EE559 Homework 8

Jingquan Yan

USC ID: 1071912676

Email: jingquan@usc.edu

EE559 repository: Github

Problem 1:

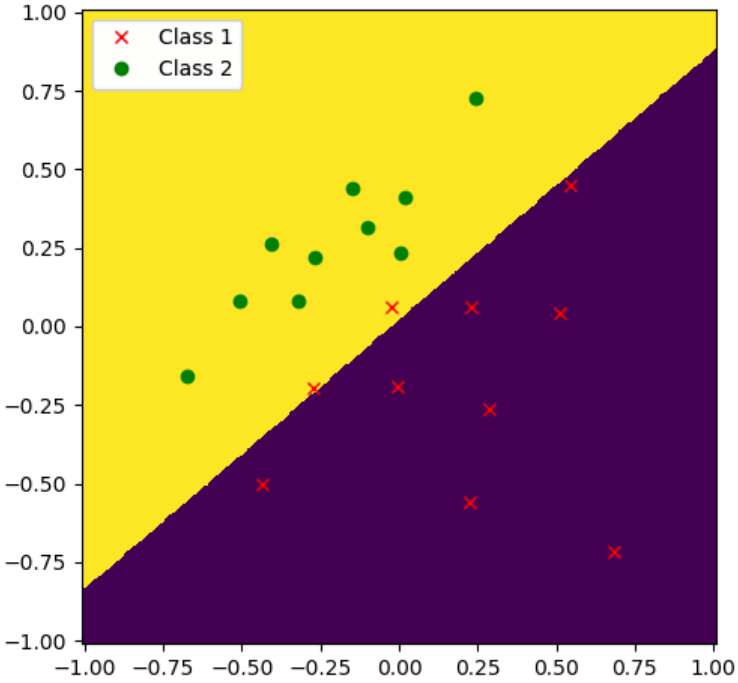
(a):

The slack parameter **C** is introduced to trade-off between the misclassified penalty and the soft-margin. For non-linearly separable dataset, as we increase the value of **C**, the penalty for misclassified data increases and in other words, the soft-margin gets relatively smaller(rigorous) and vice versa.

We plug in **C** as 1, 5, 10, 50, 100 and the respective accuracies are:

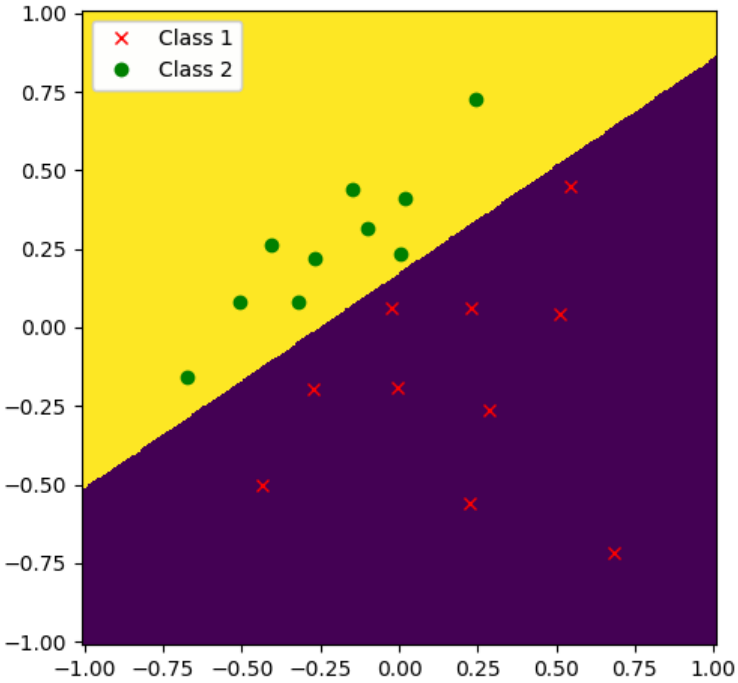
C	Accuracy
1	0.9
5	1
10	1
50	1
100	1

When **C=1**, the accuracy is 0.9 and the plot is as follows:




```
C:\Users\Yan\AppData\Local\Programs\Python\Python36\python.exe C:/Git/559/HW8/Problem_1_a.py
The accuracy when c=1 is:  0.9
```

