

**Sharif University of Technology**

**Convex Optimization Project**

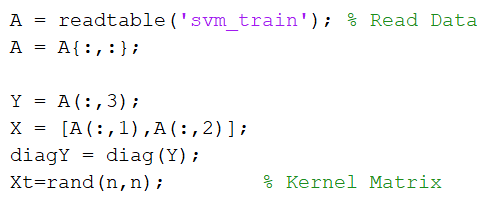
**Instructor: Dr. Babazadeh**

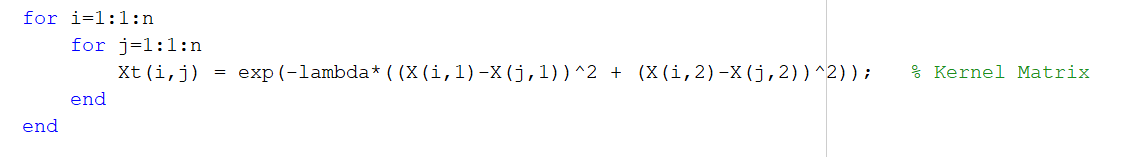
**Aliakbar Mahmoodzadeh**

**98106904**

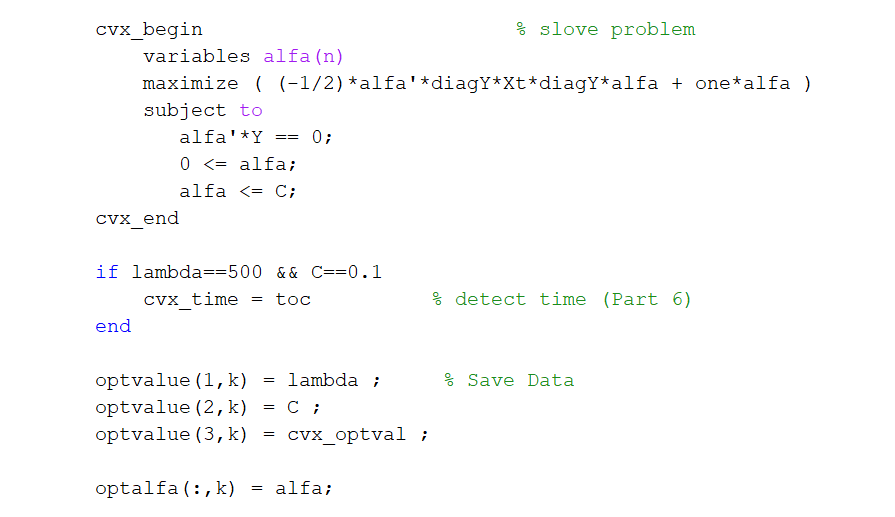
Part 1)

First, we need to extract the data from the given file and prepare the matrices needed to form the binary equation according to the following form.

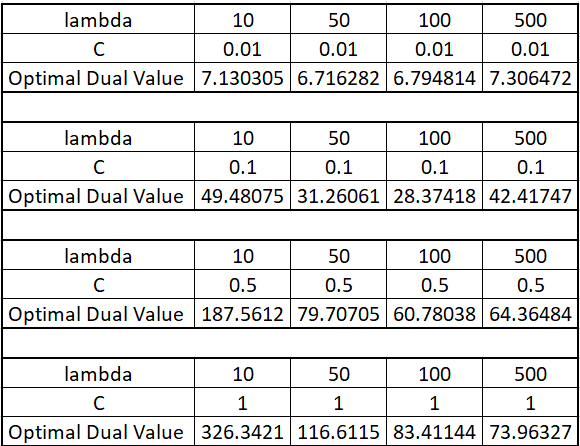




As follows, we solve the dual problem using cvx and store the results in matrices.



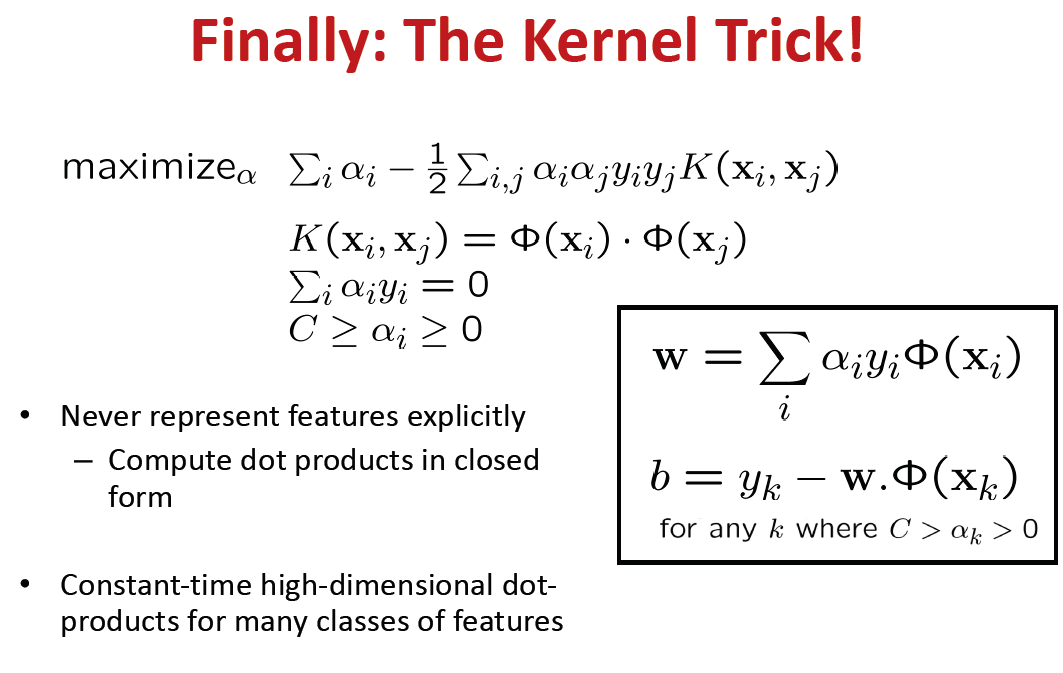
Finally, the optimal answer for each situation is obtained as follows:



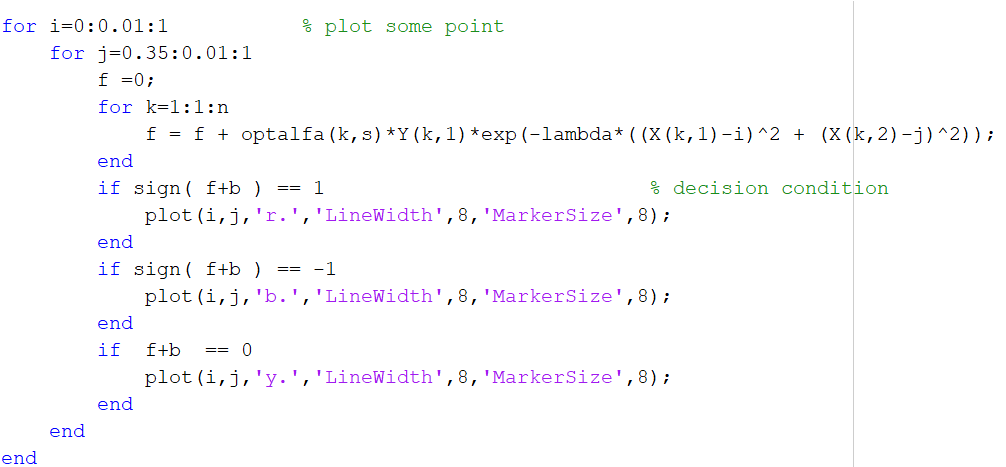
Part 2)

For the purpose of this part, we consider a large number of points for each state and label each of the points according to the values ​​obtained from the first part. Then we plot these points on the page and see the classification result. We do this using the following relationship and the following code.

The reason for using (1, 2) Y or the second data for each state is because in all states this data number has the required condition which is given below.

