HW 10

Rakshitha Panduranga

Problem 2 a

In [66]:

```
import numpy as np
import pandas as pd
from sklearn import svm
import matplotlib.pyplot as plt
from sklearn.model_selection import KFold
```

In [385]:

```
feature_tr = np.genfromtxt('feature_train.csv',delimiter=',');
label_tr = np.genfromtxt('label_train.csv',delimiter=',');
feature_te = np.genfromtxt('feature_test.csv',delimiter=',');
label_te = np.genfromtxt('label_test.csv',delimiter=',');

```

In [386]:

```
feature_tr
```

Out[386]:

```
array([[1.375e+01, 1.730e+00, 2.410e+00, ..., 1.150e+00, 2.900e+00,
        1.320e+03],
       [1.320e+01, 1.780e+00, 2.140e+00, ..., 1.050e+00, 3.400e+00,
        1.050e+03],
       [1.307e+01, 1.500e+00, 2.100e+00, ..., 1.180e+00, 2.690e+00,
        1.020e+03],
       ...,
       [1.253e+01, 5.510e+00, 2.640e+00, ..., 8.200e-01, 1.690e+00,
        5.150e+02],
       [1.288e+01, 2.990e+00, 2.400e+00, ..., 7.400e-01, 1.420e+00,
        5.300e+02],
       [1.373e+01, 4.360e+00, 2.260e+00, ..., 7.800e-01, 1.750e+00,
        5.200e+02]])
```

In [387]:

```
label_tr
```

Out[387]:

```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 2., 2., 2., 2.,
       2., 2., 2., 2., 2., 2., 2., 2., 2., 2., 2., 2., 2., 2., 2., 2.,
       2., 2., 2., 2., 2., 2., 2., 2., 2., 2., 2., 2., 2., 3., 3., 3.,
       3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3.,
       3., 3., 3., 3.])
```

In [388]:

```
label_tr = np.vstack(label_tr)
label_te = np.vstack(label_te)
```

In [389]:

```
train_data = feature_tr[:,0:2]
test_data = feature_te[:,0:2]
```

In [130]:

```
train_data_wlabels = np.column_stack((train_data,label_tr))
```

In [97]:

```
n_cv = 5
cv = StratifiedKFold(n_cv)
precomputed_folds = list(cv.split(train_data, label_tr))
```

In [98]:

```
1 precomputed_folds
```

Out[98]:

```
[(array([ 6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22,
         23, 24, 25, 26, 27, 28, 29, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46,
         47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63,
         64, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85,
         86, 87, 88]),
  array([ 0,  1,  2,  3,  4,  5, 30, 31, 32, 33, 34, 35, 36, 65, 66, 67, 68,
         69])),
 (array([ 0,  1,  2,  3,  4,  5, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22,
         23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 44, 45, 46,
         47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63,
         64, 65, 66, 67, 68, 69, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85,
         86, 87, 88]),
  array([ 6,  7,  8,  9, 10, 11, 37, 38, 39, 40, 41, 42, 43, 70, 71, 72, 73,
         74])),
 (array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 18, 19, 20, 21, 22,
         23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39,
         40, 41, 42, 43, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63,
         64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 80, 81, 82, 83, 84, 85,
         86, 87, 88]),
  array([12, 13, 14, 15, 16, 17, 44, 45, 46, 47, 48, 49, 50, 75, 76, 77, 78,
         79])),
 (array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
         17, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39,
         40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 58, 59, 60, 61, 62, 63,
         64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 85,
         86, 87, 88]),
  array([18, 19, 20, 21, 22, 23, 51, 52, 53, 54, 55, 56, 57, 80, 81, 82, 83,
         84])),
 (array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
         17, 18, 19, 20, 21, 22, 23, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39,
         40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56,
         57, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80,
         81, 82, 83, 84]),
  array([24, 25, 26, 27, 28, 29, 58, 59, 60, 61, 62, 63, 64, 85, 86, 87, 8
8]))]
```

In [155]:

```
accuracy = [];
for train_index, test_index in cv.split(train_data, label_tr):
    count = 0;
    X_train, X_test = train_data[train_index], train_data[test_index];
    clf = svm.SVC(C = 1,kernel='rbf',gamma = 1)
    clf.fit(X_train,train_data_wlabels[train_index,2:3])
    labels = clf.predict(X_test)
    sample = train_data_wlabels[test_index,2:3]
    for i in range(0,len(labels)):
        if(labels[i] == sample[i]):
            count = count+1;
    accuracy.append((count/len(labels))*100);
average_acc = np.mean(accuracy);
```

```
C:\Users\Hp\Anaconda3\lib\site-packages\sklearn\utils\validation.py:578: Dat
aConversionWarning: A column-vector y was passed when a 1d array was expecte
d. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

In [156]:

```
accuracy
```

Out[156]:

```
[94.44444444444444,
 83.33333333333334,
 66.66666666666666,
 88.88888888888889,
 76.47058823529412]
```

In [308]:

```
print("The average cross-validation accuracy =",average_acc)
```

```
The average cross-validation accuracy = 81.96078431372548
```

Problem 2 (b)

In [237]:

```
Ny = 50;
Nc = 50;
y = np.logspace((-3),(3),Ny,base = 10)
c=  np.logspace((-3),(3),Nc,base = 10)
```

In [292]:

```
c
```

Out[292]:

```
array([1.00000000e-03, 1.32571137e-03, 1.75751062e-03, 2.32995181e-03,
       3.08884360e-03, 4.09491506e-03, 5.42867544e-03, 7.19685673e-03,
       9.54095476e-03, 1.26485522e-02, 1.67683294e-02, 2.22299648e-02,
       2.94705170e-02, 3.90693994e-02, 5.17947468e-02, 6.86648845e-02,
       9.10298178e-02, 1.20679264e-01, 1.59985872e-01, 2.12095089e-01,
       2.81176870e-01, 3.72759372e-01, 4.94171336e-01, 6.55128557e-01,
       8.68511374e-01, 1.15139540e+00, 1.52641797e+00, 2.02358965e+00,
       2.68269580e+00, 3.55648031e+00, 4.71486636e+00, 6.25055193e+00,
       8.28642773e+00, 1.09854114e+01, 1.45634848e+01, 1.93069773e+01,
       2.55954792e+01, 3.39322177e+01, 4.49843267e+01, 5.96362332e+01,
       7.90604321e+01, 1.04811313e+02, 1.38949549e+02, 1.84206997e+02,
       2.44205309e+02, 3.23745754e+02, 4.29193426e+02, 5.68986603e+02,
       7.54312006e+02, 1.00000000e+03])
```

In [239]:

```
n_cv = 5;
def acc(C_value, g_value):
    accuracy = [];
    cv = StratifiedKFold(n_cv)
    clf = svm.SVC(C = C_value,kernel='rbf',gamma = g_value)
    for train_index, test_index in cv.split(train_data, label_tr):
        count = 0;
        X_train, X_test = train_data[train_index], train_data[test_index];
        clf.fit(X_train,train_data_wlabels[train_index,2:3])
        labels = clf.predict(X_test)
        sample = train_data_wlabels[test_index,2:3]
        for i in range(0,len(labels)):
            if(labels[i] == sample[i]):
                count = count+1;
        accuracy.append((count/len(labels)));
    average_acc = np.mean(accuracy);

    sd_acc = np.std(accuracy);
    return average_acc,sd_acc;
```

In [327]:

```
n_cv = 5;
def acc(C_value, g_value):
    accuracy = [];
    cv = StratifiedKFold(n_cv)
    clf = svm.SVC(C = C_value,kernel='rbf',gamma = g_value)
    for train_index, test_index in cv.split(train_data, label_tr):
        count = 0;
        X_train, X_test = train_data[train_index], train_data[test_index];
        clf.fit(X_train,train_data_wlabels[train_index,2:3])
        labels = clf.predict(X_test)
        sample = train_data_wlabels[test_index,2:3]
        for i in range(0,len(labels)):
            if(labels[i] == sample[i]):
                count = count+1;
        accuracy.append((count/len(labels)));
    average_acc = np.mean(accuracy);
    squares = [];
    M = len(accuracy);
    for pm in range(0,M):
        squares.append(((accuracy[pm])**2) - ((average_acc)**2))
    sd_acc = (((1/(M-1))*np.sum(squares))**(1/2));

    #sd_acc = np.std(accuracy);
    return average_acc,sd_acc;
```

In [328]:

```
ACC = np.zeros((Ny,Nc));
SD = np.zeros((Ny,Nc));
for i in range(0,Ny):
    for j in range(0,Nc):
        #SD[i][j],ACC[i][j] = acc(values[i],values[j]);
        a,b = acc(values[j],values[i])
        SD[i][j] = b
        ACC[i][j] = a


```

```
C:\Users\Hp\Anaconda3\lib\site-packages\sklearn\utils\validation.py:578: Dat
aConversionWarning: A column-vector y was passed when a 1d array was expecte
d. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

In [329]:

```
1 ACC
```

Out[329]:

```
array([[0.39346405, 0.39346405, 0.39346405, ..., 0.80915033, 0.80915033,
        0.82026144],
       [0.39346405, 0.39346405, 0.39346405, ..., 0.80915033, 0.80915033,
        0.80915033],
       [0.39346405, 0.39346405, 0.39346405, ..., 0.80915033, 0.80915033,
        0.82026144],
       ...,
       [0.39346405, 0.39346405, 0.39346405, ..., 0.43856209, 0.43856209,
        0.43856209],
       [0.39346405, 0.39346405, 0.39346405, ..., 0.42745098, 0.42745098,
        0.42745098],
       [0.39346405, 0.39346405, 0.39346405, ..., 0.42745098, 0.42745098,
        0.42745098]])
```

In [330]:

```
1 SD
```

Out[330]:

```
array([[0.01023038, 0.01023038, 0.01023038, ..., 0.08372011, 0.08372011,
        0.06056782],
       [0.01023038, 0.01023038, 0.01023038, ..., 0.08372011, 0.08372011,
        0.08372011],
       [0.01023038, 0.01023038, 0.01023038, ..., 0.08372011, 0.08372011,
        0.09920333],
       ...,
       [0.01023038, 0.01023038, 0.01023038, ..., 0.04942101, 0.04942101,
        0.04942101],
       [0.01023038, 0.01023038, 0.01023038, ..., 0.03678467, 0.03678467,
        0.03678467],
       [0.01023038, 0.01023038, 0.01023038, ..., 0.03678467, 0.03678467,
        0.03678467]])
```

In [331]:

```
1 best_fit = np.where(ACC == ACC.max())
```

In [332]:

```
1 best_fit = np.column_stack((best_fit[0],best_fit[1]))
```

In [333]:

```
1 best_fit
```

Out[333]:

```
array([[21, 37],
       [22, 32],
       [22, 33],
       [22, 34],
       [24, 28],
       [25, 27]], dtype=int64)
```

In [345]:

```
ACC.max()
```

Out[345]:

```
0.85359477124183
```

In [348]:

```
best_fit = np.where(ACC == ACC.max())
best_fit = np.column_stack((best_fit[0],best_fit[1]))
len(best_fit)

```

Out[348]:

```
6
```

In [355]:

```
SD_best = []
for i in range (0,len(best_fit)):
    sp = best_fit[i,:]
    SD_best.append(SD[sp[0]][sp[1]])
SDmin = np.min(SD_best)
for i in range(0,50):
    for j in range(0,50):
        if(ACC[i][j] == ACC.max() and SD[i][j] == SDmin):
            iind = i;
            jind = j;
            break;
best_C = c[jind];
best_y = y[iind];
```

In [356]:

```
best_C
```

Out[356]:

```
8.286427728546842
```

In [357]:

```
best_y
```

Out[357]:

```
0.49417133613238334
```

In [358]:

```
SDmin
```

Out[358]:

```
0.08501130083463712
```

In [359]:

```
ACC.max()
```

Out[359]:

```
0.85359477124183
```

Problem 2 (c)

Without shuffling the data

In [365]:

```
ACC1 = np.zeros((Ny,Nc,20));
SD1 = np.zeros((Ny,Nc,20));
for t in range(0,20):
    for i in range(0,Ny):
        for j in range(0,Nc):
            a,b = acc(values[j],values[i])
            SD1[i][j][t] = b
            ACC1[i][j][t] = a


```

```
C:\Users\Hp\Anaconda3\lib\site-packages\sklearn\utils\validation.py:578: Dat
aConversionWarning: A column-vector y was passed when a 1d array was expecte
d. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

In [370]:

```
results = np.zeros((20,4))
```

In [371]:

```
for t in range(0,20):
    ACC = ACC1[:,:,t]
    SD = SD1[:,:,t]
    best_fit = np.where(ACC == ACC.max())
    best_fit = np.column_stack((best_fit[0],best_fit[1]))
    SD_best = []
    for i in range (0,len(best_fit)):
        sp = best_fit[i,:]
        SD_best.append(SD[sp[0]][sp[1]])
    SDmin = np.min(SD_best)
    for i in range(0,50):
        for j in range(0,50):
            if(ACC[i][j] == ACC.max() and SD[i][j] == SDmin):
                iind = i;
                jind = j;
                break;
    best_C = c[jind];
    best_y = y[iind];
    results[t,0] = best_c;
    results[t,1] = best_y;
    results[t,2] = ACC.max();
    results[t,3] = SDmin;

```

In [372]:

```
results
```

Out[372]:

```
array([[33.93221772,  0.49417134,  0.85359477,  0.0850113 ],
       [33.93221772,  0.49417134,  0.85359477,  0.0850113 ],
       [33.93221772,  0.49417134,  0.85359477,  0.0850113 ],
       [33.93221772,  0.49417134,  0.85359477,  0.0850113 ],
       [33.93221772,  0.49417134,  0.85359477,  0.0850113 ],
       [33.93221772,  0.49417134,  0.85359477,  0.0850113 ],
       [33.93221772,  0.49417134,  0.85359477,  0.0850113 ],
       [33.93221772,  0.49417134,  0.85359477,  0.0850113 ],
       [33.93221772,  0.49417134,  0.85359477,  0.0850113 ],
       [33.93221772,  0.49417134,  0.85359477,  0.0850113 ],
       [33.93221772,  0.49417134,  0.85359477,  0.0850113 ],
       [33.93221772,  0.49417134,  0.85359477,  0.0850113 ],
       [33.93221772,  0.49417134,  0.85359477,  0.0850113 ],
       [33.93221772,  0.49417134,  0.85359477,  0.0850113 ],
       [33.93221772,  0.49417134,  0.85359477,  0.0850113 ],
       [33.93221772,  0.49417134,  0.85359477,  0.0850113 ],
       [33.93221772,  0.49417134,  0.85359477,  0.0850113 ],
       [33.93221772,  0.49417134,  0.85359477,  0.0850113 ],
       [33.93221772,  0.49417134,  0.85359477,  0.0850113 ],
       [33.93221772,  0.49417134,  0.85359477,  0.0850113 ]])
```

In [377]:

```
print("The final C = ",results[0][0]);
print("The final y = ",results[0][1]);
print("The final mean = ",results[0][2]);
print("The final SD = ",results[0][3]);

```

```
The final C =  33.9322177189533
The final y =  0.28117686979742307
The final mean =  0.8764705882352942
The final SD =  0.024279968132030615
```

With Shuffling of data

In [375]:

```
n_cv = 5;
def acc1(C_value, g_value):
    accuracy = [];
    cv = StratifiedKFold(n_cv,shuffle = True)
    clf = svm.SVC(C = C_value,kernel='rbf',gamma = g_value)
    for train_index, test_index in cv.split(train_data, label_tr):
        count = 0;
        X_train, X_test = train_data[train_index], train_data[test_index];
        clf.fit(X_train,train_data_wlabels[train_index,2:3])
        labels = clf.predict(X_test)
        sample = train_data_wlabels[test_index,2:3]
        for i in range(0,len(labels)):
            if(labels[i] == sample[i]):
                count = count+1;
        accuracy.append((count/len(labels)));
    average_acc = np.mean(accuracy);
    squares = [];
    M = len(accuracy);
    for pm in range(0,M):
        squares.append(((accuracy[pm])**2) - ((average_acc)**2))
    sd_acc = (((1/(M-1))*np.sum(squares))**(1/2));

    #sd_acc = np.std(accuracy);
    return average_acc,sd_acc;
```

In [376]:

```
ACC1 = np.zeros((Ny,Nc,20));
SD1 = np.zeros((Ny,Nc,20));
for t in range(0,20):
    for i in range(0,Ny):
        for j in range(0,Nc):
            a,b = acc1(values[j],values[i])
            SD1[i][j][t] = b
            ACC1[i][j][t] = a
for t in range(0,20):
    ACC = ACC1[:,:,t]
    SD = SD1[:,:,t]
    best_fit = np.where(ACC == ACC.max())
    best_fit = np.column_stack((best_fit[0],best_fit[1]))
    SD_best = []
    for i in range (0,len(best_fit)):
        sp = best_fit[i,:]
        SD_best.append(SD[sp[0]][sp[1]])
    SDmin = np.min(SD_best)
    for i in range(0,50):
        for j in range(0,50):
            if(ACC[i][j] == ACC.max() and SD[i][j] == SDmin):
                iind = i;
                jind = j;
                break;
    best_C = c[jind];
    best_y = y[iind];
    results[t,0] = best_c;
    results[t,1] = best_y;
    results[t,2] = ACC.max();
    results[t,3] = SDmin;
```

```
C:\Users\Hp\Anaconda3\lib\site-packages\sklearn\utils\validation.py:578: Dat
aConversionWarning: A column-vector y was passed when a 1d array was expecte
d. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

In [379]:

```
1 results
```

Out[379]:

```
array([[3.39322177e+01, 2.81176870e-01, 8.76470588e-01, 2.42799681e-02],
       [3.39322177e+01, 2.81176870e-01, 8.66666667e-01, 1.00921678e-01],
       [3.39322177e+01, 3.72759372e-01, 8.64705882e-01, 5.14952218e-02],
       [3.39322177e+01, 9.10298178e-02, 8.66666667e-01, 8.42541716e-02],
       [3.39322177e+01, 9.10298178e-02, 8.76470588e-01, 8.22335476e-02],
       [3.39322177e+01, 6.55128557e-01, 8.54901961e-01, 6.22032019e-02],
       [3.39322177e+01, 2.81176870e-01, 8.76470588e-01, 8.22335476e-02],
       [3.39322177e+01, 4.94171336e-01, 8.65359477e-01, 4.90413895e-02],
       [3.39322177e+01, 3.72759372e-01, 8.66013072e-01, 8.35093617e-02],
       [3.39322177e+01, 2.81176870e-01, 8.64705882e-01, 3.32948644e-02],
       [3.39322177e+01, 2.12095089e-01, 8.66013072e-01, 6.23489674e-02],
       [3.39322177e+01, 3.72759372e-01, 8.66013072e-01, 9.22877206e-02],
       [3.39322177e+01, 3.90693994e-02, 8.66013072e-01, 9.22877206e-02],
       [3.39322177e+01, 6.55128557e-01, 8.65359477e-01, 4.90413895e-02],
       [3.39322177e+01, 3.72759372e-01, 8.66666667e-01, 8.42541716e-02],
       [3.39322177e+01, 2.81176870e-01, 8.76470588e-01, 6.06295028e-02],
       [3.39322177e+01, 6.55128557e-01, 8.65359477e-01, 2.93572479e-02],
       [3.39322177e+01, 3.72759372e-01, 8.77777778e-01, 7.24355823e-02],
       [3.39322177e+01, 6.86648845e-02, 8.66666667e-01, 8.42541716e-02],
       [3.39322177e+01, 8.68511374e-01, 8.66013072e-01, 8.35093617e-02]])
```

In [380]:

```
1 max_accuracy = np.max(results[:,2])
```

In [381]:

```
1 max_accuracy
```

Out[381]:

```
0.8777777777777779
```

In [382]:

```
1 for i in range(0,len(results)):
2     if(results[i][2] == max_accuracy):
3         best_c = results[i][0];
4         best_y = results[i][1];
5         best_mean = results[i][2];
6         best_sd = results[i][3];
7
```

In [383]:

```
print("The final C = ",best_c);
print("The final y = ",best_y);
print("The final mean = ",best_mean);
print("The final SD = ",best_sd);
```

```
The final C =  33.9322177189533
The final y =  0.3727593720314938
The final mean =  0.8777777777777779
The final SD =  0.07243558228002855
```

Problem 2 d

In [396]:

```
accuracy = 0;
count = 0;
clf = svm.SVC(C = best_c,kernel='rbf',gamma = best_y)
clf.fit(train_data,label_tr)
labels = clf.predict(test_data)
for i in range(0,len(label_te)):
    if(labels[i] == label_te[i]):
        count = count+1;
accuracy= ((count/len(labels))*100);

```

```
C:\Users\Hp\Anaconda3\lib\site-packages\sklearn\utils\validation.py:578: Dat
aConversionWarning: A column-vector y was passed when a 1d array was expecte
d. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

In [397]:

```
accuracy
```

Out[397]:

```
78.65168539325843
```

Problem 1 a

In [398]:

```
from sklearn import svm
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [399]:

```
dataset1_x = pd.DataFrame.as_matrix(pd.read_csv('train_x.csv'))
dataset1_y = pd.read_csv('train_y.csv')

dataset1_y = dataset1_y.values;
dataset1_y = dataset1_y.ravel();

dataset2_x = pd.DataFrame.as_matrix(pd.read_csv('train_x2.csv'))
dataset2_y = pd.read_csv('train_y2.csv')

dataset2_y = dataset2_y.values;
dataset2_y = dataset2_y.ravel();
```

```
C:\Users\Hp\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: FutureWarni
ng: Method .as_matrix will be removed in a future version. Use .values inste
ad.
  """Entry point for launching an IPython kernel.
C:\Users\Hp\Anaconda3\lib\site-packages\ipykernel_launcher.py:7: FutureWarni
ng: Method .as_matrix will be removed in a future version. Use .values inste
ad.
  import sys
```

In [404]:

```
def accuracy_classification(labels,length,dataset_y):
    count = 0;
    for i in range(0,length):
        if(labels[i] == dataset_y[i]):
            count += 1;
    accuracy = (count/length)*100;
    return accuracy;
```

In [405]:

```
def plotSVMBoundaries(training, label_train, classifier, support_vectors = []):

    #Plot the decision boundaries and data points for minimum distance to
    #class mean classifier
    #
    # training: traning data
    # label_train: class lables correspond to training data
    # classifier: sklearn classifier model, must have a predict() function
    #
    # Total number of classes
    nclass =  max(np.unique(label_train))

    # Set the feature range for ploting
    max_x = np.ceil(max(training[:, 0])) + 0.01
    min_x = np.floor(min(training[:, 0])) - 0.01
    max_y = np.ceil(max(training[:, 1])) + 0.01
    min_y = np.floor(min(training[:, 1])) - 0.01

    xrange = (min_x, max_x)
    yrange = (min_y, max_y)

    # step size for how finely you want to visualize the decision boundary.
    inc = 0.005

    # generate grid coordinates. this will be the basis of the decision
    # boundary visualization.
    (x, y) = np.meshgrid(np.arange(xrange[0], xrange[1]+inc/100, inc), np.arange(yrange

    # size of the (x, y) image, which will also be the size of the
    # decision boundary image that is used as the plot background.
    image_size = x.shape
    xy = np.hstack( (x.reshape(x.shape[0]*x.shape[1], 1, order='F'), y.reshape(y.shape

    # distance measure evaluations for each (x,y) pair.
    pred_label = classifier.predict(xy)

    # reshape the idx (which contains the class label) into an image.
    decisionmap = pred_label.reshape(image_size, order='F')

    #turn on interactive mode
    plt.figure()
    #plt.ion()

    #show the image, give each coordinate a color according to its class label
    plt.imshow(decisionmap, extent=[xrange[0], xrange[1], yrange[0], yrange[1]], origi

    unique_labels = np.unique(label_train)
    # plot the class training data.
    plt.plot(training[label_train == unique_labels[0], 0],training[label_train == uniq
    plt.plot(training[label_train == unique_labels[1], 0],training[label_train == uniq
    if nclass == 3:
        plt.plot(training[label_train == unique_labels[2], 0],training[label_train ==

    # include legend for training data
    if nclass == 3:
        l = plt.legend(('Class 1', 'Class 2', 'Class 3'), loc=2)
    else:
        l = plt.legend(('Class 1', 'Class 2'), loc=2)
    plt.gca().add_artist(l)
```

```
60
61      # plot support vectors
62      if len(support_vectors)>0:
63          sv_x = support_vectors[:, 0]
64          sv_y = support_vectors[:, 1]
65          plt.scatter(sv_x, sv_y, s = 100, c = 'blue')
66
67      plt.show()
```

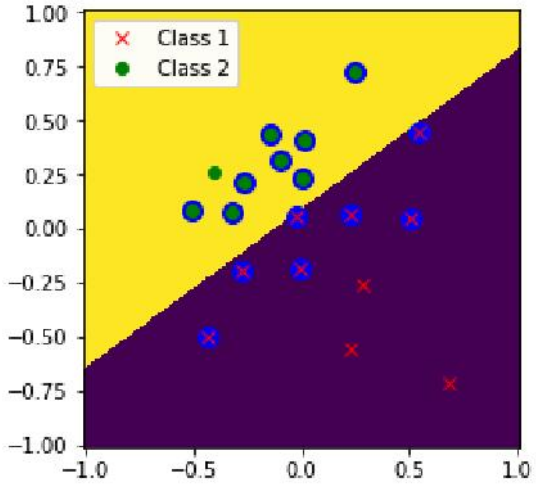
In [412]:

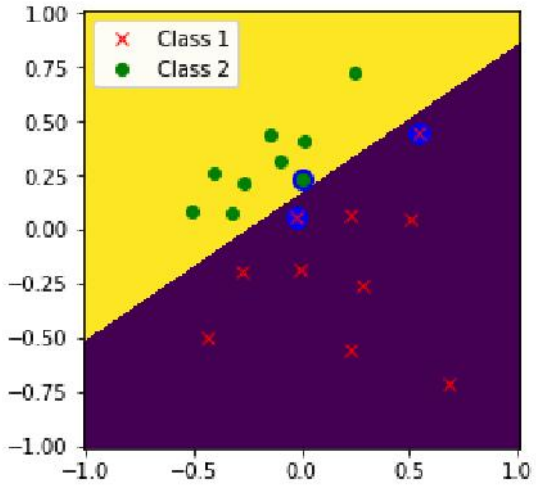
```
1
2  def linear_SVM(C_value):
3      clf = svm.SVC(C = C_value,kernel='linear')
4      clf.fit(dataset1_x, dataset1_y)
5      labels = clf.predict(dataset1_x)
6      weight0 = clf.intercept_;
7      weight12 = clf.coef_;
8      weights = [];
9      weights.append(weight0);
10     weights.append(weight12);
11     sv = clf.support_vectors_;
12     decision_boundary = clf.decision_function;
13     return clf,sv,weights,labels,decision_boundary;
14
15 def RBF_Kernel(C_value):
16     clf = svm.SVC(C = C_value,kernel='rbf')
17     clf.fit(dataset2_x, dataset2_y)
18     labels = clf.predict(dataset2_x)
19     sv = clf.support_vectors_
20     return clf,sv,labels;
```

In [413]:

```
#1a
clf1,sv1,weights1,labels1,decision_boundary1 = np.array(linear_SVM(1));
accuracy1 = accuracy_classification(labels1,len(dataset1_y),dataset1_y);

clf100,sv100,weights100,labels100,decision_boundary100 = np.array(linear_SVM(100));
accuracy100 = accuracy_classification(labels100,len(dataset1_y),dataset1_y);

print("Accuracy when C = 1 is",accuracy1);
print("Accuracy when C = 100 is",accuracy100);

#plot
plot_1 = plotSVMBoundaries(dataset1_x,dataset1_y,clf1,sv1);
plot_100 = plotSVMBoundaries(dataset1_x,dataset1_y,clf100,sv100)


```

```
Accuracy when C = 1 is 100.0
Accuracy when C = 100 is 100.0
```





The value of C doesnt affect the classification in case of linear kenel. The accuracy for both C =1 and C =100 are the same.

problem 1 b

In [421]:

```
1
2 print("The weights when C = 100 are",weights100 );
3 print("The support vectors when C = 100 are",sv100);
4
```

```
The weights when C = 100 are [array([-1.79836766]), array([[-7.11966384, 10.
40264821]])]
The support vectors when C = 100 are [[-0.023855   0.06042  ]
 [ 0.54579    0.45029  ]
 [ 0.0064864  0.23394  ]]
```

Problem 1 c

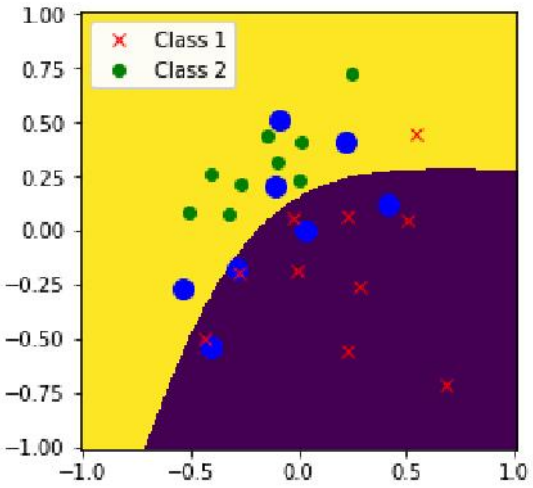
The g(x) is given by = -1.798366 + (-7.11966384)x1 + (10.40264821)x2 Yes, they lie on the boundary of the margin.

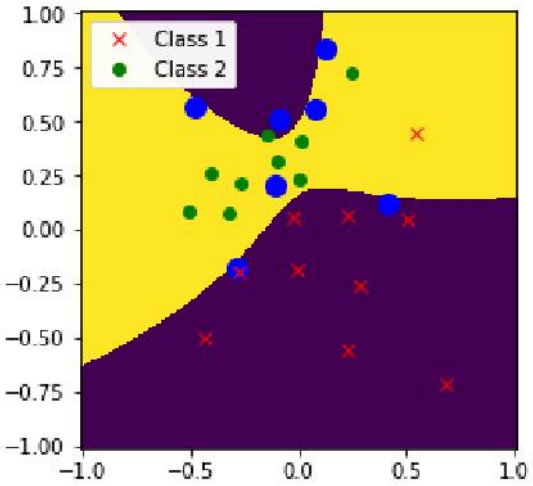
Problem 1 d

In [416]:

```
#1d
clf50,sv50,labels50 = np.array(RBF_Kernel(50));
accuracy50 = accuracy_classification(labels50,len(dataset2_y),dataset2_y);

clf5000,sv5000,labels5000 = np.array(RBF_Kernel(5000));
accuracy5000 = accuracy_classification(labels5000,len(dataset2_y),dataset2_y);

print("Accuracy when C = 50 is",accuracy50);
print("Accuracy when C = 5000 is",accuracy5000);

#plot
plot_50 = plotSVMBoundaries(dataset1_x,dataset1_y,clf50,sv50);
plot_5000 = plotSVMBoundaries(dataset1_x,dataset1_y,clf5000,sv5000)
```

```
Accuracy when C = 50 is 94.73684210526315
Accuracy when C = 5000 is 100.0
```





The decision boundaries are non- linear. the main differnece between the two decison boundaries are that tey are not in the same eregion.

Problem 1 e

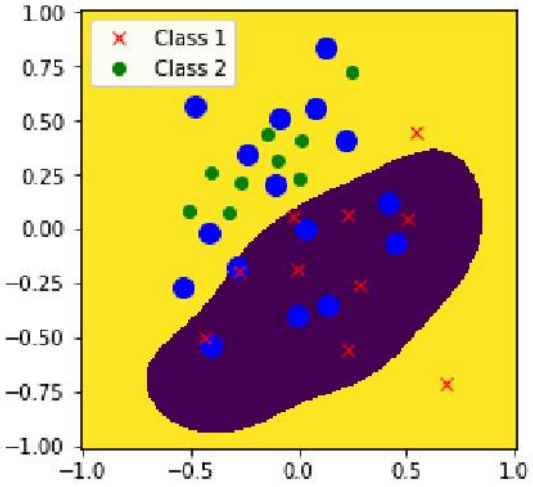
In [417]:

```
def RBF_Kernel1(gamma_value):
    clf = svm.SVC(kernel='rbf',gamma = gamma_value)
    clf.fit(dataset2_x, dataset2_y)
    labels = clf.predict(dataset2_x)
    sv = clf.support_vectors_
    return clf,sv,labels;
```

In [418]:

```
clf10,sv10,labels10 = np.array(RBF_Kernel1(10));
accuracy10 = accuracy_classification(labels10,len(dataset2_y),dataset2_y);
print("Accuracy when gamma = 10 is",accuracy10);
#plot
plot_10 = plotSVMBoundaries(dataset1_x,dataset1_y,clf10,sv10);
```

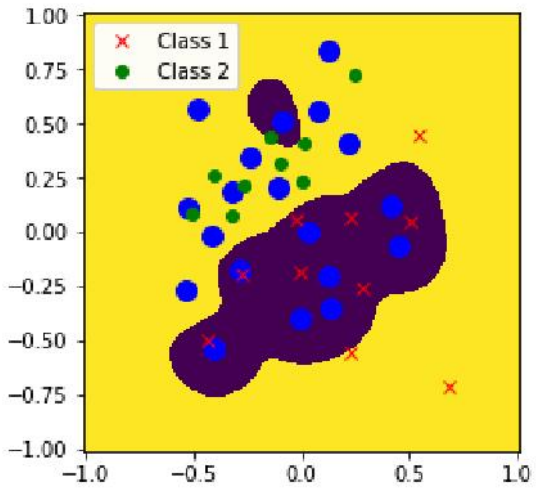
```
Accuracy when gamma = 10 is 94.73684210526315
```



In [419]:

```
clf50,sv50,labels50 = np.array(RBF_Kernel1(50));
accuracy50 = accuracy_classification(labels50,len(dataset2_y),dataset2_y);
print("Accuracy when gamma = 50 is",accuracy50);
#plot
plot_50 = plotSVMBoundaries(dataset1_x,dataset1_y,clf50,sv50);

```

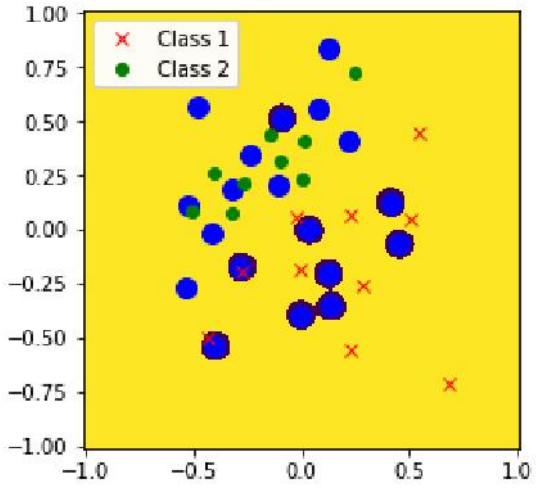
```
Accuracy when gamma = 50 is 100.0
```



In [420]:

```
clf500,sv500,labels500 = np.array(RBF_Kernel1(500));
accuracy500 = accuracy_classification(labels500,len(dataset2_y),dataset2_y);
print("Accuracy when gamma = 500 is",accuracy500);
#plot
plot_500 = plotSVMBoundaries(dataset1_x,dataset1_y,clf500,sv500);

```

```
Accuracy when gamma = 500 is 100.0
```



With the incr3ease in the gamma value, the deciosn regions are reducing drastically. Because of which, there is over fittign problem in the gamma= 500 plot.

In [ ]:

```
1
```