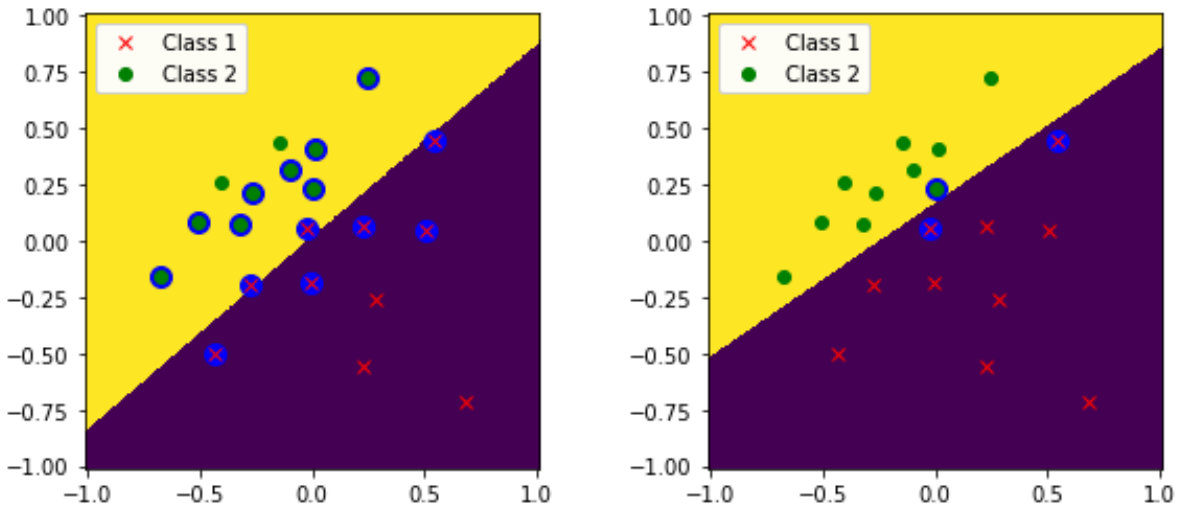
1-a

C is the parameter before slack variable called slack-variable parameter in the loss function of SVM learning without linearly separable assumption:

$$J = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i$$

The meaning of C is the tolerance for the mis-classified data or the data between boundary of margins. The value of C will also suggest the degree to which SVM fits the training data. Because if a data point x_i is wrongly classified or lies between boundary of margins, it will contribute $C * \xi_i$ to the cost function. The larger C is, the more it will contribute to. The plot is shown in Fig. 1



(a) C = 1, accuracy = 90%　　(b) C = 100, accuracy = 100%

Fig. 1 Decision boundary of 1-a with blue circle denoting support vectors

Fig. 1 shows that when C=1, the classification accuracy is 90%, and the number of support vectors is 15, when C=100, the classification accuracy is 100%, and the number of support vectors is 3 (blue circle denotes the support vectors). This means that the value of C could affect the width between the boundary of margins, the value of weights and the classification accuracy, which could be seen in the expression of loss function: if C is small, the impact of ξ_i is small, so the algorithm will try to find $\|w\|$ as small as possible, and the distance between two boundary of margins is $d = \frac{2}{\|w\|}$, so d is large when $\|w\|$ is small, and the penalty of mis-classification is not so large . If C is large, the impact of ξ_i is large, so the algorithm tends to find a decision boundary try to correctly classify all data points and leaving the number of points between two boundary of margins as small as possible, so the distance between two boundary of margins should be small, and they should classify the data points as correct as possible.

1-b

The support vectors are shown in Fig. 1 (b) with blue circle.

They are: [-0.023855, 0.06042], [0.54579, 0.45029], [0.0064864, 0.23394], where the first position denotes $x_1$and the second position denotes x_2, [x_1, x_2].

w_0=-1.79836766, w_1=-7.11966384, w_2=10.40264821

decision boundary equation is g(x) = -1.79836766 -7.11966384x_1 + 10.40264821x_2

1-c

for x = [-0.023855, 0.06042], g(x) = -1, x lies on the boundary of margin,

for x = [0.54579, 0.45029], g(x) = -1, x lies on the boundary of margin,

for x = [0.0064864, 0.23394], g(x) = 0. 589, x does not lie on the boundary of margin

As shown in 1-a, because the paneity of misclassification or the data point lies between two boundary of margins are not large enough, i.e. C is not large enough, so the algorithm choose a w with smaller norm which makes one of these support vectors not on the margin.

If C is large enough (in my approach, C is set to 10000), these support vectors should be on the boundary of margins.

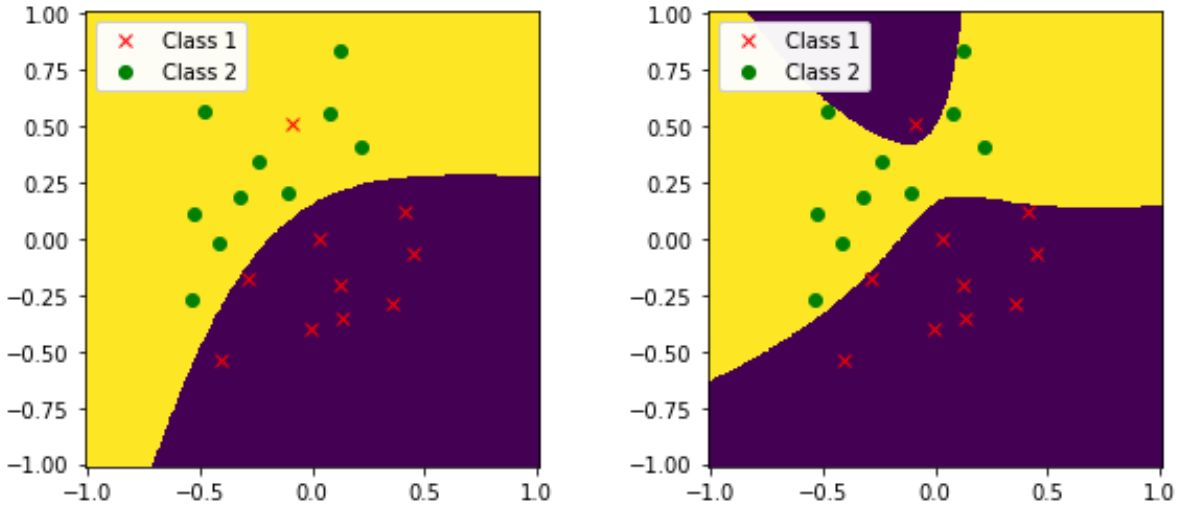
Through experiment, when C=10000, the support vectors are: [-0.023855, 0.06042], [0.54579, 0.45029] and [0.0064864, 0.23394], their corresponding g(x) for each of them are -0.99968838, -1.00015654, and 0.99984428, which means if C is large enough, the support vectors lies on the boundary of margins.

1-d

With C=50, the accuracy is 95%

With C=5000, the accuracy is 100%

The decision boundaries are shown in Fig. 2



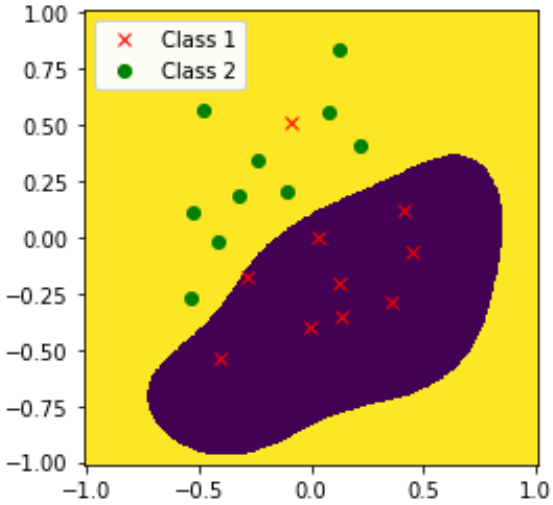
(a) C = 50, accuracy = 95% (b) C = 5000, accuracy = 100%

Fig. 2 Decision boundary of 1-d

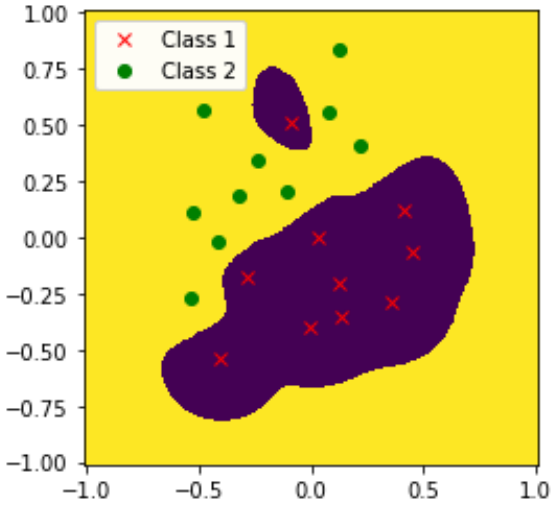
The decision boundary in this problem is nonlinear because of the kernel in this problem is rbf, which will calculate decision boundaries in higher dimensional space, and then project back into the original feature space, which cause the nonlinearity.

The difference of the two decision boundaries is caused by difference of C, as illustrated in 1-a, when C becomes larger, the SVM tend to fit the training set better because of the high cost of mis-classification and the data points between margins of boundaries, so a higher accuracy is obtained. When C is small, the cost of mis-classification is not so high, so there could be several mis-classified data points.

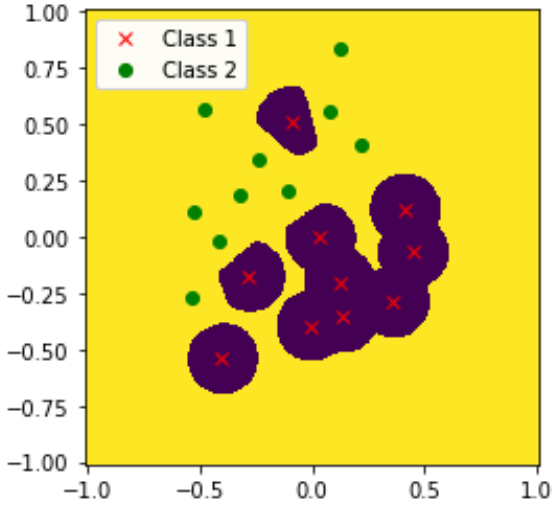
1-e



(a) gamma = 10, accuracy is 95%



(b) gamma = 50, accuracy is 100%



(c) gamma = 500, accuracy is 100%

Fig. 3 Decision boundary of 1-e

With gamma = 10, accuracy = 95%,

With gamma = 50, accuracy = 100%

With gamma = 500, accuracy = 100%

With the value of gamma becomes smaller, the area of within boundary (the purple part in the plot in Fig. 3) tend to cover the data points more tightly. Gamma could be explained as the width of rbf kernel, which also influence the area of decision region and is shown in the following formula.

$$K_{rbf}(x_i, x_j) = exp\{-\gamma \|x_i - x_j\|^2\}, \gamma > 0$$

When gamma is small, in this problem, gamma = 10, the width of K_{rbf} is relatively large, so the decision boundary is relatively large and could not fit the training set totally correctly. When gamma becomes larger, in this problem, gamma = 50, the width of K_{rbf} becomes smaller, so the decision boundary could fit the data points better and become narrower, but when gamma becomes too large, in this problem, when gamma = 500, the decision boundary will overfit the training point, looking like island surrounding each data points.

2-a

The average cross validation accuracy is 81.961%

2-b-(i)

The visualization of ACC matrix is shown in Fig. 4, both gamma and C are range from 1e-3 to 1e3 in log scale, 50 values for each parameter are generated. The x-axis stands for the index of C and y-axis stands for the index of gamma.

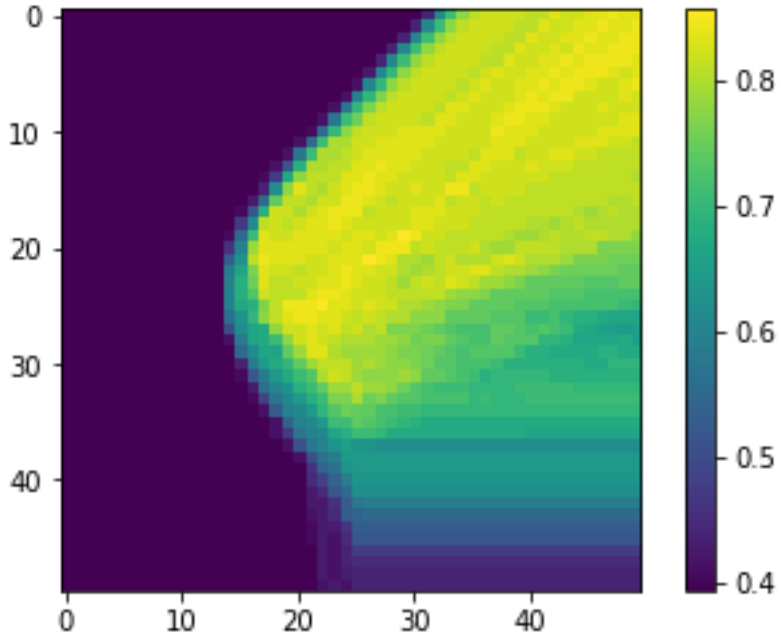


Fig. 4 Visualization of ACC

2-b-(ii)

Because the data set is random initialized every time, so the outcome may vary. After several experiment, in most of these experiments, there is a clear choice for best values of [γ, C], but there are also few results shows that there are several candidate pairs which get the maximal value at the same time.

In the latest trial of mine, there is a clear choice for the best values of [γ, C] as reported below:

Gamma = 0.212, C = 3.556, mean cross-validation accuracy is 0.8556, standard deviation is 0.0969

2-c-(i)

The 20 chosen pairs of [γ, C] is listed below:

t	γ	C
0	1.526	0.869
1	0.494	25.595
2	0.655	10.985
3	0.655	8.286
4	0.869	2.024
5	0.494	1.151
6	0.160	429.193
7	0.212	10.985
8	0.373	6.251
9	0.281	138.950
10	2.024	0.281
11	0.212	8.286
12	0.017	323.746
13	0.039	59.636
14	0.091	138.950
15	0.281	4.715
16	0.160	14.563
17	0.039	184.207
18	0.281	25.595
19	0.212	33.932

2-c-(ii)

Yes, because the training data and validation data are shuffled for several times and the experiment could get closer to the real situation where the test data is unknown to us and has similar distribution with training data, and because the training set and validation set are random shuffled for several times for each pair of [γ, C], so the performance of each pair could become more close to the theoretical performance or average performance to the whole training set, so the values are more well-defined.

The best value of [γ, C], its mean cross-validation accuracy and standard deviation are reported below:
Gamma = 0.212, C = 8.286, mean cross-validation accuracy is 0.8417, standard deviation is 0.0705

2-d

the accuracy on test set is 77.53%

Yes, this estimate is within approximately 1 standard deviation of your mean cross-validation accuracy from (c) (ii), actually it is 0.941 standard deviation of mean cross-validation accuracy from (c) (ii).