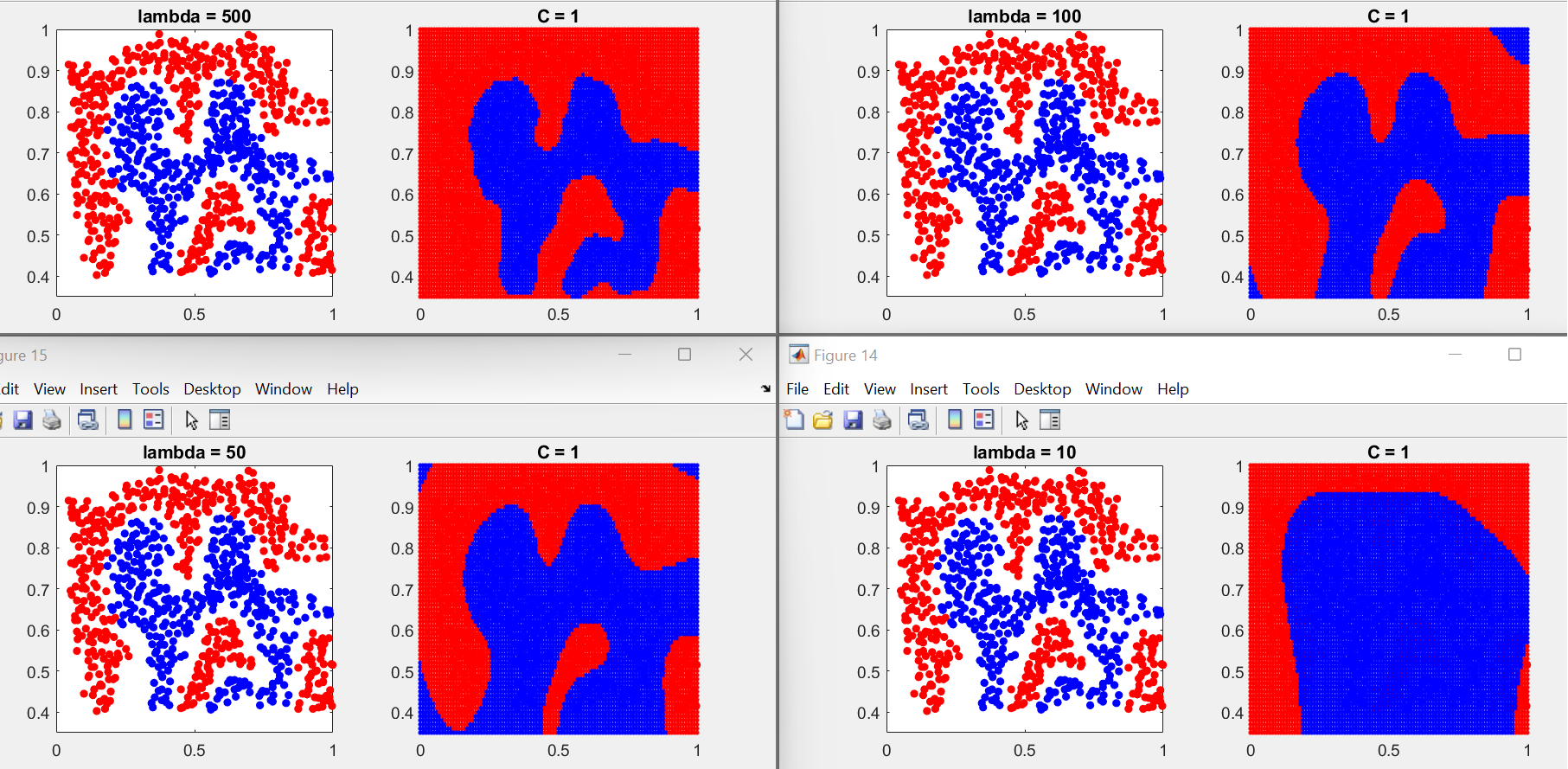


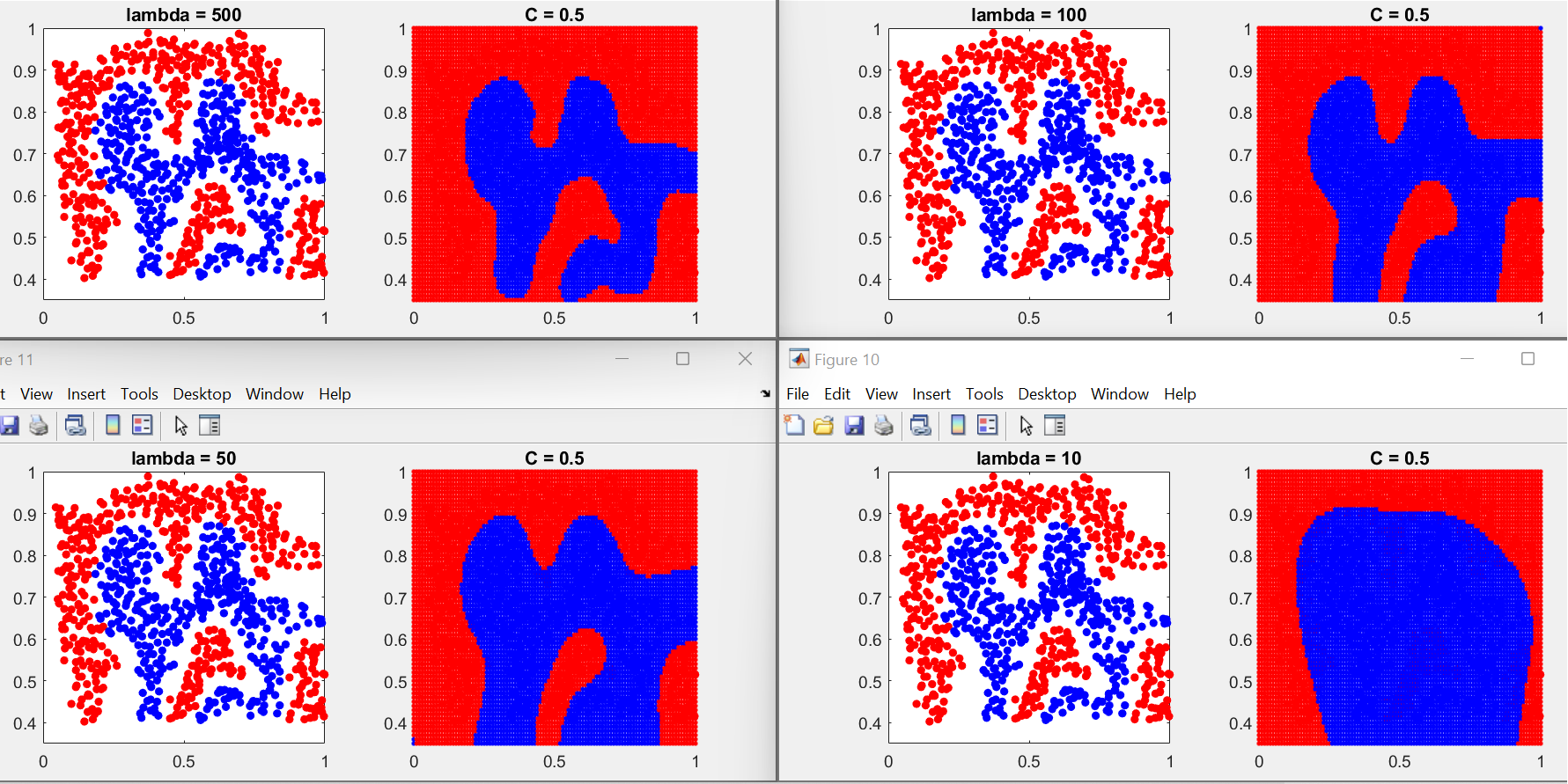
page4image7869312

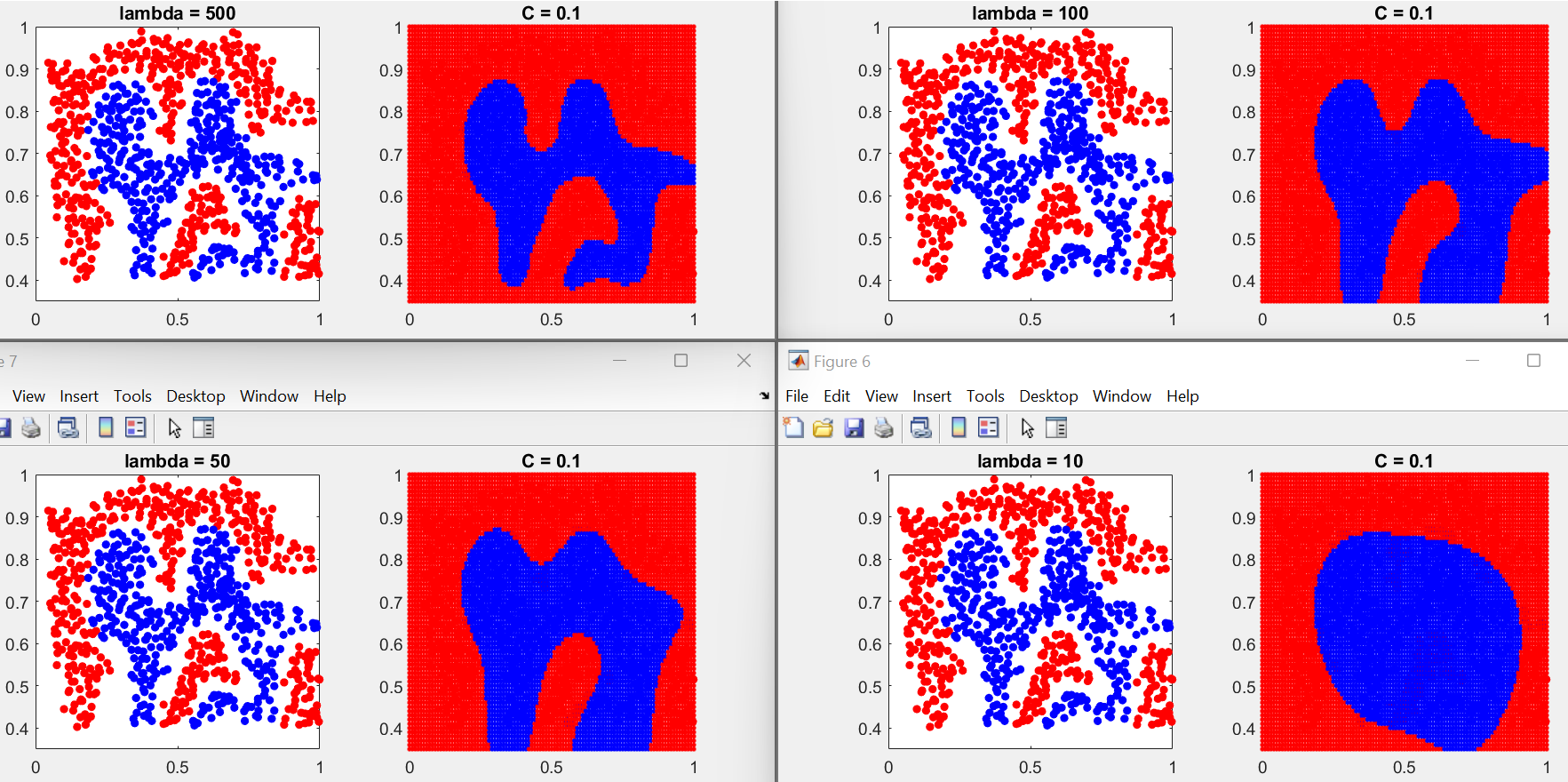


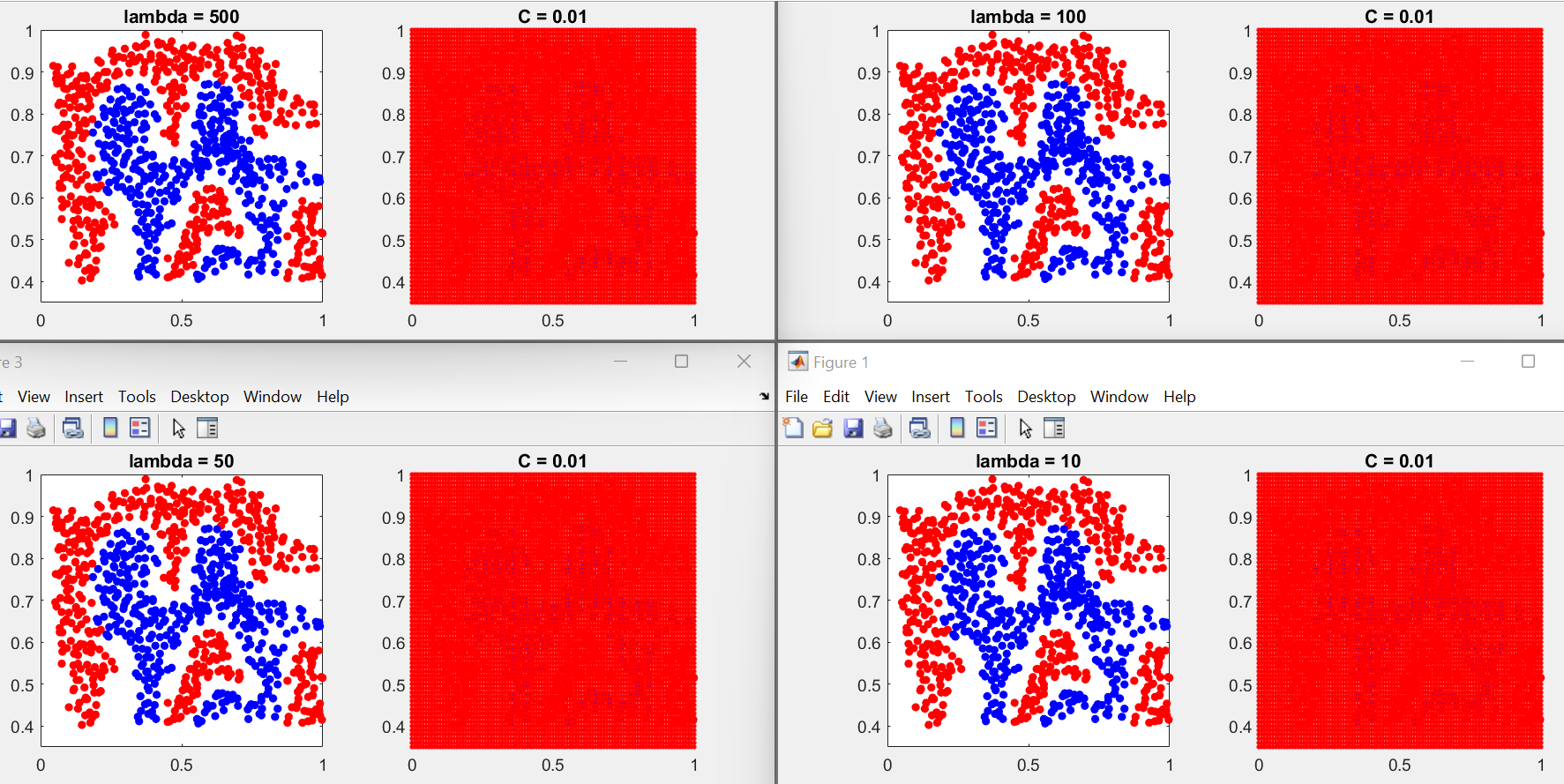
Pay attention, in each image, on the left side of the given data and on the right side, there are a large number of points on the page that are labeled, also on the right side of the given original data, we see a little larger than the rest of the plot points.

Landa and C corresponding to each shell are given above.





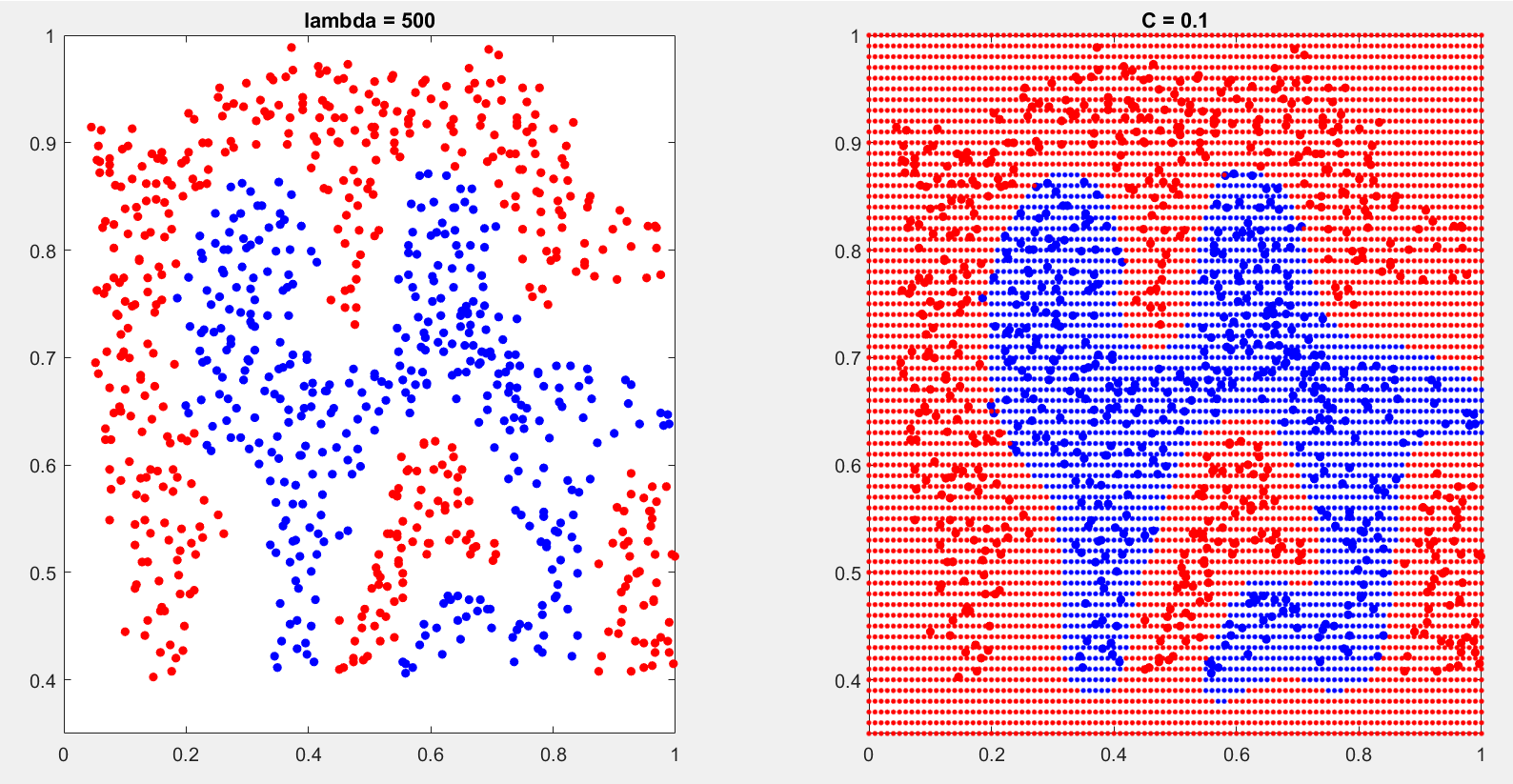




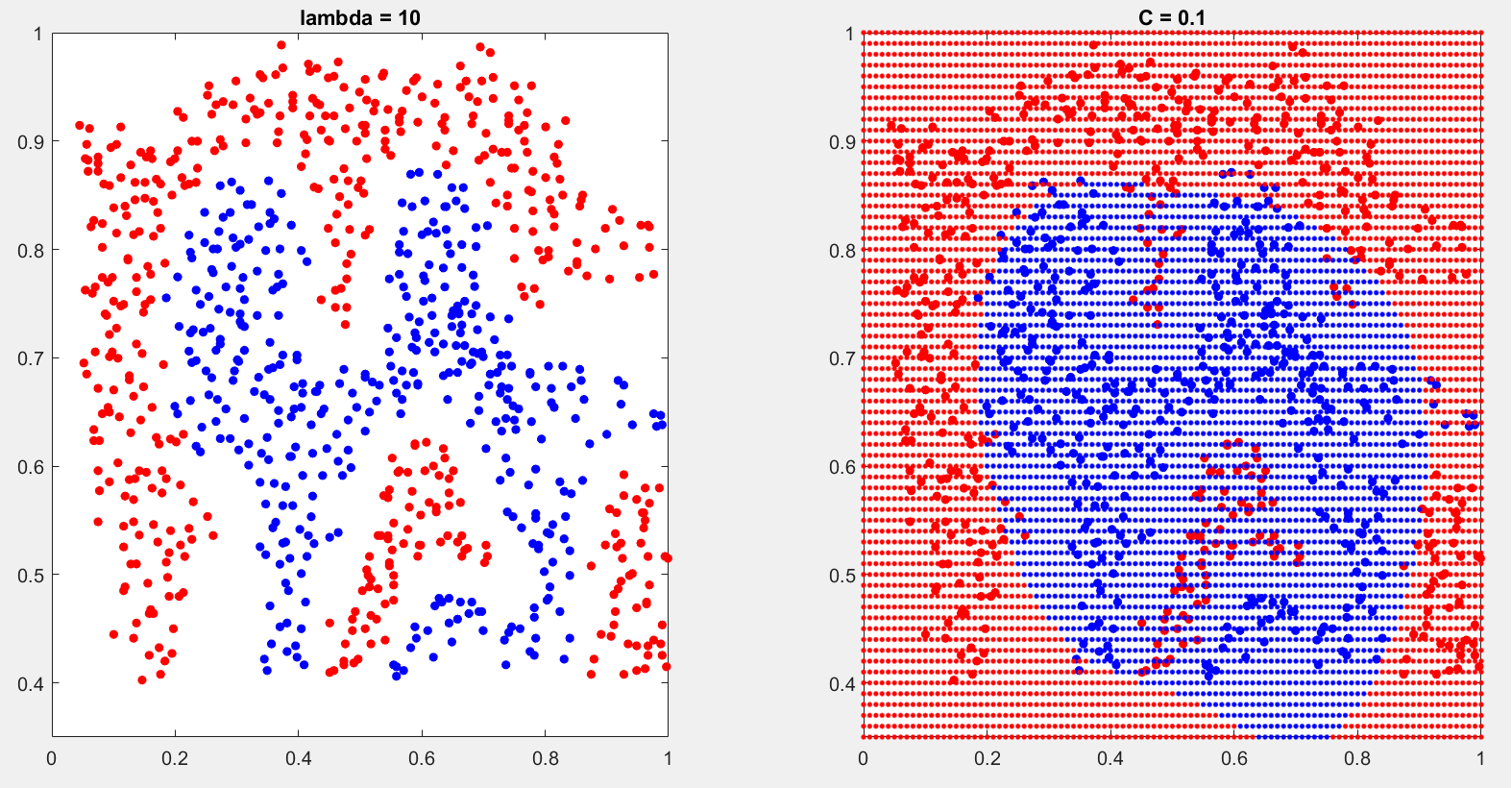
It is clear that the accuracy of data separation is directly related to C and lambda. In the sense that whatever these two parameters

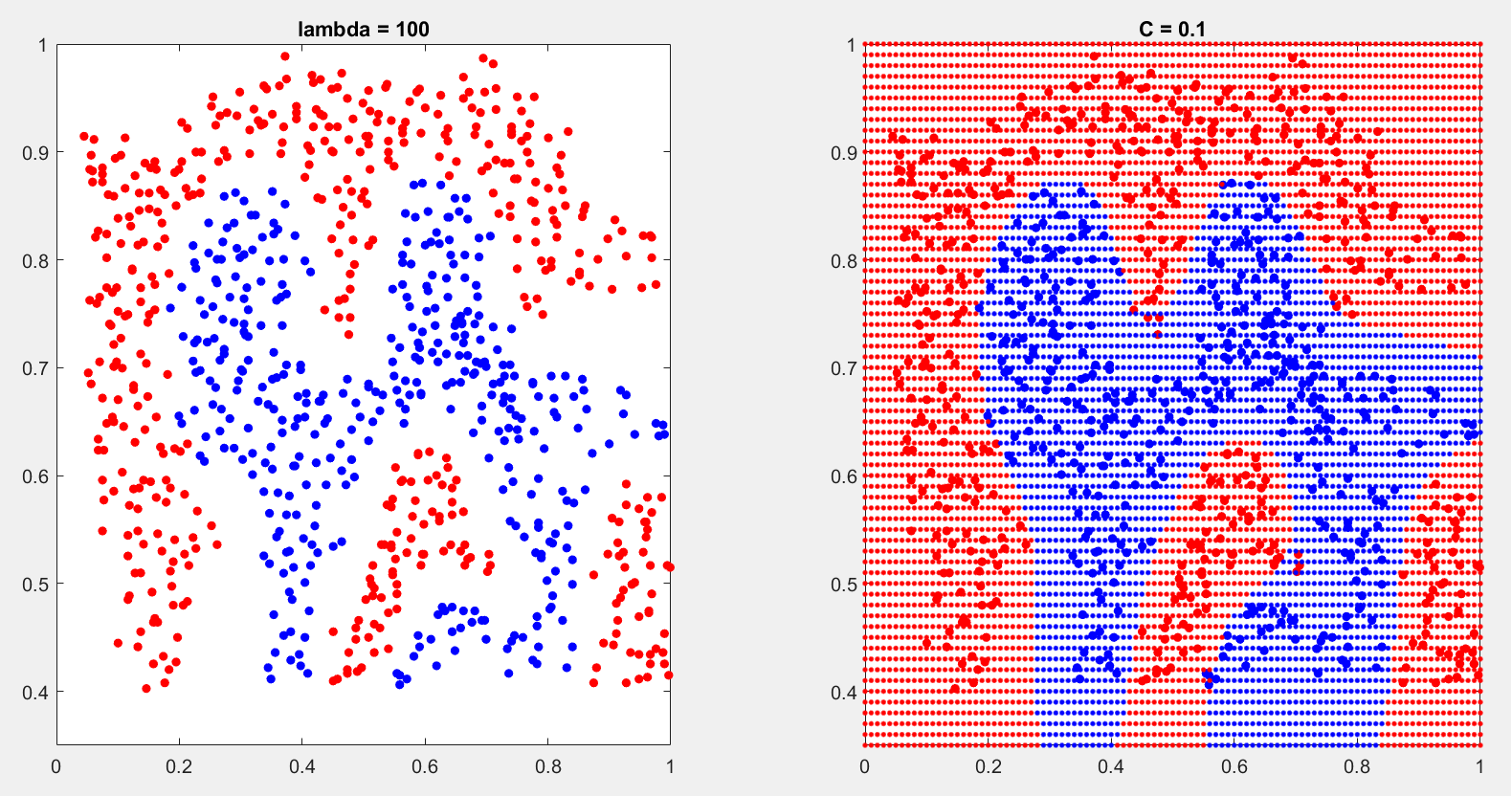
The bigger they are, the better our separation will be. It is also clear that the effect of C is greater in this case

As an example, let's put the bigger ones:



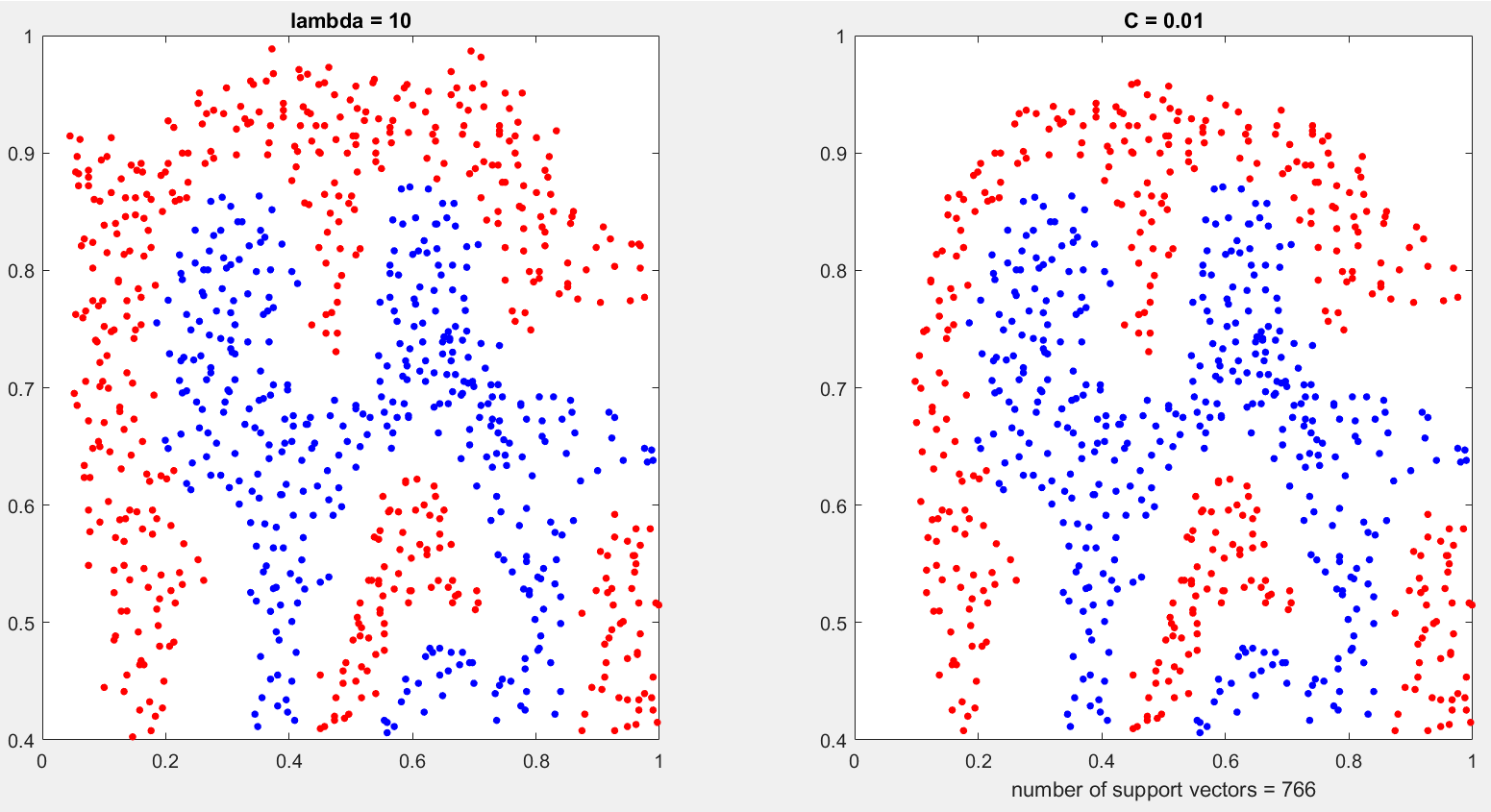


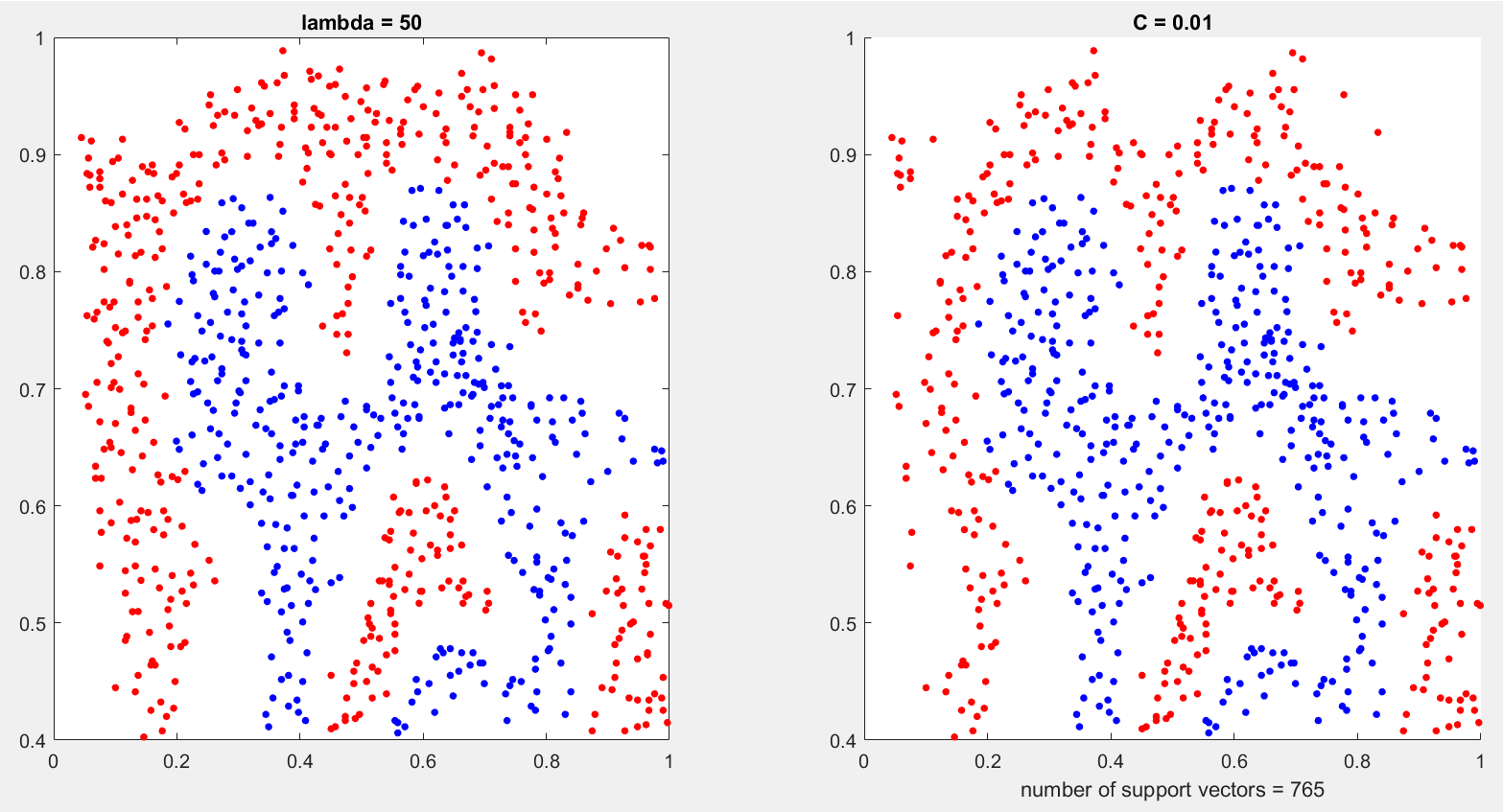


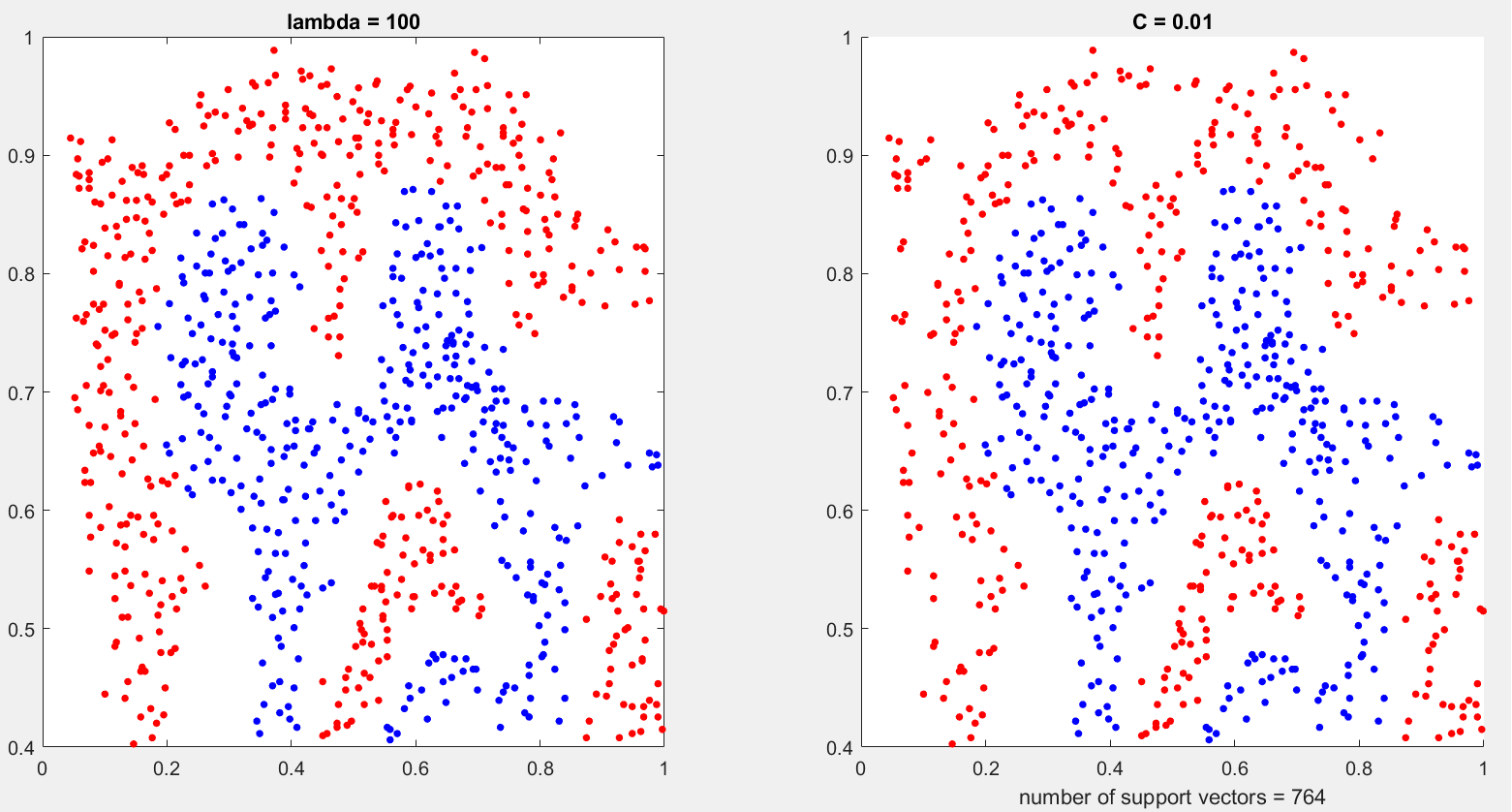


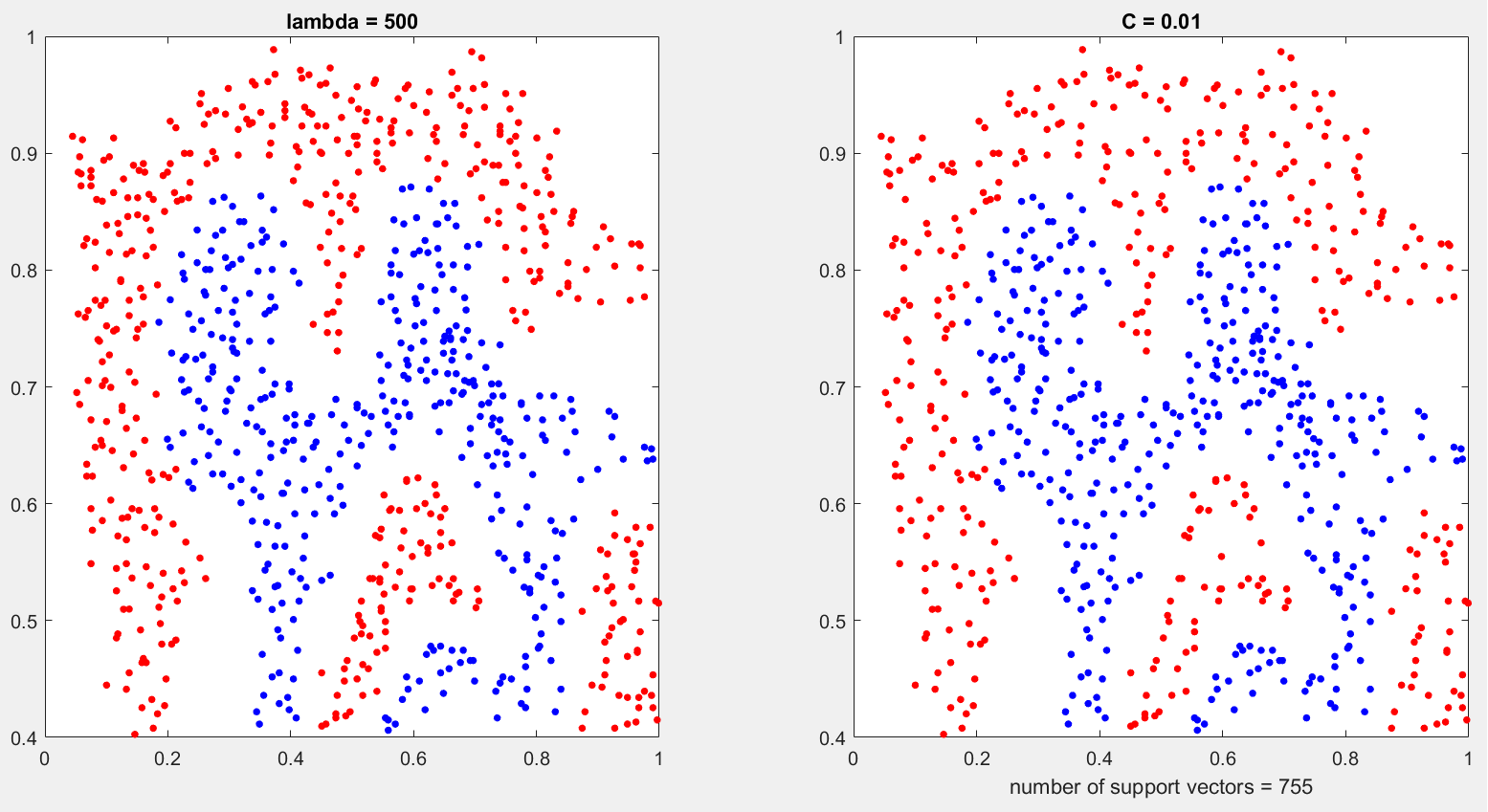
Part 3)

Using the code below, we check the necessary condition and draw it. Like the previous part, each image contains two parts; The left side of the original data and the right side of the support vector data that have applied in the condition. At the bottom of the right photo, the number of support vectors is indicated.