When **C=100**, the accuracy is 1 and the plot is as follows:

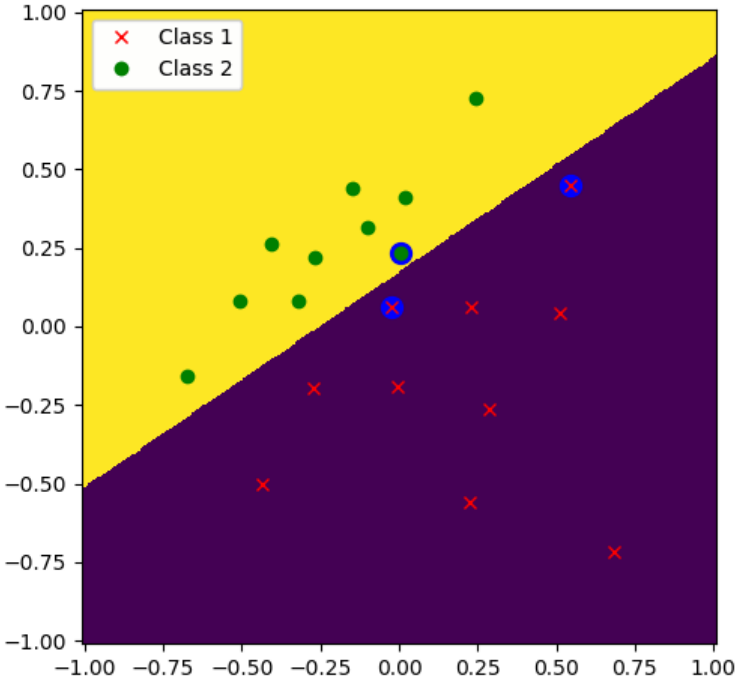



```
C:\Users\Yan\AppData\Local\Programs\Python\Python36\python.exe C:/Git/559/HW8/Problem_1_a.py
The accuracy when c=100 is:  1.0
```

Explanation: The soft-margin gets larger as the slack parameter **C** decreases and this give rise to more misclassified data points. As we increases **C**, the margin gets smaller and it ends up with less misclassified data points.

(b):

When **C=100**, the plot looks like the following ad the support vectors are circled with blue.



The accuracy, weight vector and decision boundary equation are:

```
C:\Users\Yan\AppData\Local\Programs\Python\Python36\python.exe C:/Git/559/HW8/Problem_1_a.py
The accuracy when c=100 is:  1.0
The support vector is: ( 10.402648206879697 -7.119663837669007 -1.7983676588125648 )
The decision boundary equation is :  -7.119663837669007 x1+ 10.402648206879697 x2+ -1.7983676588125648 = 0
```

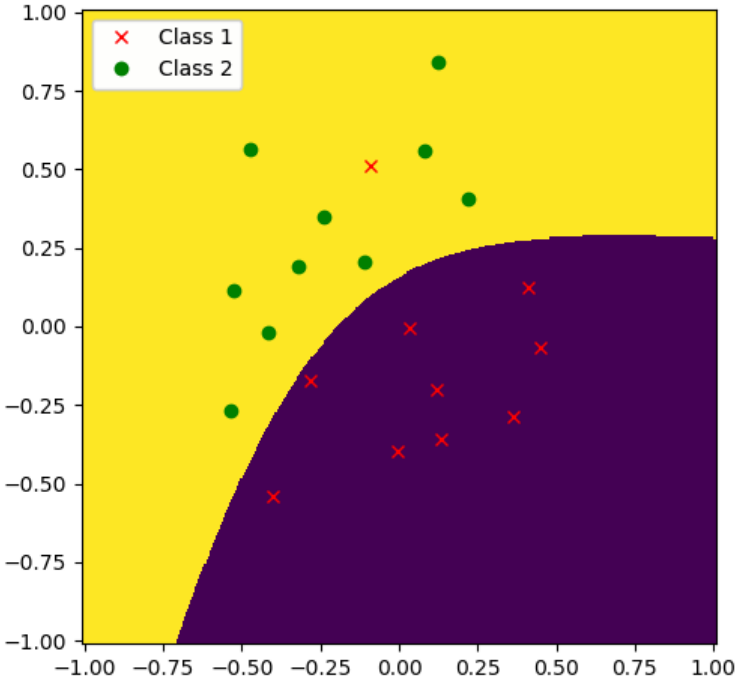
(c):

