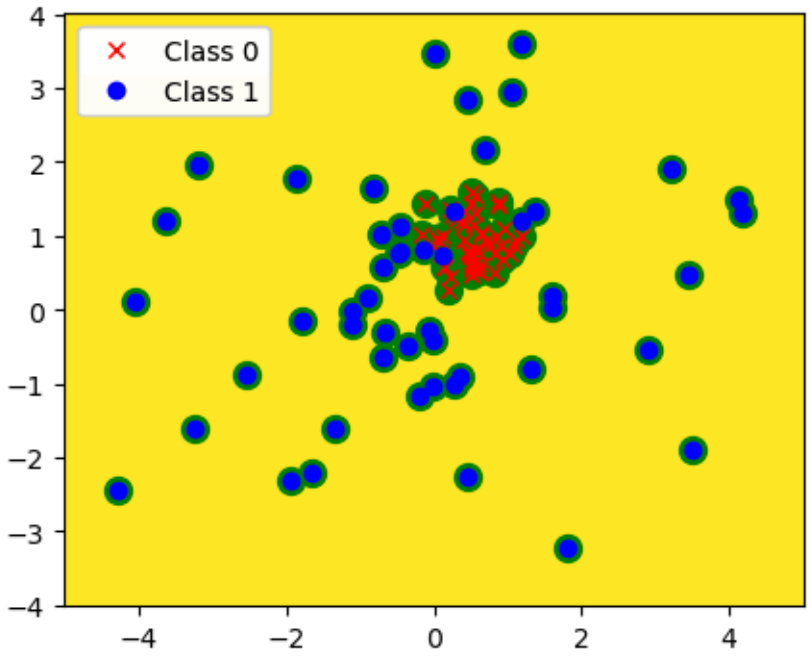


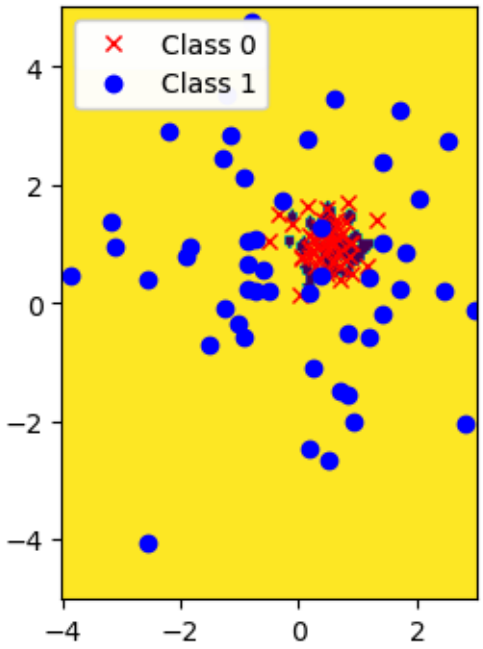


[68]:
```
model2_d_3 = SVC(C=1.0, kernel='rbf', gamma = 200.0)
model2_d_3.fit(xdata3_train, ydata3_train)
```

```
train3_acc = model2_d_3.score(xdata3_train, ydata3_train)
test3_acc = model2_d_3.score(xdata3_test, ydata3_test)

print('train acc = {}, test acc = {}'.format(train3_acc, test3_acc))

plotSVMBoundaries(xdata3_train, ydata3_train, model2_d_3, support_vectors =␣
 ↪model2_d_3.support_vectors_)
plotSVMBoundaries(xdata3_test, ydata3_test, model2_d_3)
```

```
train acc = 0.98, test acc = 0.77
```