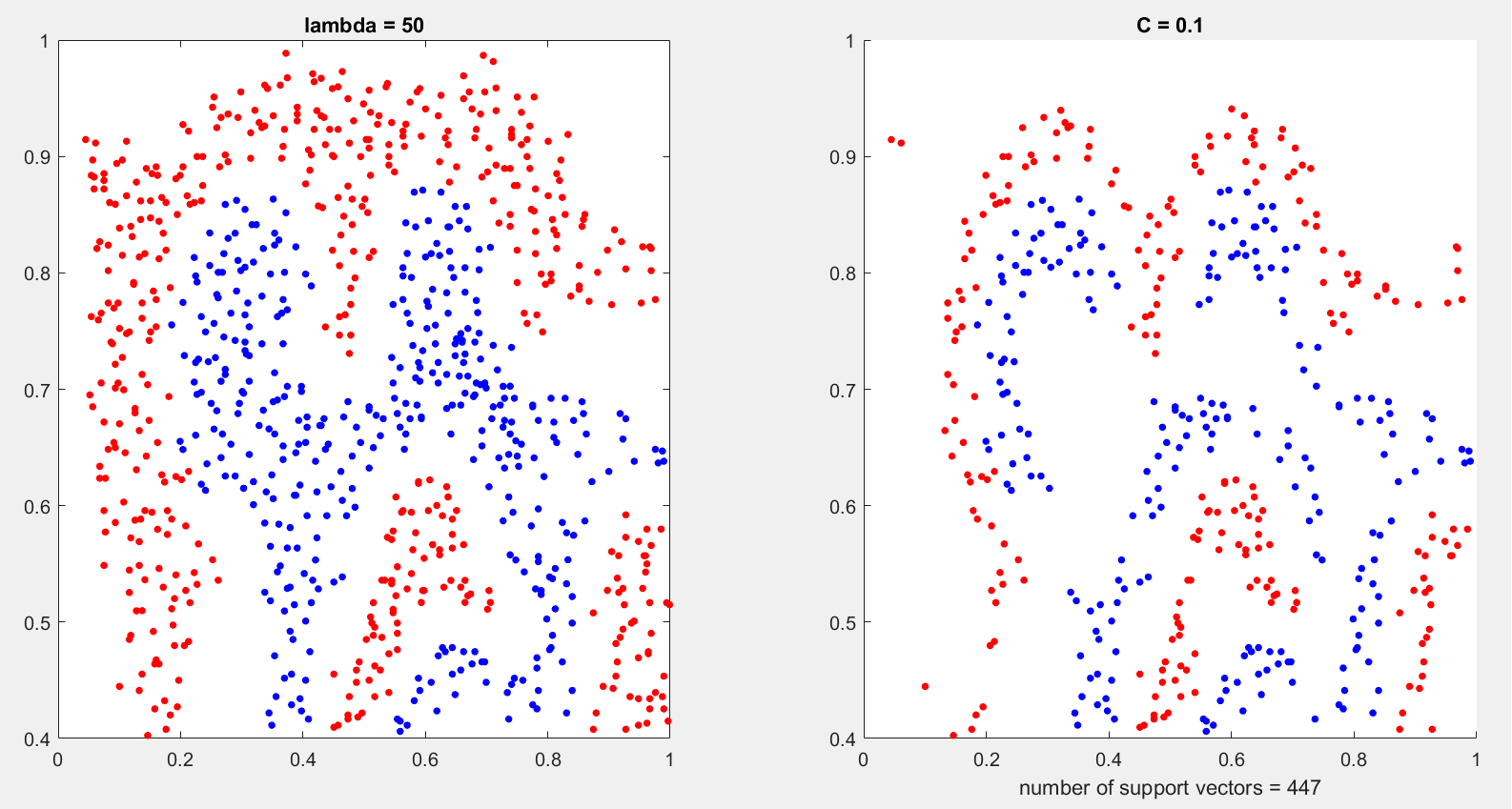


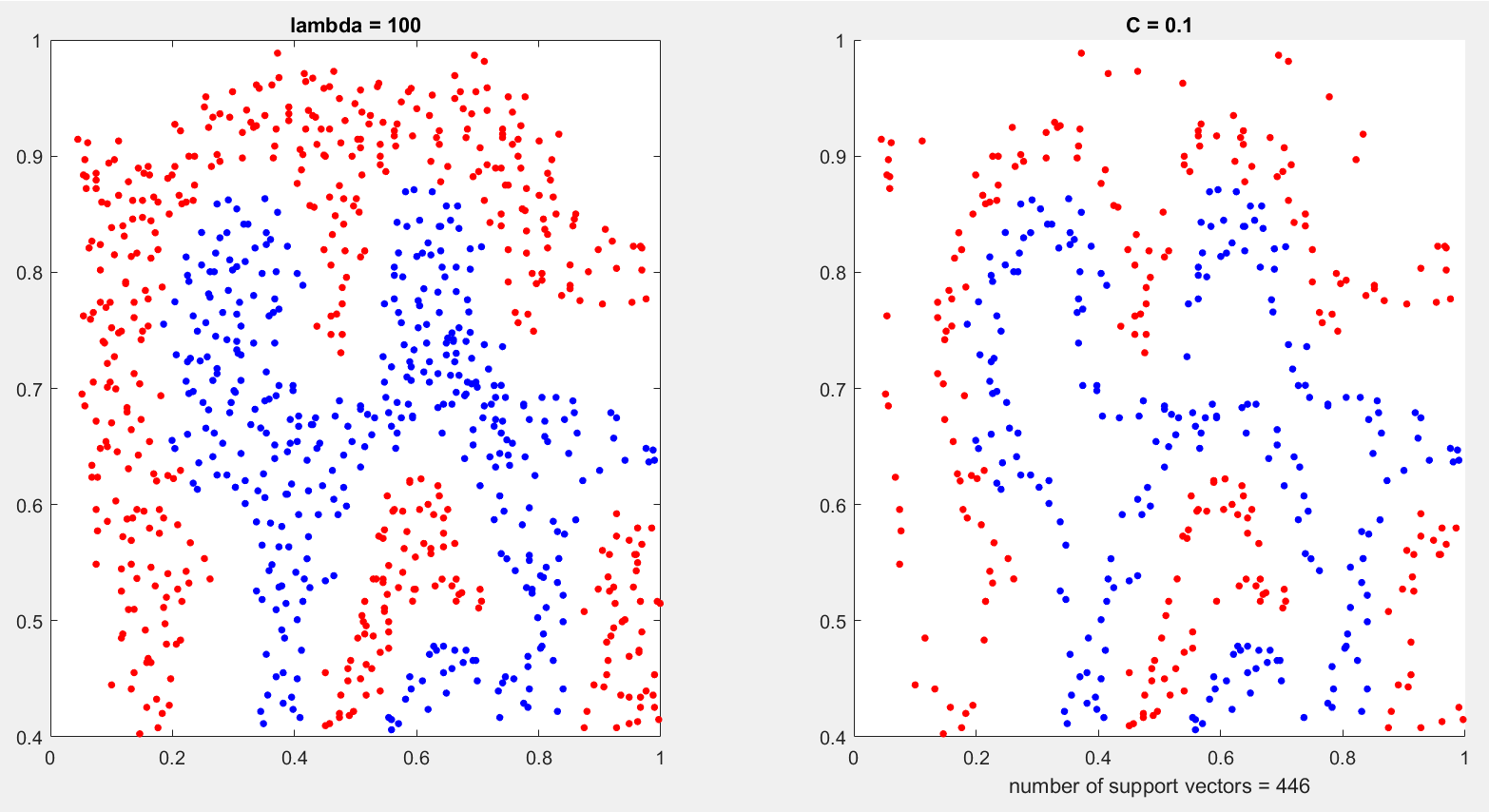


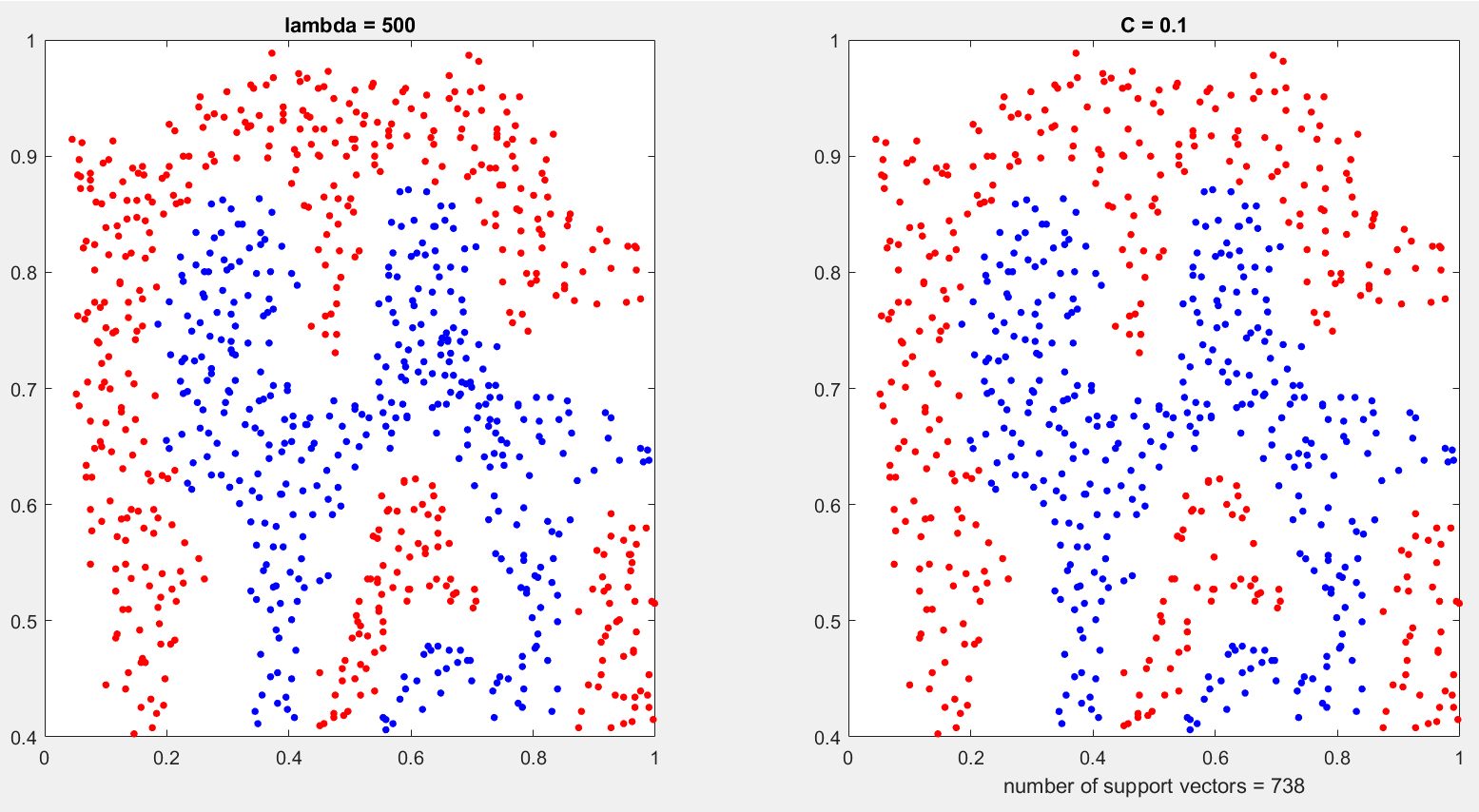


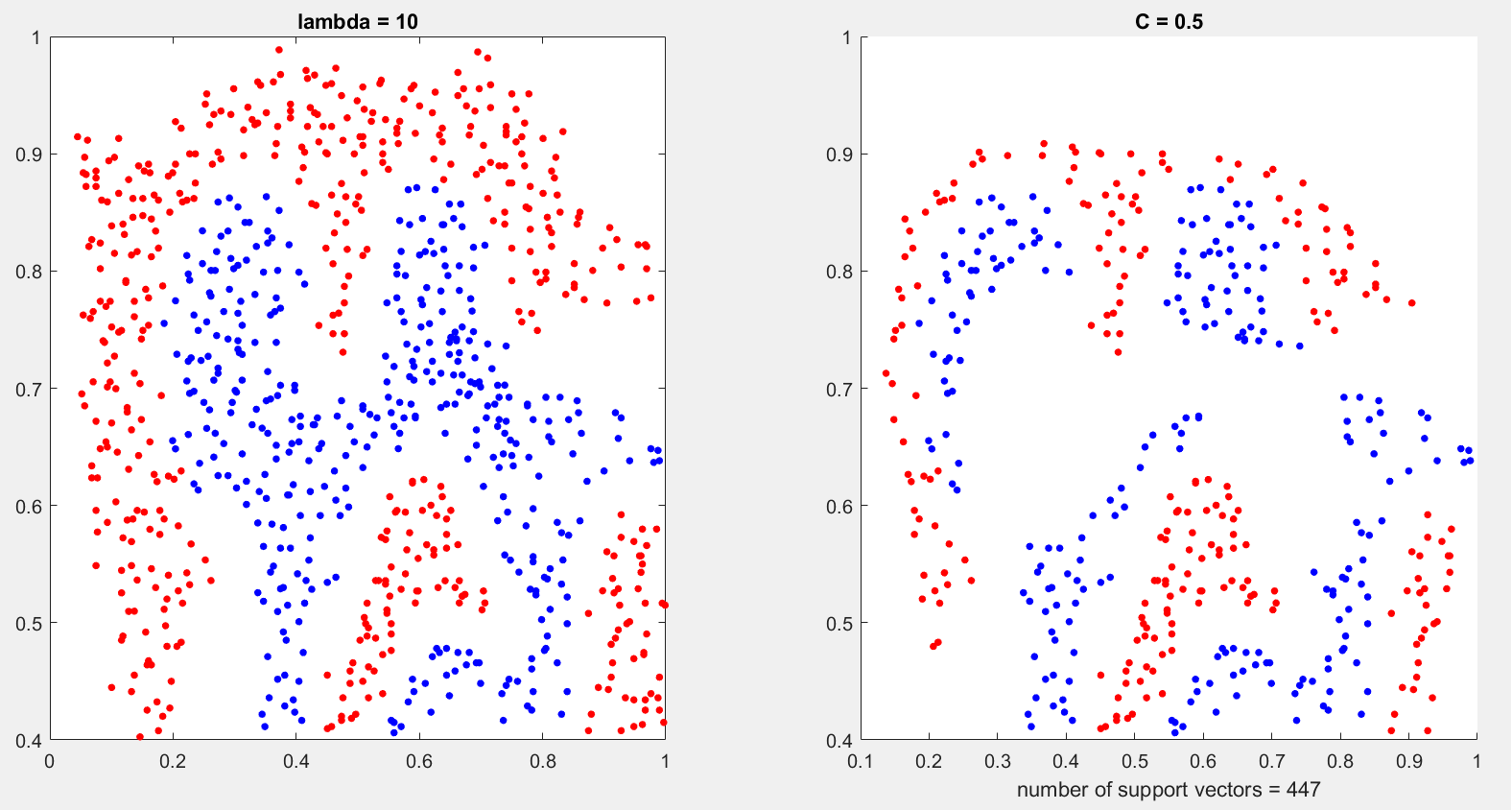




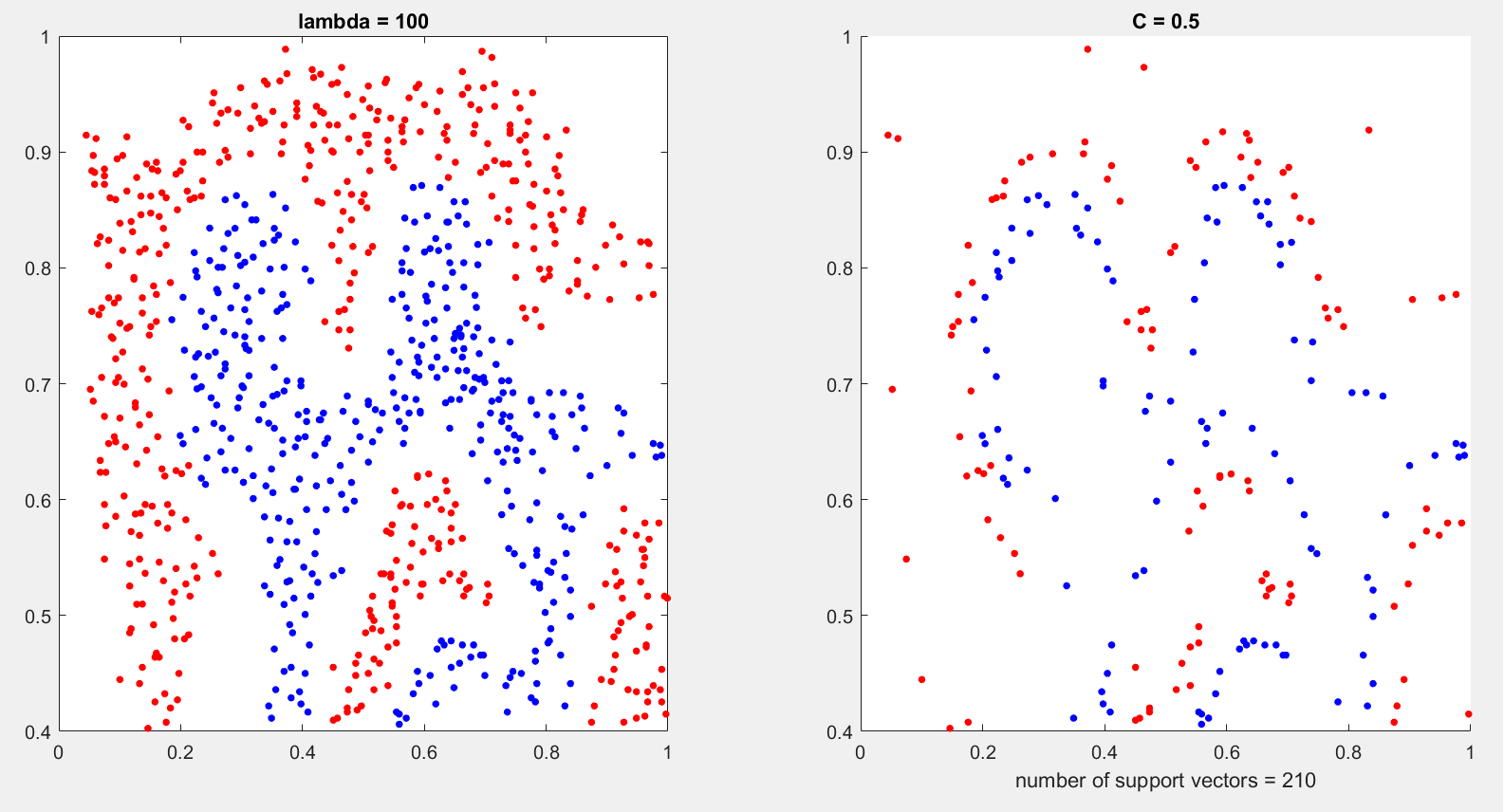


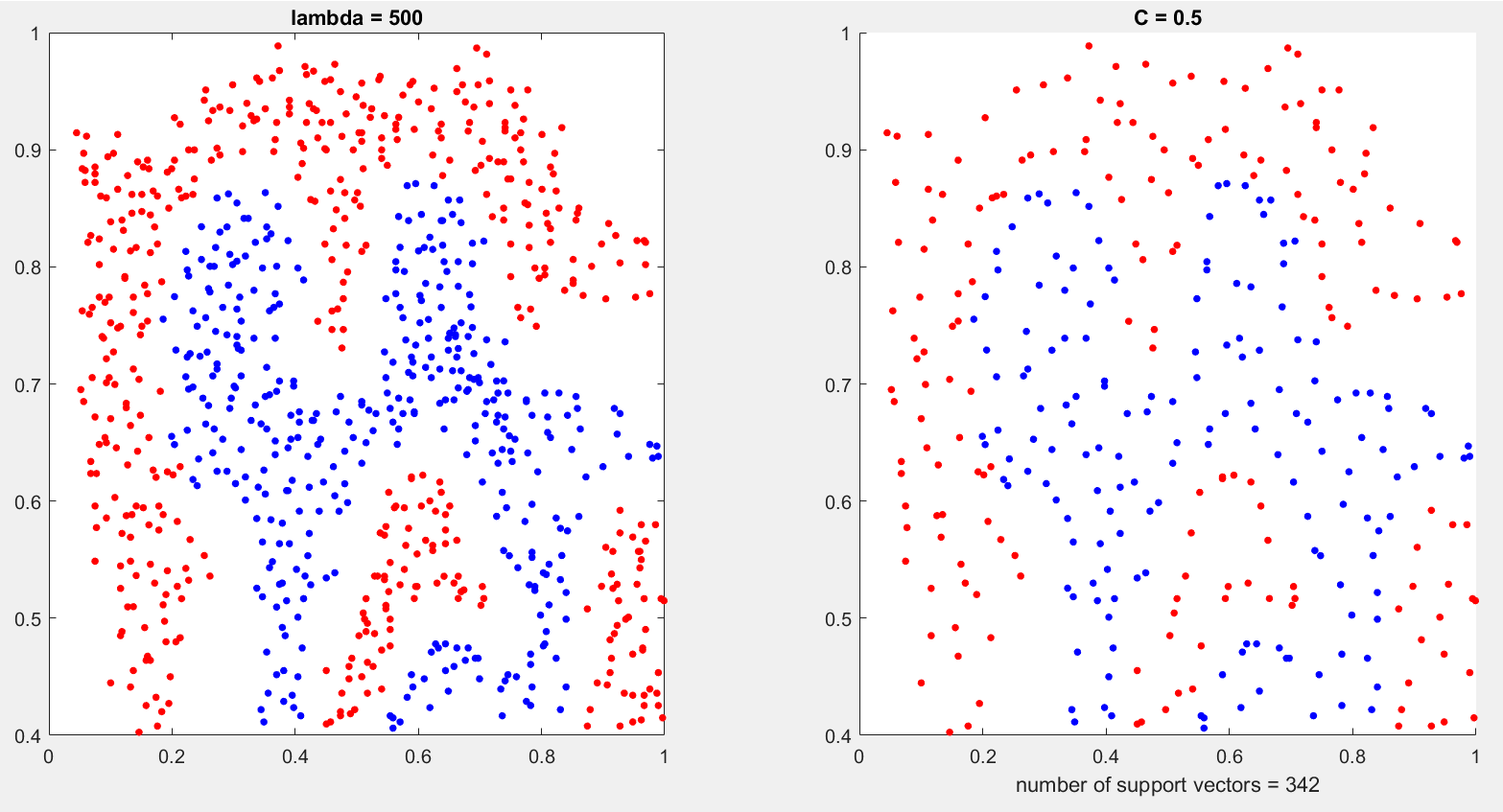


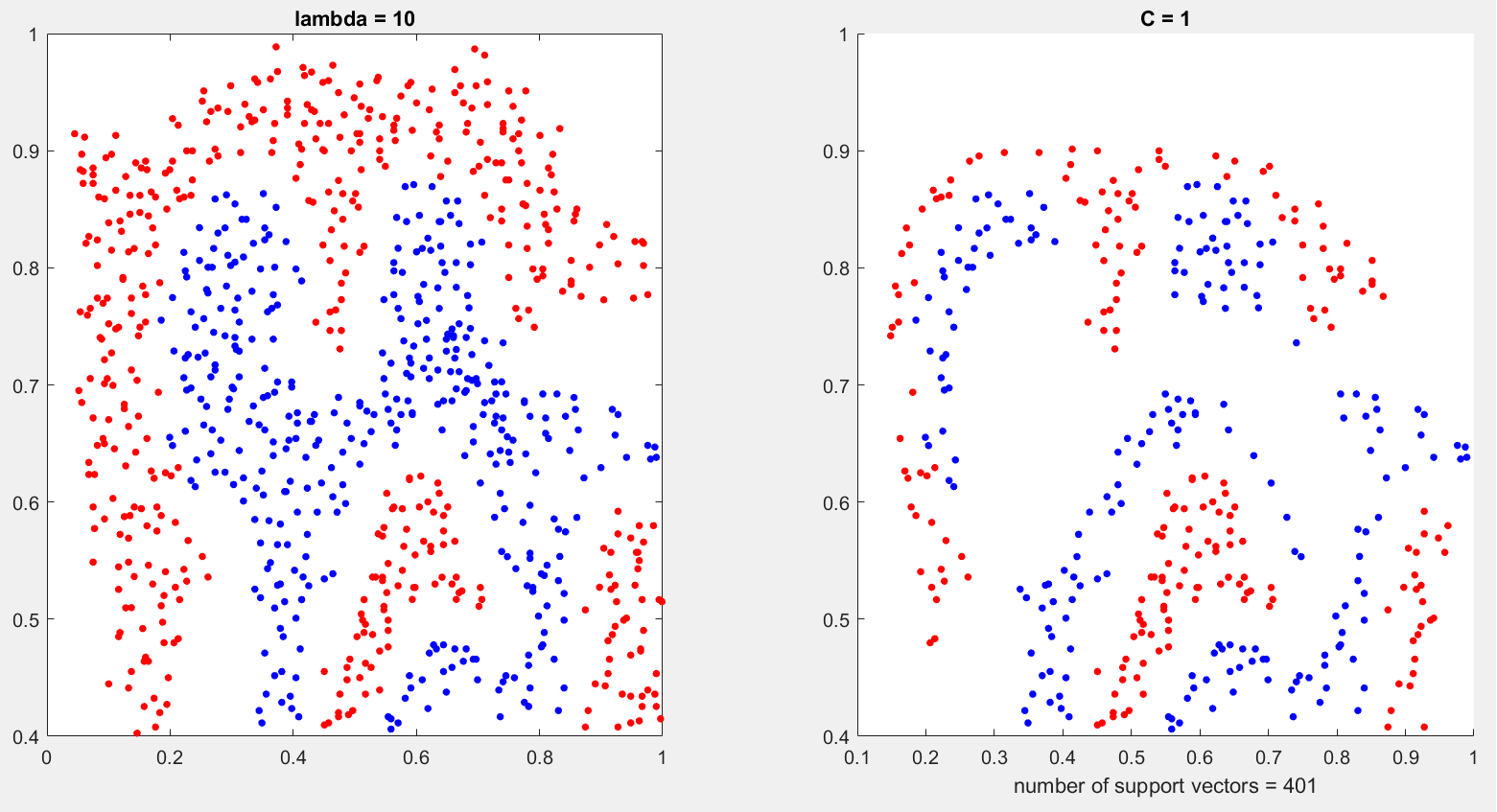


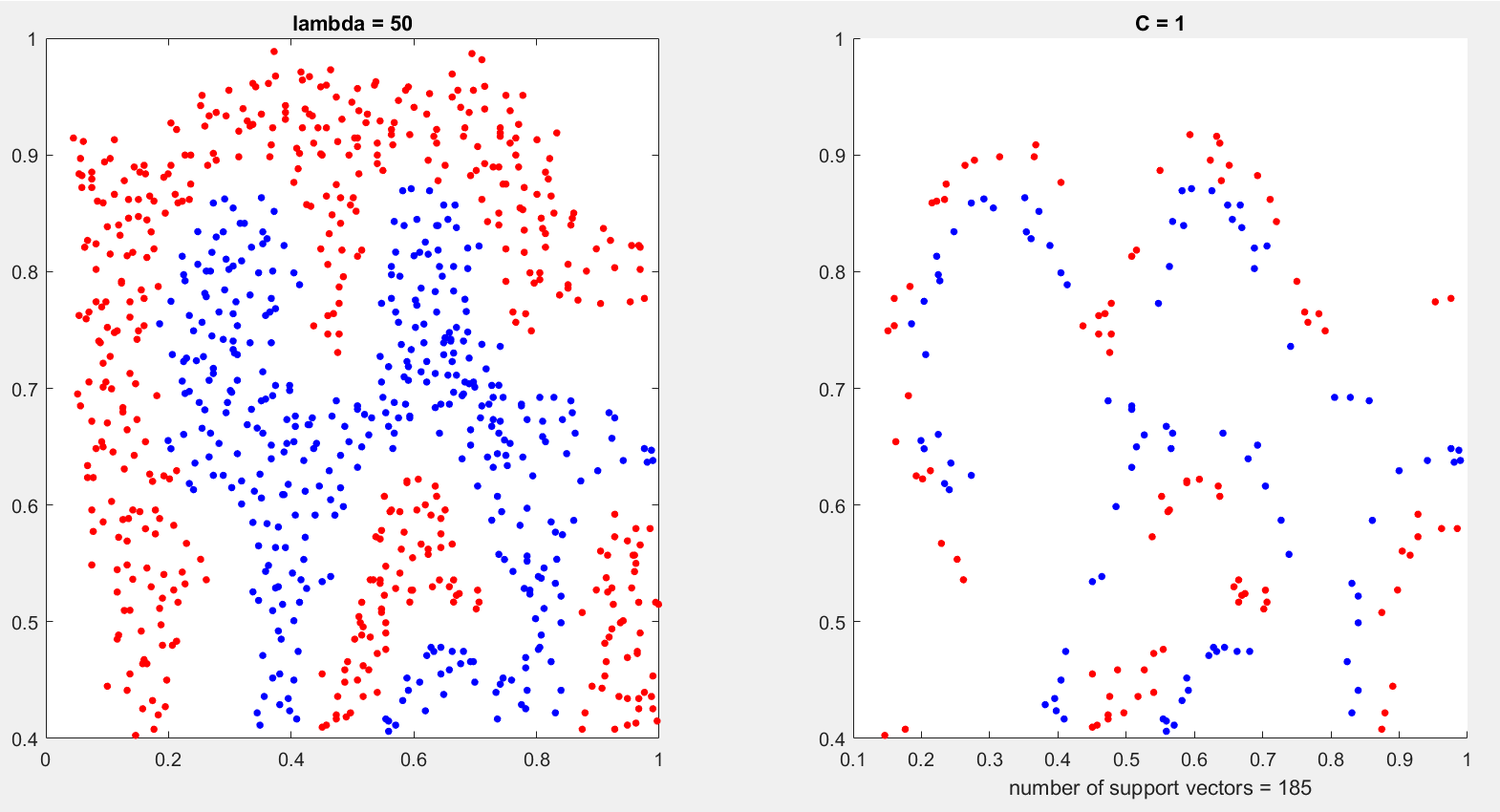


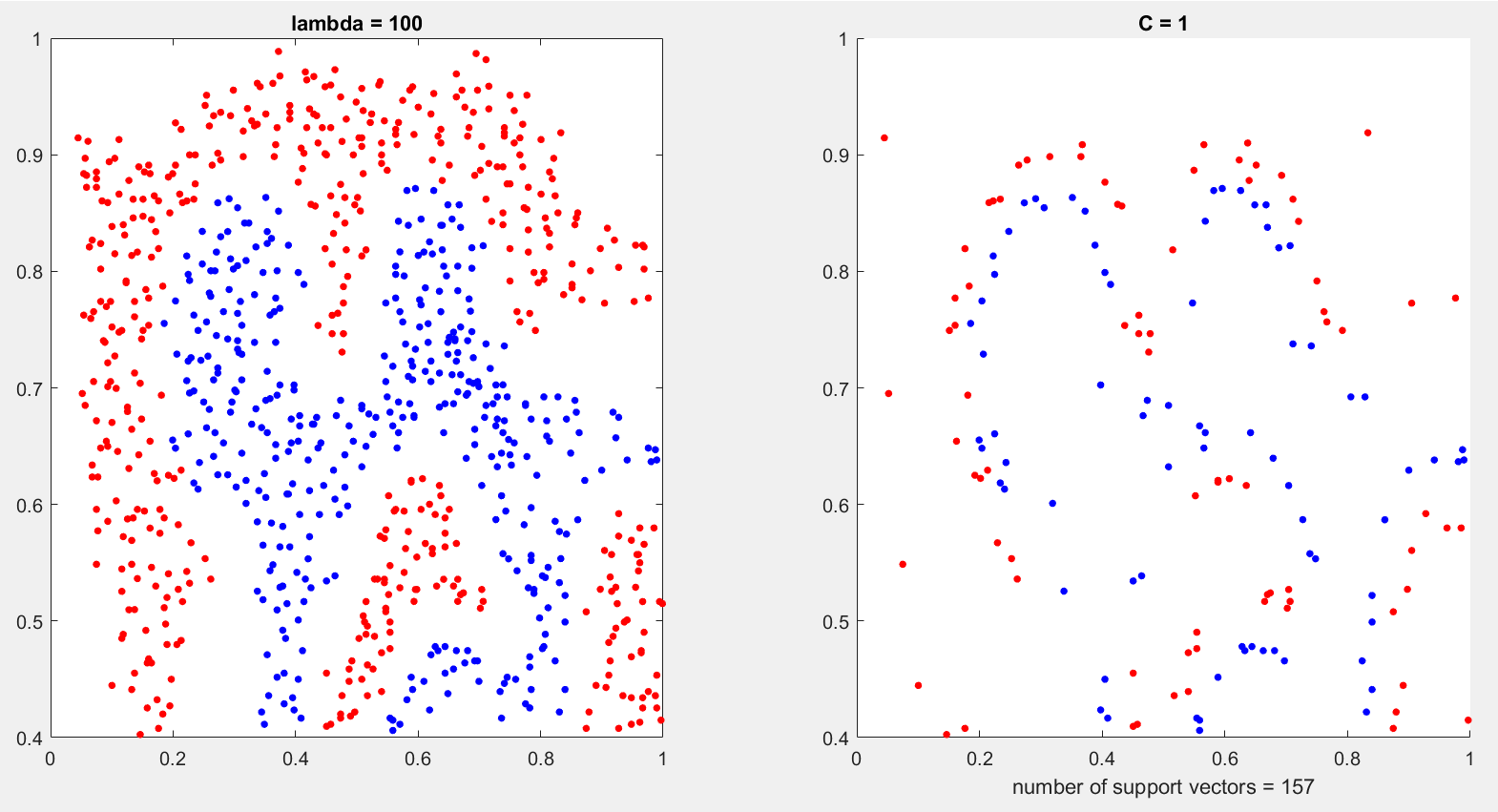


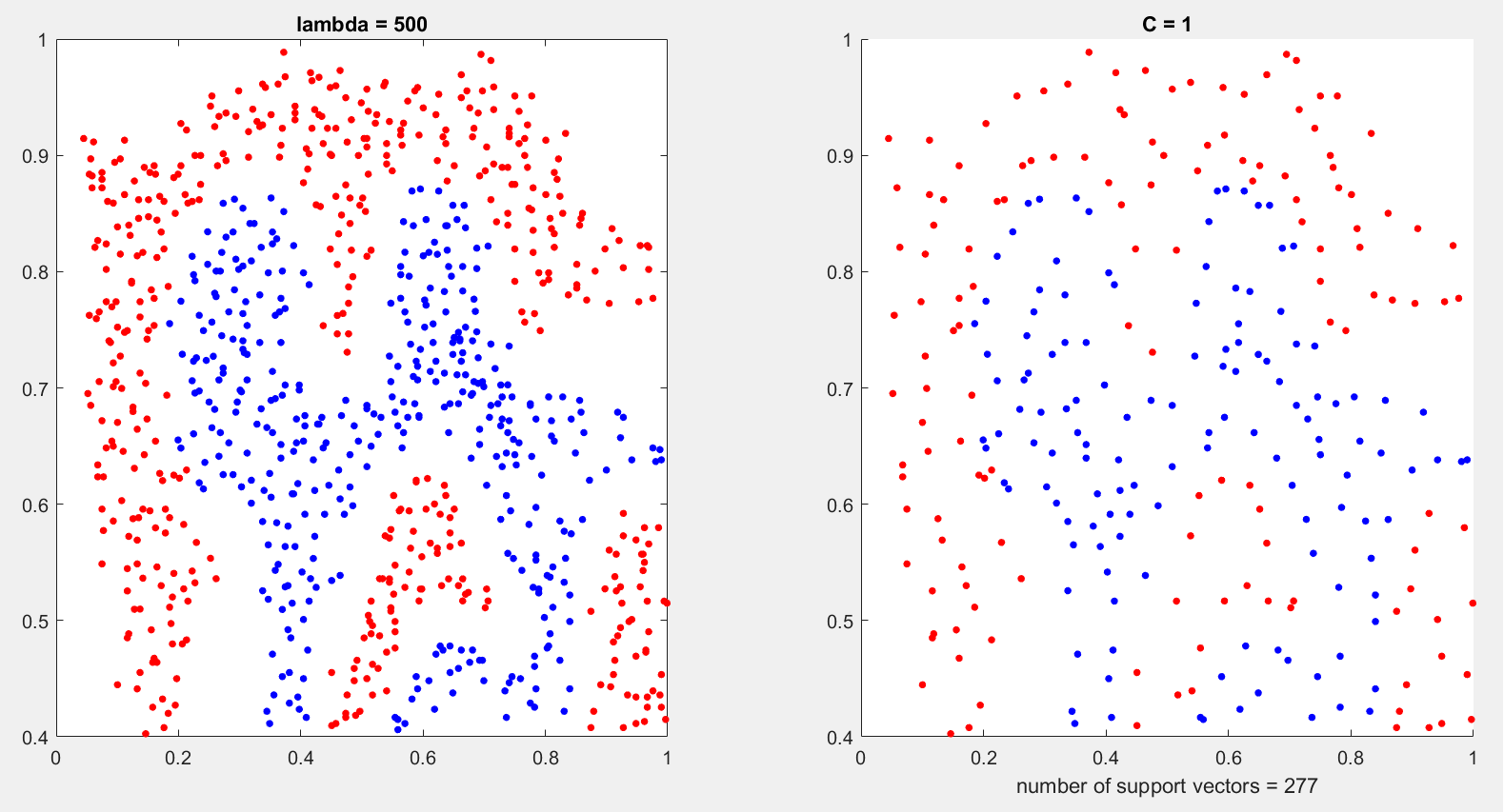












Part 4)

Part 5)

First, we specify the required constants. We can make η smaller or larger, and using testing to

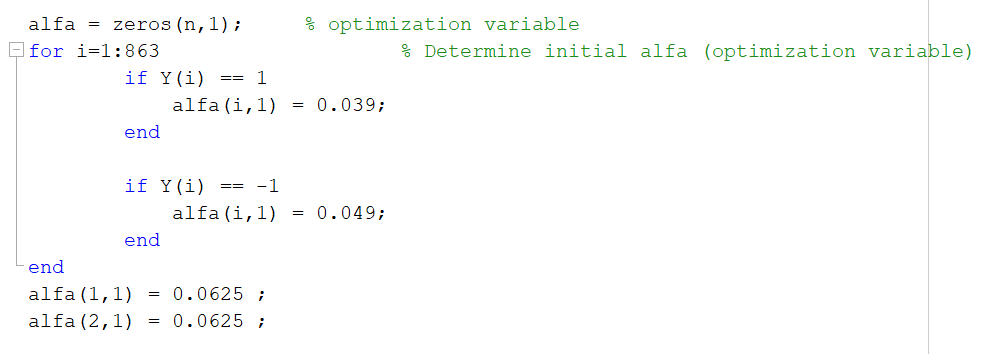
We get a value of 0.4. We follow the same procedure for tint.



In the next step, we specify the starting point of the feasible algorithm. First, we determine the number of positive and negative labels. Then we specify two numbers. One of these numbers

we use for the outputs and binary number for the negative output. We try to make αT Y close to zero; We can achieve this by error testing. Finally, we divide the remaining value by two and add it to the two alpha components that are labeled as . In all the previous steps, we make sure that the components are out of the desired opening.

don't be Finally, we reach the following result:

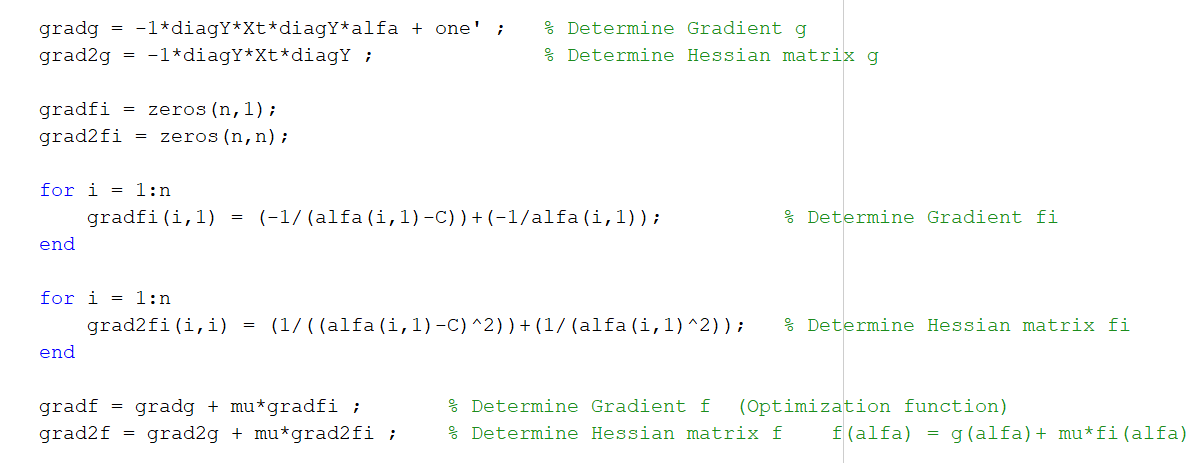


Algorithm start:

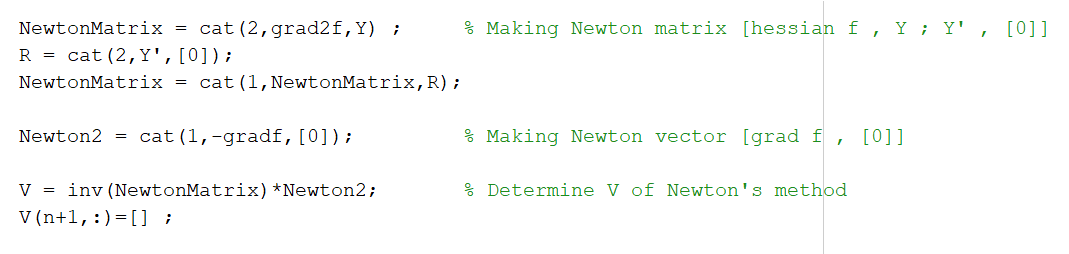
I consider Newton's stopping condition to be the decrease of the difference of two consecutive values. (abs(f 2 − f) > 0.0001) First, we obtain the gradient and Hessian function of the principle gradf, grad2f based on the relations calculated in the fourth part.

f(α) = g(α) + μΦ(α)

We also have

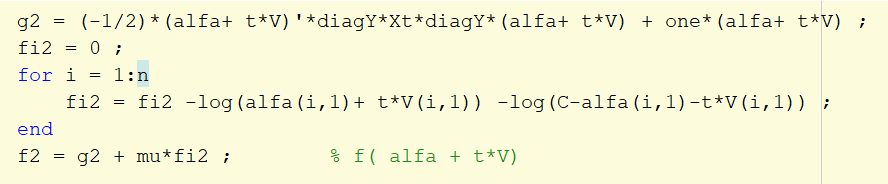


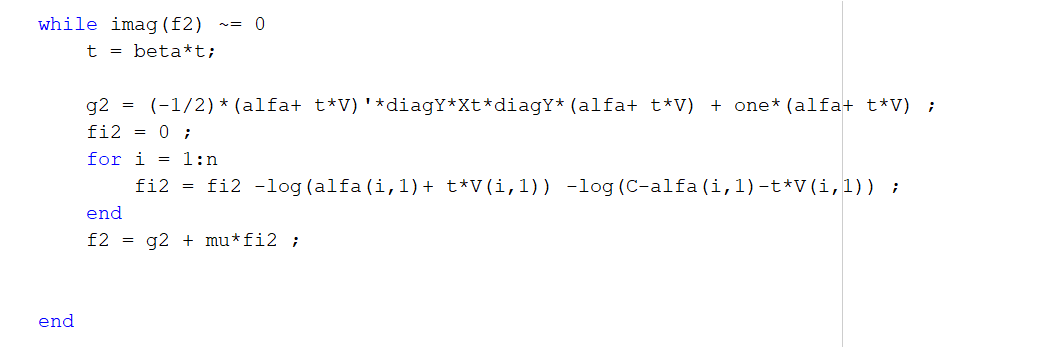
In the next step, we use Newton to get V, which is the direction of movement. The process of forming two required matrices is clear.

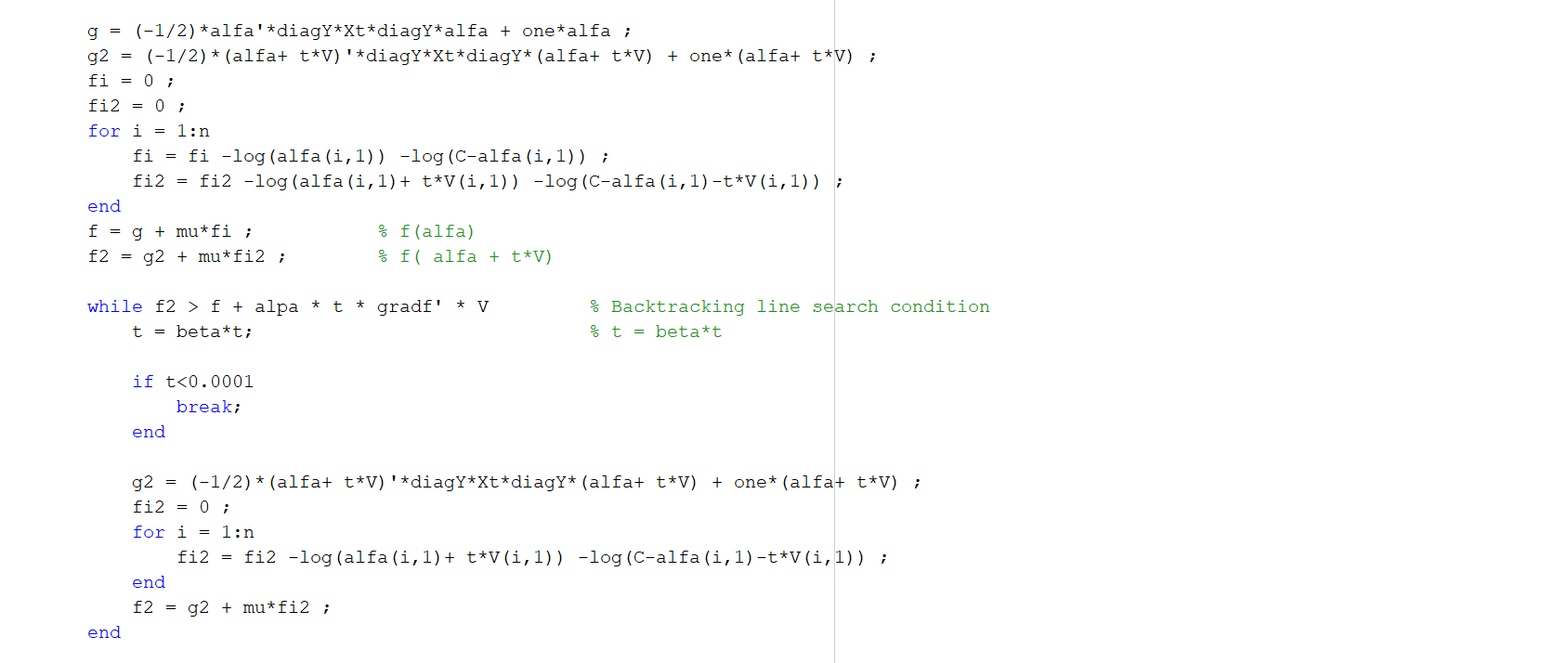


After getting V, now we need to get t, that is, the step size of the movement. According to the request of the project form, we use Tracing Line Search with the given constants. First, we set t to the initial value, then, in order not to break the algorithm, we check whether the value of α + tV, which is the next step, is negative or not with the initial step; If it is negative, then the input to the algorithm is negative and gives a mixed value, and everything is ruined. Therefore, with while

We prevent this and reduce the value of t as much as needed. t = βt Then we perform the original process of line search with the relevant condition.







With this expression inside the search line, we prevent t from moving too much so that it does not stop Newton's condition and repeats this loop a good amount. (abs(f 2 − f) > 0.0001)

At the end of Newton's way, α can be updated according to the determination of t and V.

page21image7786768

All of Newton's method is placed in the loop for the barrier method to satisfy the condition <8-10 mμ. After each Newton step

We will update μ.

page21image7786144

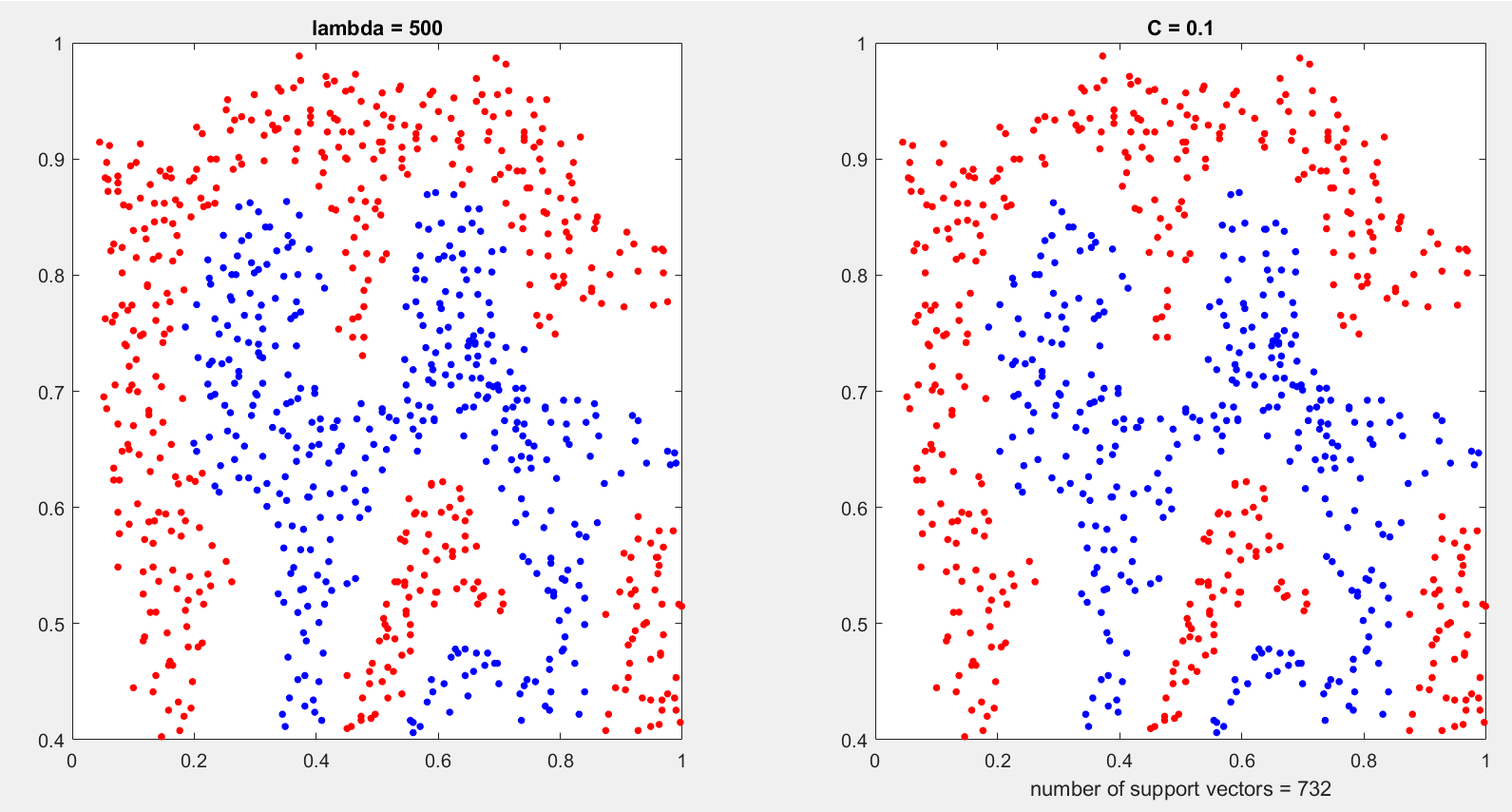
In the last step, we reach the obtained results.

Part 6)

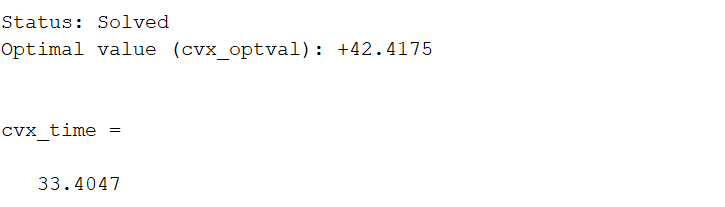
First, we compare the support vectors.

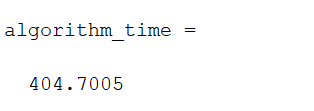
We can see that there are 732 support vectors here, and in the same case using our CVX function, 738 supports.

We had a vector that these two numbers are very close.



If we want to compare in terms of time, we can see that CVX takes about 11 times the time of our algorithm.





In terms of performance and separation, CVX has a better performance.

If we pay attention to α, we will notice that in solving the algorithm, it is close to the optimal alpha value obtained in CVX, and this is the confirmation of our way. But there is still an error.