Plug in the individual support vectors to the decision boundary equation and we have the respective g(x) values are **[-1, -1, 0.589]**. So we can say that the first two values are on the boundary (correctly classified) and the last one is in the margin (misclassified).

```
C:\Users\Yan\AppData\Local\Programs\Python\Python36\python.exe C:/Git/559/HW8/Problem_1_a.py
The accuracy when c=100 is:  1.0
The support vector is: ( 10.402648206879697 -7.119663837669007 -1.7983676588125648 )
The decision boundary equation is :  -7.119663837669007 x1+ 10.402648206879697 x2+ -1.7983676588125648 = 0
The g(x) value of the support vectors are:  [-1.0000000733052992, -1.000000523698073, 0.5890468751882154]
```

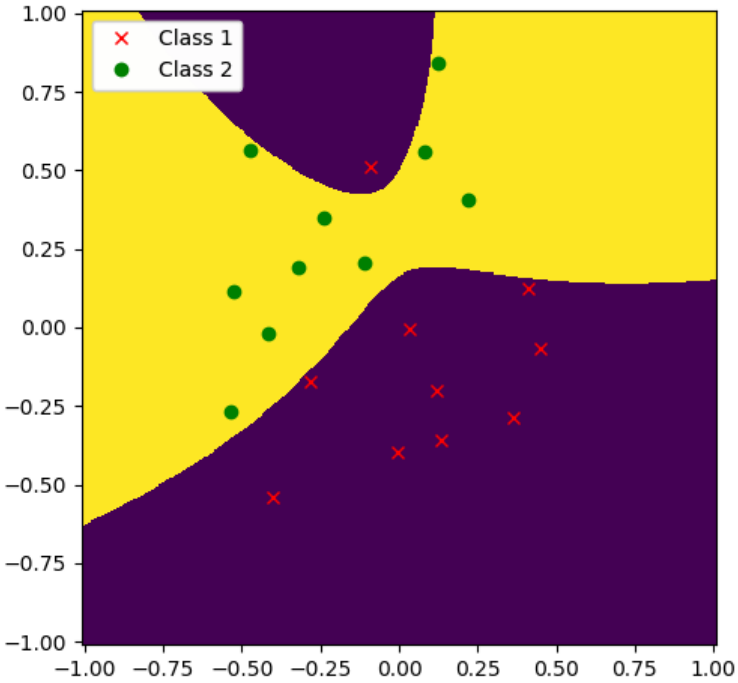
(d):

With RBF kernel and **C=50**, we obtain the **accuracy=0.95**.




```
C:\Users\Yan\AppData\Local\Programs\Python\Python36\python.exe C:/Git/559/HW8/Problem_1_a.py
The SVM accuracy when c=50 with rbf is:  0.95
```

With RBF kernel and **C=5000**, we obtain the **accuracy=1**.