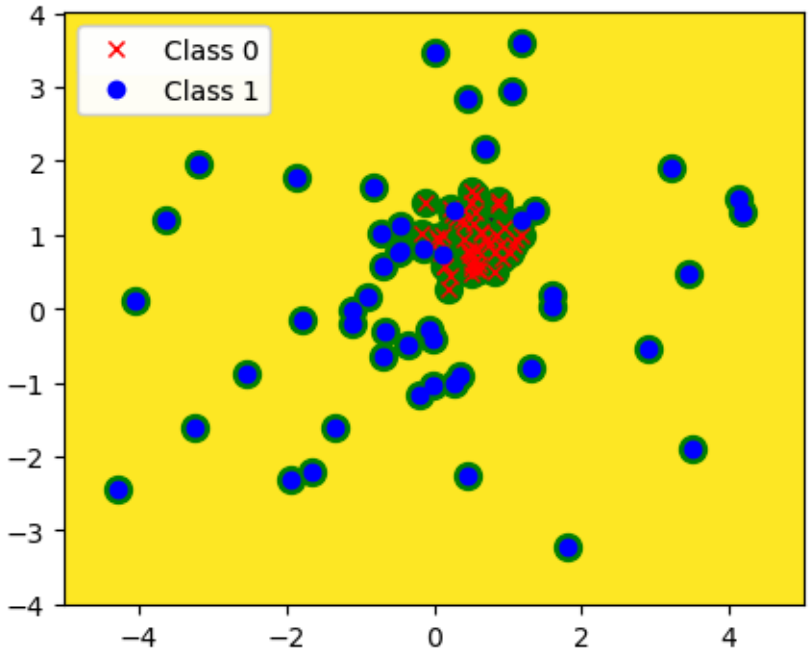


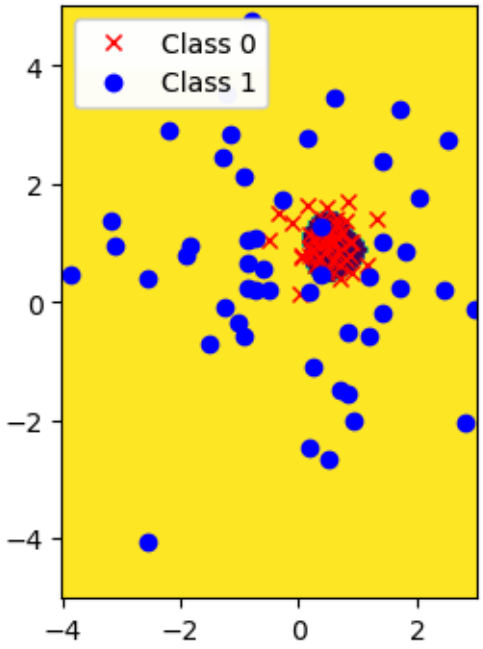


(e) Use a Gaussian (RBF) Kernel and pick the parameter from part (d) (from the 3 given values) that results in the minimum test error. Set C = 0.01,1,100. Report the above items and also provide the support vectors in the plots for each value of C. Explain your observations in the different decision boundaries and the support vectors for the different values of C. You will provide 6 plots in total.

[74]:
```
model2_e_1 = SVC(C=0.01, kernel='rbf', gamma = 10.0)
model2_e_1.fit(xdata3_train, ydata3_train)

train3_acc = model2_e_1.score(xdata3_train, ydata3_train)
test3_acc = model2_e_1.score(xdata3_test, ydata3_test)

print('train acc = {}, test acc = {}'.format(train3_acc, test3_acc))

plotSVMBoundaries(xdata3_train, ydata3_train, model2_e_1, support_vectors =␣
↪model2_e_1.support_vectors_)
plotSVMBoundaries(xdata3_test, ydata3_test, model2_e_1)
```

```
train acc = 0.88, test acc = 0.86
```