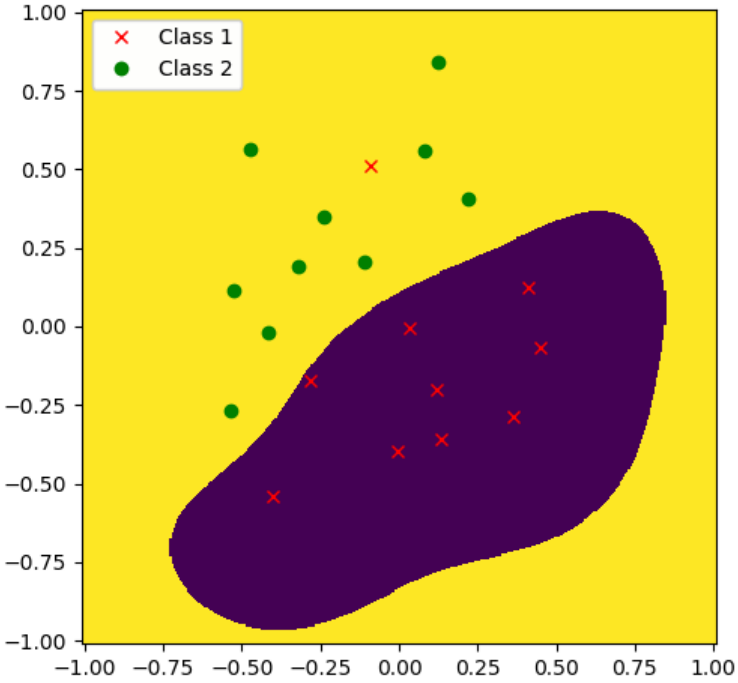

```
C:\Users\Yan\AppData\Local\Programs\Python\Python36\python.exe C:/Git/559/HW8/Problem_1_a.py
The SVM accuracy when c=5000 with rbf is:  1.0
```

Explanation: both **C=50** and **C=5000** come up with non-linear boundaries because the feature spaces are initially mapped to higher dimensions and then reduced into 2D.

As the **C** gets larger, the classification gets more rigorous and thus the boundary gets more fitting. This improvement comes along with the increasing of the model complexity.

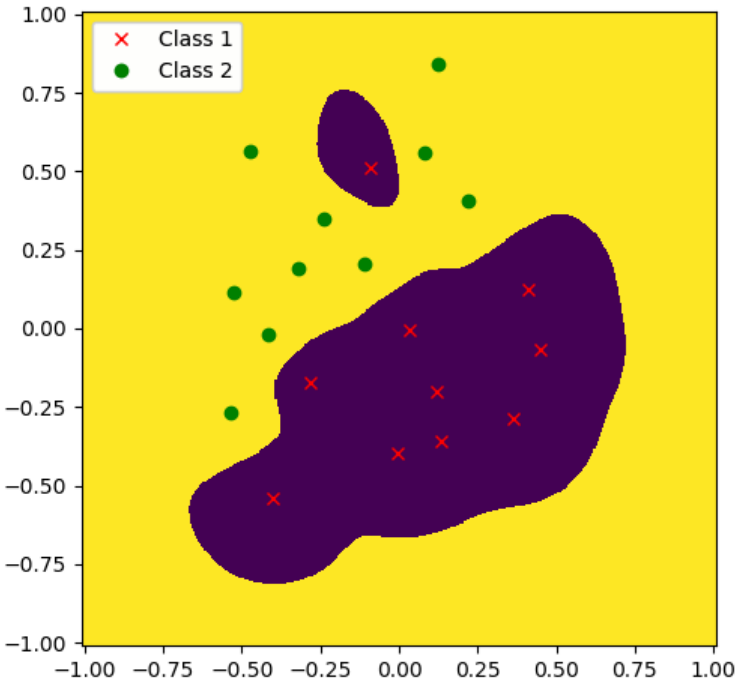
(e):

When **C='default', gamma=10**:



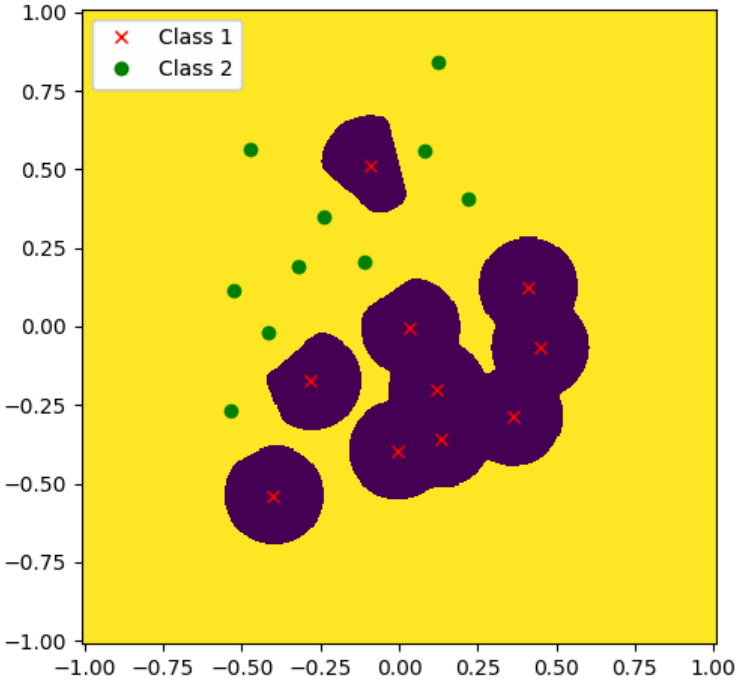

```
C:\Users\Yan\AppData\Local\Programs\Python\Python36\python.exe C:/Git/559/HW8/Problem_1_a.py
The SVM accuracy when c=5000 with rbf is:  0.95
```

When **C='default', gamma=100**:




```
C:\Users\Yan\AppData\Local\Programs\Python\Python36\python.exe C:/Git/559/HW8/Problem_1_a.py
The SVM accuracy when c=5000 with rbf is:  1.0
```

When **C='default', gamma=1000**:




```
C:\Users\Yan\AppData\Local\Programs\Python\Python36\python.exe C:/Git/559/HW8/Problem_1_a.py
The SVM accuracy when c=5000 with rbf is:  1.0
```

Comment: For the same **C** and smaller **gamma** value, the classification is more tolerable with the misclassified data and the edges of the margin are more flat and smooth. As the **gamma** increases, the edge of the margin gets more complicated and rigorous with the misclassified data.

When **gamma=50**, the accuracy is not so desirable and not fitting very well. When **gamma=500**, the accuracy is ideal and the generalization capacity is great as well. However, as **gamma** increases to **5000**, there exists overfitting problem and the generalization capacity is reduced.

Problem 2:

(a):

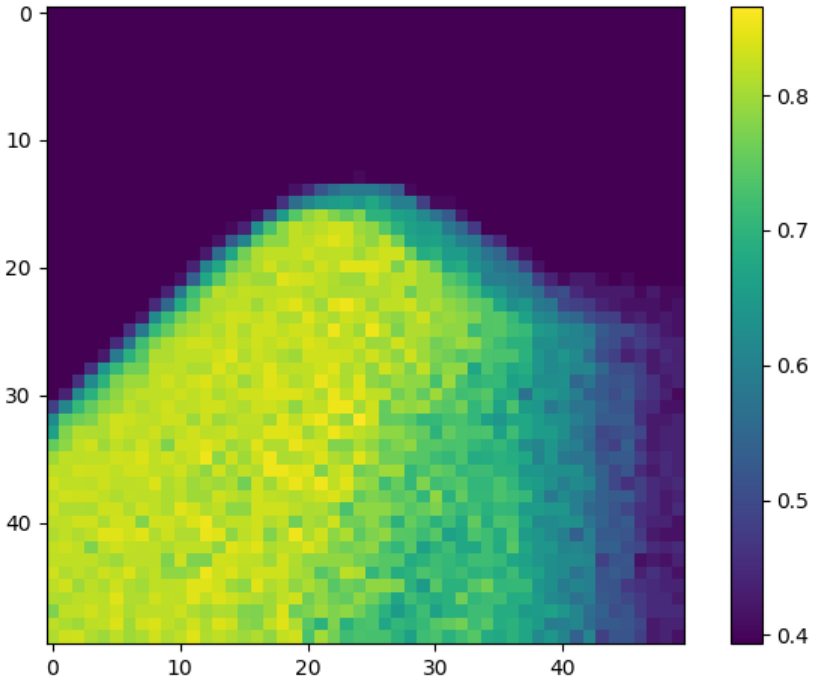
With RBF kernel, **γ=1, C=1**, the average cross-validation accuracy is **0.8203**:

```
C:\Users\Yan\AppData\Local\Programs\Python\Python36\python.exe C:/Git/559/HW8/Problem_2.py
The accuracy for each fold respectively:  [0.83333333 0.88888889 0.72222222 0.83333333 0.82352941]
The mean of cross-validation accuracy is: 0.8202614379084968
```

(b):

(i)

Visualization of ACC, the X-axis represents c and the Y-axis represents gamma.



(ii)

The criterion picking the best value is to choose the pair with the largest accuracy and when there are multiple pairs who share the same largest accuracy, we pick the one with the least deviation.

Here, we pick the best pair is **C=8.286 γ=0.869"** and the **best accuracy is 0.866** and **deviation is 0.08**.

```
C:\Users\Yan\AppData\Local\Programs\Python\Python36\python.exe C:/Git/559/HW8/Problem_2.py
The accuracy is the best when C = [8.28642773] and gamma = [0.86851137]
The best accuracy and deviation are: 0.8660130718954248 [0.08254465]
```

(c):

(i)

The values of the 20 chosen pairs of **C** and **γ** are:

```
[[2.68269580e+00 1.59985872e-01]
 [2.02358965e+00 1.15139540e+00]
 [8.28642773e+00 4.94171336e-01]
 [1.04811313e+02 2.81176870e-01]
 [6.55128557e-01 1.15139540e+00]
 [4.49843267e+01 1.20679264e-01]
 [2.12095089e-01 1.15139540e+00]
 [1.93069773e+01 3.72759372e-01]
 [1.09854114e+01 2.12095089e-01]
 [1.09854114e+01 2.81176870e-01]
 [1.45634848e+01 1.59985872e-01]
 [2.02358965e+00 8.68511374e-01]
 [7.54312006e+02 1.26485522e-02]
 [1.93069773e+01 3.72759372e-01]
 [1.84206997e+02 1.59985872e-01]
 [1.45634848e+01 2.94705170e-02]
 [6.25055193e+00 3.72759372e-01]
 [6.25055193e+00 6.55128557e-01]
 [1.93069773e+01 1.59985872e-01]
 [2.68269580e+00 2.81176870e-01]]
```

The corresponding **accuracy and deviation** are:

```
[[0.86535948 0.0899827 ]
 [0.86535948 0.05620155]
 [0.86470588 0.04605873]
 [0.86601307 0.102554  ]
 [0.86535948 0.05620155]
 [0.86535948 0.02625792]
 [0.86405229 0.09295303]
 [0.87777778 0.1237281 ]
 [0.87647059 0.02171666]
 [0.87647059 0.04130595]
 [0.8745098  0.08789817]
 [0.86666667 0.07535922]
 [0.86535948 0.07501833]
 [0.86601307 0.08254465]
 [0.86535948 0.10863052]
 [0.85555556 0.08314794]
 [0.86535948 0.07501833]
 [0.86535948 0.0899827 ]
 [0.86535948 0.07501833]
 [0.86601307 0.07469304]]
```

(ii)

Comment: For each run over the 20 iterations, the criterion picking the best value is to choose the pair with the largest accuracy and when there are multiple pairs who share the same largest accuracy, we pick the one with the least deviation.

The pick of best values runs more **reproducible** because theoretically, the SVM problem is a convex optimization and with strong duality property. The result of the dual optimization should be the global optimal result. So the parameter can converge and the pair of best **C** and **γ** may be more reproducible.

The best pair of **C** and **γ** is **[1.93069773e+01 3.72759372e-01]**

The corresponding **accuracy** and **deviation** is **[0.87777778 0.1237281]**

(d):

The classification accuracy on the test set is **0.8314**. This value is between **0.8778±0.1237** so it is in the one standard deviation from the cross-validation.

```
C:\Users\Yan\AppData\Local\Programs\Python\Python36\python.exe C:/Git/559/HW8/Problem_2.py
The accuracy of test dataset is: 0.8314606741573034
```