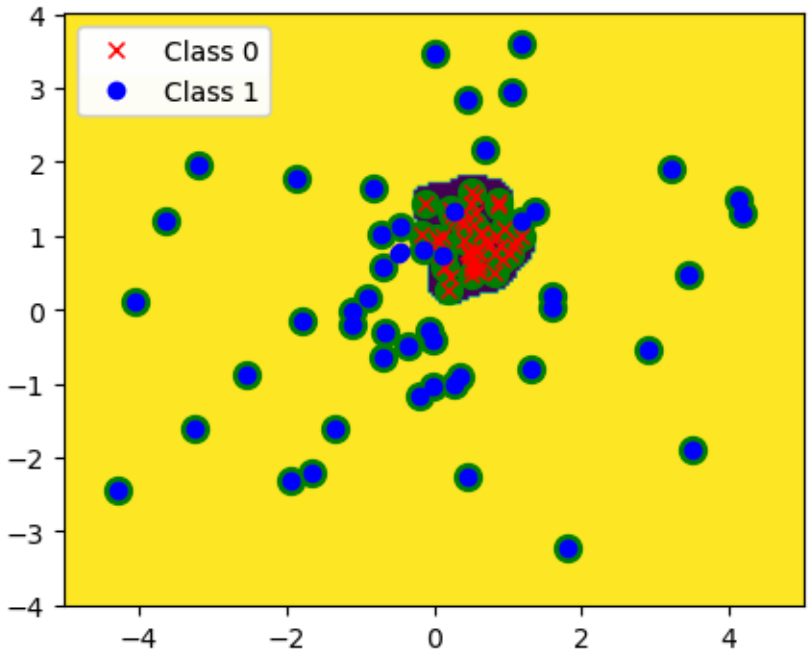


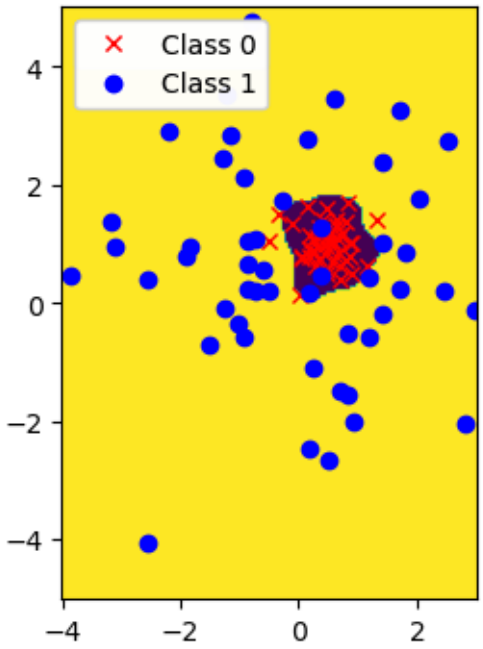


[75]:
```
model2_e_2 = SVC(C=1.0, kernel='rbf', gamma = 10.0)
model2_e_2.fit(xdata3_train, ydata3_train)
```

```
train3_acc = model2_e_2.score(xdata3_train, ydata3_train)
test3_acc = model2_e_2.score(xdata3_test, ydata3_test)

print('train acc = {}, test acc = {}'.format(train3_acc, test3_acc))

plotSVMBoundaries(xdata3_train, ydata3_train, model2_e_2, support_vectors =␣
 ↪model2_e_2.support_vectors_)
plotSVMBoundaries(xdata3_test, ydata3_test, model2_e_2)
```

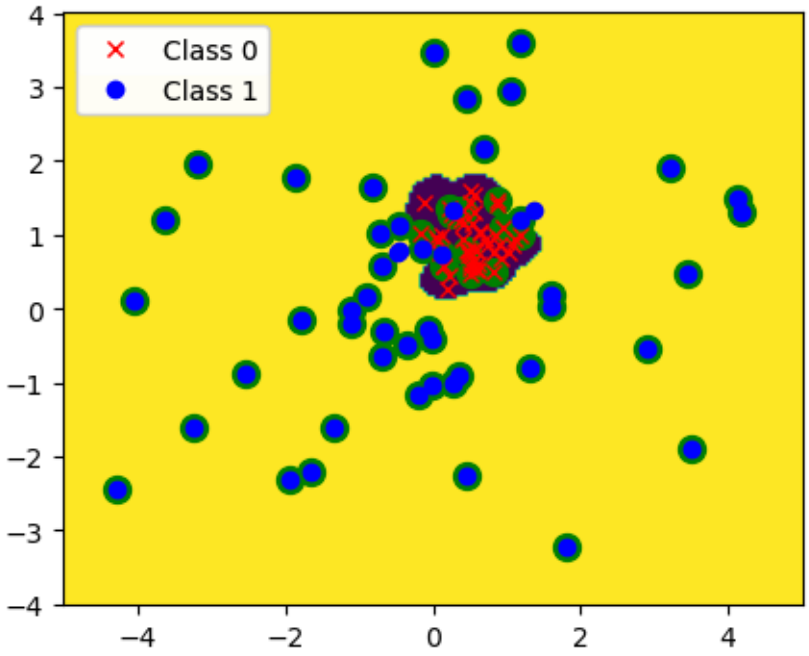
```
train acc = 0.98, test acc = 0.94
```

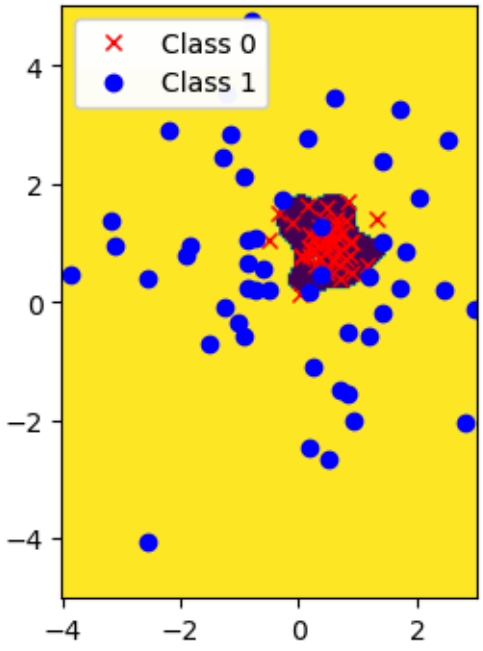




[76]:
```
model2_e_3 = SVC(C=100.0, kernel='rbf', gamma = 10.0)
model2_e_3.fit(xdata3_train, ydata3_train)

train3_acc = model2_e_3.score(xdata3_train, ydata3_train)
test3_acc = model2_e_3.score(xdata3_test, ydata3_test)
print('train acc = {}, test acc = {}'.format(train3_acc, test3_acc))
plotSVMBoundaries(xdata3_train, ydata3_train, model2_e_3, support_vectors =␣
↪model2_e_3.support_vectors_)
plotSVMBoundaries(xdata3_test, ydata3_test, model2_e_3)
```

```
train acc = 0.98, test acc = 0.95
```


Class 0
Class 1
4
3
2
1
0
−1
−2
−3
−4
−4
−2
0
2
4


Class 0
Class 1
4
2
0
−2
−4
−4
−2
0
2

[ ]: