Problem 1: Support Vector Machines (19pts)

Consider a dataset consisting of points in the form of (x, y), where x is a real value, and $y \in \{-1, 1\}$ is the class label. There are only three points $(x_1, y_1) = (-1, -1)$, $(x_2, y_2) = (1, -1)$, and $(x_3, y_3) = (0, 1)$, shown in Figure 1.

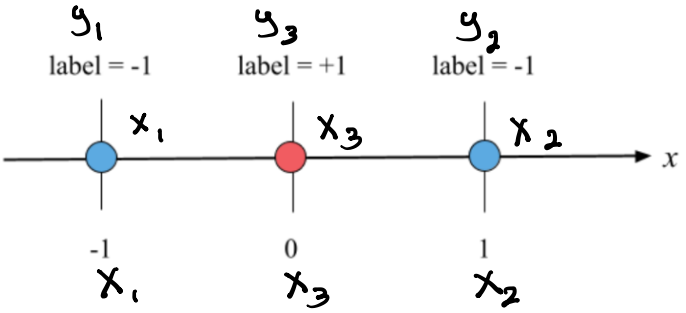


Figure 1: Three data points considered in Problem 1

1.1 (2pts) Can these three points in their current one-dimensional feature space be perfectly separated with a linear classifier? Why or why not?

No, we cannot linearly seperate here in one dimension. one point has to be miss-classified.

given $sig(w^T x + b)$ and $y = w^T x + b$

$$y = \begin{cases} +1 & \text{if } x \geq \theta \\ -1 & \text{else} \end{cases}, \quad \theta \text{ is threshold}$$

in this case if we want x_1, x_2 in the same class, x_3 must be miss-classified.

1.2 (3pts) Now we define a simple feature mapping $\phi(x) = [x, x^2]^T$ to transform the three points from one-dimensional to two-dimensional feature space. Plot the transformed points in the new two-dimensional feature space. Is there a linear model $\mathbf{w}^T\mathbf{x} + b$ for some $\mathbf{w} \in \mathbb{R}^2$ and $b \in \mathbb{R}$ that can correctly separate the three points in this new feature space? Why or why not?

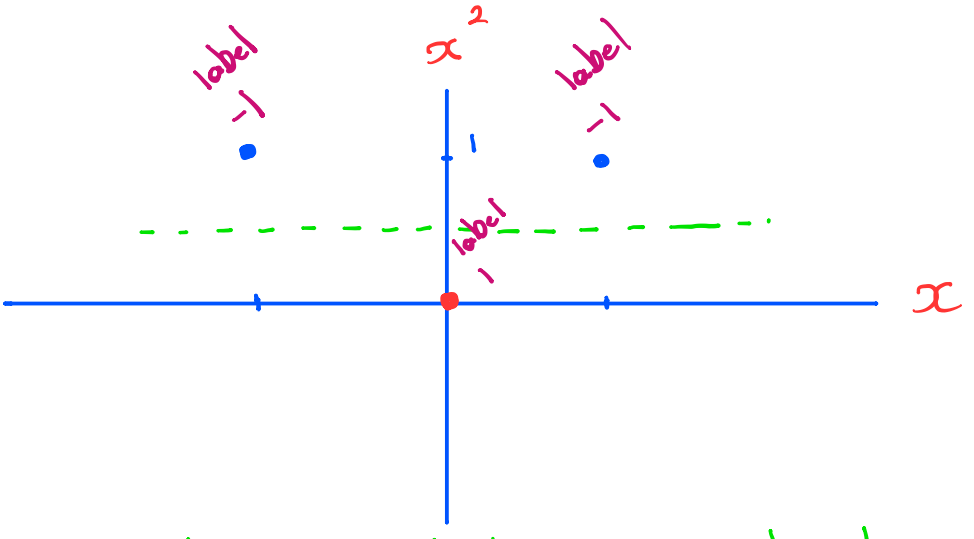
$\text{sign}(w^T \phi(x) + b)$

$(x_1, y_1) = (-1, -1)$, $(x_2, y_2) = (1, -1)$, and $(x_3, y_3) = (0, 1)$,

$$\phi_1(x) = [-1, (-1)^2]^T = \overset{x \quad x^2}{[-1, 1]^T} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

$$\phi_2(x) = [1, 1^2]^T = \overset{x \quad x^2}{[1, 1]^T} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\phi_3(x) = [0, (0)^2]^T = \overset{x \quad x^2}{[0, 0]^T} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$



Any dotted line between 0 to 1 on x^2 axis will correctly classify.

An example:

if we pick $w = (0, 1)$ & $b = 0$

① $(0,1) \cdot [-1, 1]^T + 0 = 1 \rightarrow \text{sign}(1)$ ✓

② $(0,1) \cdot [1, 1]^T + 0 = 1 \rightarrow \text{sign}(1)$ ✓

③ $(0,1) \cdot [0, 0]^T + 0 = 0 \rightarrow \text{sign}(0)$ ✓

1.3 (2pts) Given the feature mapping $\phi(x) = [x, x^2]^T$, write down the 3×3 kernel/Gram matrix $\mathbf{K}$ for this dataset.

$$k_{i,j} = \phi(x_i)^T \phi(x_j)$$

$$\phi(x_1) = [-1, 1]^T \qquad \phi(x_3) = [0, 0]$$

$$\phi(x_2) = [1, 1]^T$$

$$k = \begin{bmatrix} \phi(x_1)^T \phi(x_1) & \phi(x_1)^T \phi(x_2) & \phi(x_1)^T \phi(x_3) \\ \phi(x_2)^T \phi(x_1) & \phi(x_2)^T \phi(x_2) & \phi(x_2)^T \phi(x_3) \\ \phi(x_3)^T \phi(x_1) & \phi(x_3)^T \phi(x_2) & \phi(x_3)^T \phi(x_3) \end{bmatrix} =$$

$$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\phi(x_1)^T \phi(x_1) =$$

$$[-1, 1] \begin{bmatrix} -1 \\ 1 \end{bmatrix} =$$

$$(-1)(-1) + (1)(1) = 2$$

$$\phi(x_2)^T \phi(x_2) =$$

$$[1, 1] \begin{bmatrix} 1 \\ 1 \end{bmatrix} =$$

$$(1)(1) + (1)(1) = 2$$

$$\vdots$$

1.4 (4pts) Now write down the primal and dual formulations of SVM for this dataset in the two-dimensional feature space. Note that when the data is separable, we set the hyperparameter C to be $+\infty$ which makes sure that all slack variables (ξ) in the primal formulation have to be 0 (and thus can be removed from the optimization).

Primal (General):

$$\min_{w,b,\{\epsilon_i\}} \; C\sum_i \epsilon_i + \frac{1}{2}\|w\|_2^2$$

$$\text{s.t } y_i(w^T \phi(x_i) + b) \geq 1 - \epsilon_i \quad \forall i \in [n]$$

$$\epsilon_i \geq 0 \quad \forall i \; [n]$$

Dual (General)

$$\max_{\{\alpha_i\}} \; \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} y_i y_j \alpha_i \alpha_j \phi(x_i)^T \phi(x_j)$$

$$\text{s.t } \sum_i \alpha_i y_j = 0 \;\; \& \;\; 0 \leq \alpha_i \leq C, \; \forall i \in [n]$$

Primal For separable data

$$\min_{w,b} \frac{1}{2}\|w\|_2^2, \; \text{s.t } y_i(w^T \phi(x_i) + b) \geq 1 \quad \forall i \in [n]$$

$w = (w_1, w_2)$

we have $\min_{w_1, w_2, b} \frac{1}{2}(w_1^2 + w_2^2)$

s.t for each data point:

For $(x_1, y_1) = (-1, -1)$

$$-1(w_1(-1) + w_2(1) + b) \geq 1$$

$$w_1 - w_2 - b \geq 1$$

For $(x_2, y_2) = (1, -1)$

$$-1(w_1 \cdot 1 + w_2 \cdot 1 + b) \geq 1$$

$$-w_1 - w_2 - b \geq 1 \rightarrow w_1 + w_2 + b \leq -1$$

For $(x_3, y_3) = (0, 1)$

$$1(w_1 \cdot 0 + w_2 \cdot 0 + b) \geq 1$$

$$b \geq 1$$

combining constraint: we want to choose b such that satisfies all 3 constraints and we need to have $\boxed{b \geq 1}$

Dual For separable data

$$\max_{\{\alpha_i\}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} y_i y_j \alpha_i \alpha_j k(x_i, x_j)$$

$$s.t \sum_{i=1}^{n} \alpha_i y_i = 0 \quad \text{and} \quad \alpha_i \geq 0, \ \forall i \in [n]$$

we have 3 Lagrange multiplies $\alpha_1, \alpha_2, \alpha_3$

For $(x_1, y_1) = (-1, -1)$, $(x_2, y_2) = (1, -1)$, and $(x_3, y_3) = (0, 1)$,

$$\max_{\alpha_1, \alpha_2, \alpha_3} \alpha_1 + \alpha_2 + \alpha_3 - \frac{1}{2}\Big[y_1 y_1 \alpha_1 \alpha_1 k(x_1, x_1) + y_1 y_2 \alpha_1 \alpha_2 k(x_1, x_2) +$$

$$y_1 y_3 \alpha_1 \alpha_3 k(x_1, x_3) + y_2 y_1 \alpha_2 \alpha_1 k(x_2, x_1) + y_2 y_2 \alpha_2 \alpha_2 k(x_2, x_2) +$$

$$y_2 y_3 \alpha_2 \alpha_3 k(x_2, x_3) + y_3 y_1 \alpha_3 \alpha_1 k(x_3, x_1) + y_3 y_2 \alpha_3 \alpha_1 k(x_3, x_2) +$$

$$y_3 y_3 \alpha_3 \alpha_3 k(x_3, x_3)\Big] = \alpha_1 + \alpha_2 + \alpha_3 - \frac{1}{2}(\alpha_1^2 + \alpha_2^2)$$

$$= \alpha_1 + \alpha_2 + \alpha_3 - \alpha_1^2 - \alpha_2^2$$

simplification steps done in a scratch paper.

s.t $\alpha_1(-1) + \alpha_2(-1) + \alpha_3(1) = 0$

$-\alpha_1 - \alpha_2 + \alpha_3 = 0 \Rightarrow \alpha_1 + \alpha_2 = \alpha_3$

side. $k(x_i, x_j) = \phi(x)^T \phi(x)$

For $(x_1, y_1) = (-1, -1)$

$k(x_1, x_1) = 2$

$k(x_1, x_2) = 0$

$k(x_1, x_3) = 0$

For $(x_2, y_2) = (1, -1)$

$k(x_2, x_1) = 0$

$k(x_2, x_2) = 2$

$k(x_2, x_3) = 0$

For $(x_3, y_3) = (0, 1)$

$k(x_3, x_1) = 0$

$k(x_3, x_2) = 0$

$k(x_3, x_3) = 0$

1.5 (5pts) Next, solve the dual formulation exactly (note: while this is not generally feasible, the simple form of this dataset makes it possible). Based on that, calculate the primal solution.

we have $\alpha_1 + \alpha_2 + \alpha_3 - \alpha_1^2 - \alpha_2^2$ and eliminating α_3 by $\alpha_3 = \alpha_1 + \alpha_2$

$$\max_{\alpha_1, \alpha_2 \geq 0} \alpha_1 + \alpha_2 + \alpha_1 + \alpha_2 - \alpha_1^2 - \alpha_2^2$$

$$2\alpha_1 + 2\alpha_2 - \alpha_1^2 - \alpha_2^2$$

To maximize over α_1 & α_2

$$\frac{d}{d\alpha_1}(2\alpha_1 + 2\alpha_2 - \alpha_1^2 - \alpha_2^2) = 2 - 2\alpha_1 = 0$$

$$\frac{d}{d\alpha_2}(2\alpha_1 + 2\alpha_2 - \alpha_1^2 - \alpha_2^2) = 2 - 2\alpha_2 = 0$$

$$\alpha_1^* = 1 \quad \alpha_2^* = 1 \quad \alpha_3^* = 2$$

$$w^* = (w_1^*, w_2^*) = \sum_{n=1}^{3} y_n \alpha_n^* \phi(x_n)$$

$$\phi(x) = [x, x^2]^T, \; y_1 = -1, \; y_2 = -1, \; y_3 = 1$$

$$w^* = y_1 \alpha_1^* \phi(x_1) + y_2 \alpha_2^* \phi(x_2) + y_3 \alpha_3^* \phi(x_3) =$$

$$(0, -2)^T = \begin{bmatrix} 0 \\ -2 \end{bmatrix}$$

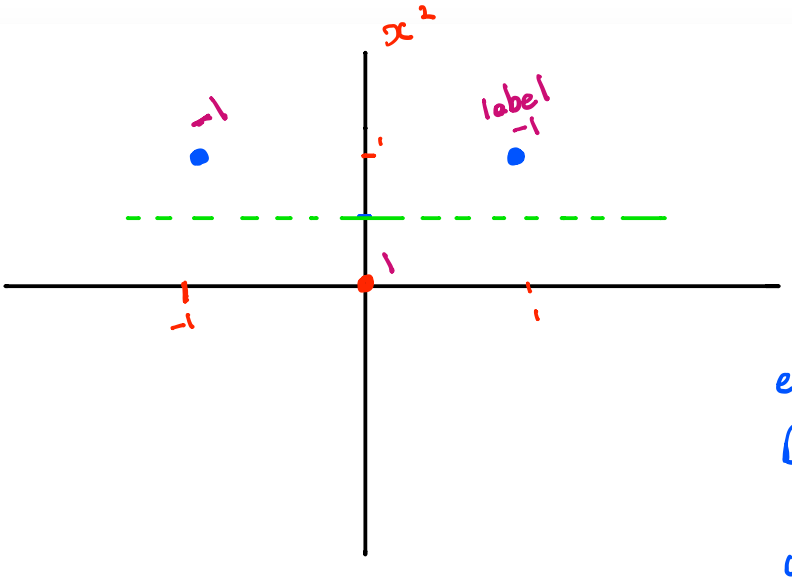
$$b^* = y_1 - w^T \phi_1(x) = -1 - \begin{bmatrix} 0 & 2 \end{bmatrix} \cdot \begin{bmatrix} -1 \\ (-1)^2 \end{bmatrix}$$

$$= -1 - (0 \cdot (-1) - 2 \cdot 1) = -1 + 2$$

$$= \boxed{1}$$

1.6 (3pts) Plot the decision boundary (which is a line) of the linear model $\mathbf{w}^{*T}\mathbf{x} + b^*$ in the two-dimensional feature space, where $\mathbf{w}^*$ and b^* are the primal solution you got from the previous question. Then circle all support vectors. Finally, plot the corresponding decision boundary in the original one-dimensional space (recall that the decision boundary is just the set of all points x such that $\mathbf{w}^{*T}\phi(x) + b^* = 0$).

a)



$\phi(x) = [x, x^2]^T$

from 1.5 $\rightarrow$

$b^* = 1$

$w^* = [0, -2]^T$

equation of decision Boundary:

$w^T \phi(x) + b = 0$

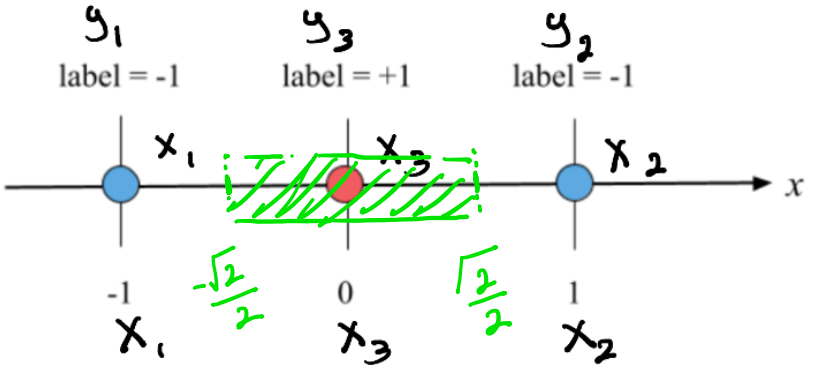
$[0, -2]^T \cdot [x, x^2]^T + 1 = 0$

$0 \cdot x + (-2)(x^2) + 1 = 0 \rightarrow -2x^2 = -1$

$x^2 = \frac{1}{2}$

b)

$x = \pm\sqrt{\frac{1}{2}} = \pm\frac{1}{\sqrt{2}} \cdot \frac{\sqrt{2}}{\sqrt{2}} = \frac{\sqrt{2}}{2}$



Q2)

Def: A function $k: R^d \times R^d \to R$ is called kernel function if there exist a function $\phi: R^d \to R^u$ so that for any $x, x' \in R^d$,

$$k(x, x') = \phi(x)^T \phi(x')$$

we have $k_1(x, x') = \phi_1(x)^T \phi_1(x')$

$$k_2(x, x') = \phi_2(x)^T \phi_2(x')$$

Assume we have L_1 for ϕ_1 & L_2 for ϕ_2 where L_1 and L_2 are dimensions.

$$\phi_1(x)^T \phi_1(x') = \sum_{l=1}^{L_1} \phi_1(x_l) \phi_1(x'_l)$$

$$\phi_2(x)^T \phi_2(x') = \sum_{m=1}^{L_2} \phi_2(x_m) \phi_2(x'_m)$$

$$k(x, x') = k_1(x, x') \cdot k_2(x, x') =$$

$$\left(\sum_{l=1}^{L_1} \phi_1(x_l) \phi_1(x'_l) \right) \left(\sum_{m=1}^{L_2} \phi_2(x_m) \phi_2(x'_m) \right) =$$

$$\sum_{l=1}^{L_1} \sum_{m=1}^{L_2} \phi_1(x_l) \phi_1(x'_l) \phi_2(x_m) \phi_2(x'_m) =$$

$$\sum_{l=1}^{L_1} \sum_{m=1}^{L_2} (\phi_1(x_l) \phi_2(x_m)) (\phi_1(x'_l) \phi_2(x'_m)) \Rightarrow$$

$\therefore \phi(x) = \phi_1(x) \phi_2(x)^T$

Problem 3

3.1 Report the training error and test error of this approach, averaged over 10 trials.

Normalized train error (linalg soln): 1.16646e-13
Normalized test error (linalg soln): 5.00664e+00

Due to the random generation of data, the results differ for each run. The normalized training error is significantly smaller than the normalized test error.

3.2 Discuss the characteristics of your plot, and also compare it to your answer to (3.1).
The following questions explore the concept of implicit regularization. This is a very active topic of research, with the idea being that optimization algorithms such as SGD can themselves act like regularizers (in the sense that they prefer the solutions to the regularized problems instead of just the original problem). There's no one correct answer we're looking for in many of these questions, and the idea is to make you think about what is happening and report your findings.

λ = {0.0005, 0.005, 0.05, 0.5, 5, 50, 500}.

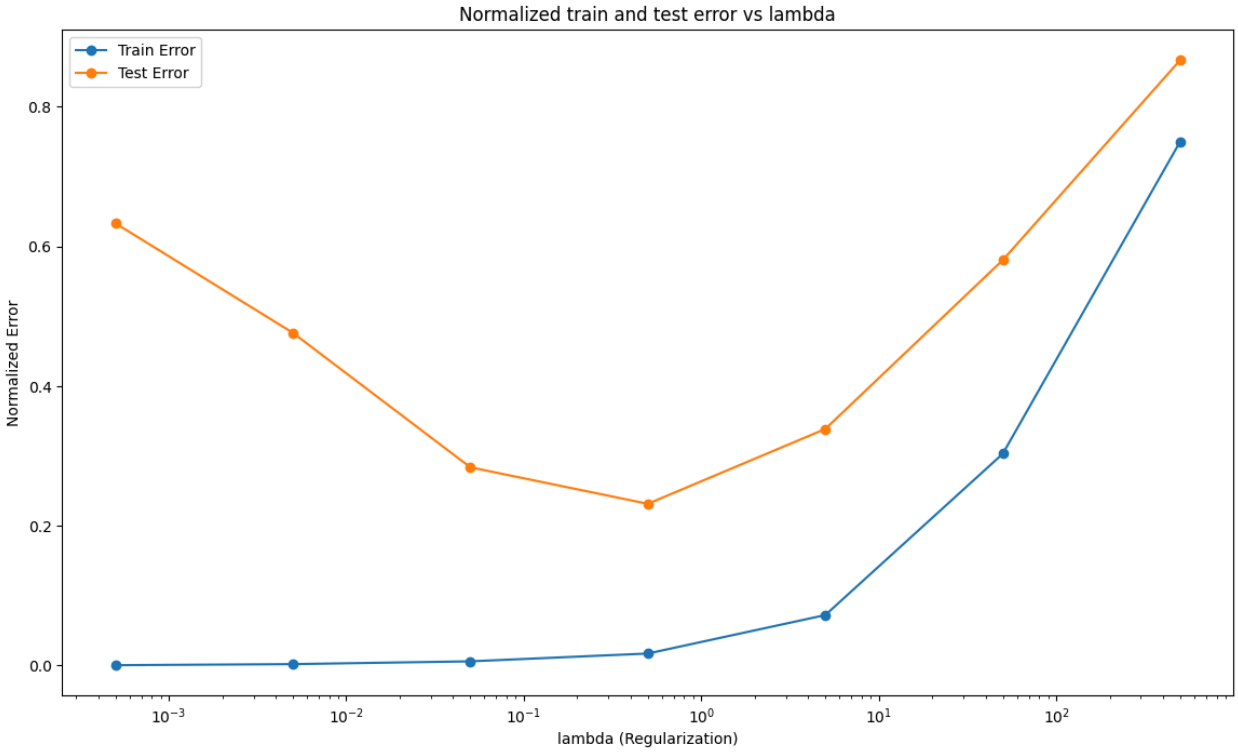
Normalized **train** error (L2 linalg soln):

[0.0005015096793651408, 0.0020217939906937653, 0.006017423588423953, 0.017183723711377205, 0.07223573716024424, 0.3031336480759129, 0.750162572611001]

Normalized **test** error (L2 linalg soln):

[0.6330528417834629, 0.47623163655710343, 0.283762886182228, 0.23149822881379403, 0.3385820975910411, 0.5803360228104979, 0.8667766246216272]

When λ = 0.5, we obtain the lowest test error, which is lower than the test error in problem 3.1. As λ increases, the training error also increases. For small λ values, the gap between the training and test errors is substantial, indicating overfitting. For λ > 0.5, both training and test errors increase, and the model fails to effectively learn from the data, performing poorly on both training and test datasets. Hence, the model might be underfitting.



3.3: Run stochastic gradient descent (SGD) on the original objective function f(w), with the initial guess of w set to be the all 0's vector. Run SGD for 1,000,000 iterations for each different choice of the step size, {0.00005, 0.0005, 0.005}. Report the normalized training error and the normalized test error for each of these three settings, averaged over 10 trials. How does the SGD solution compare with the solutions obtained using l_2 regularization? Note that SGD is minimizing the original objective function, which does *not* have any regularization. In Part (3.1) of this problem, we found the *optimal* solution to the original objective function with respect to the training data. How does the training and test error of the SGD solutions compare with those of the solution in (3.1)? Can you explain your observations?

How does the SGD solution compare with the solutions obtained using ℓ2 regularization?

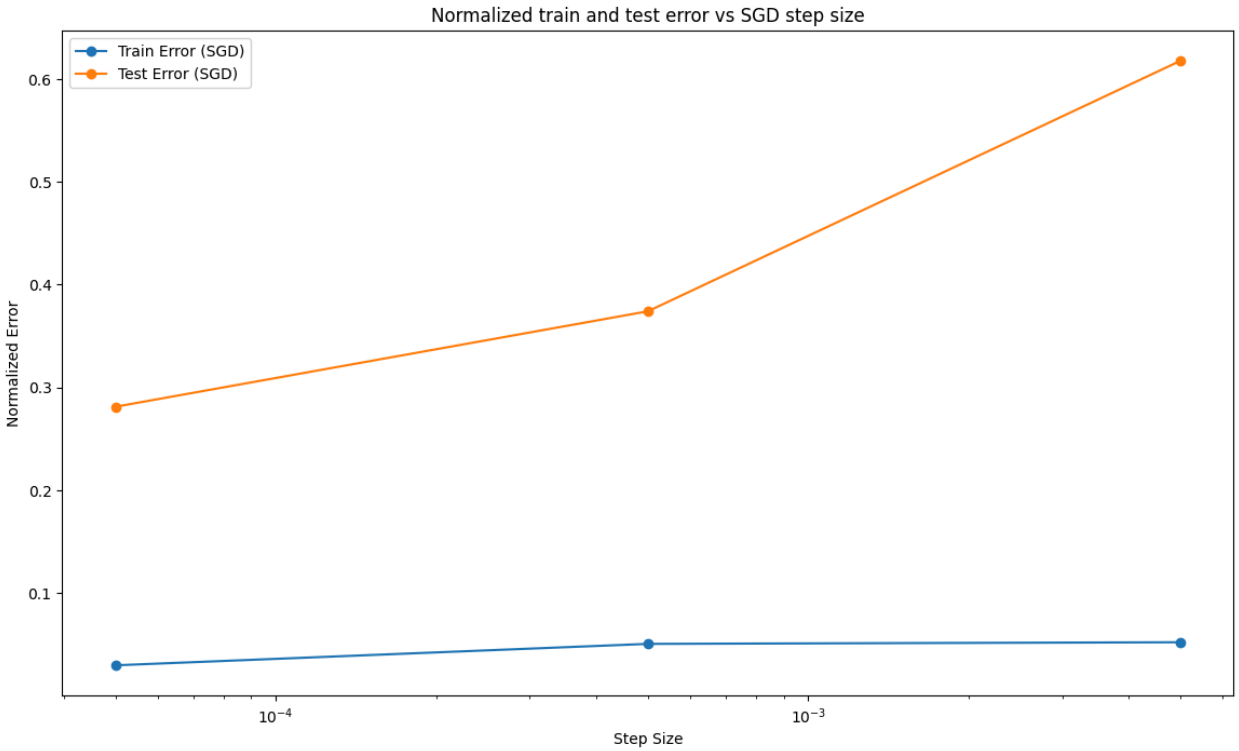
Although the best test error from SGD is slightly greater than the best test error in 3.2, the test errors for SGD are relatively lower.

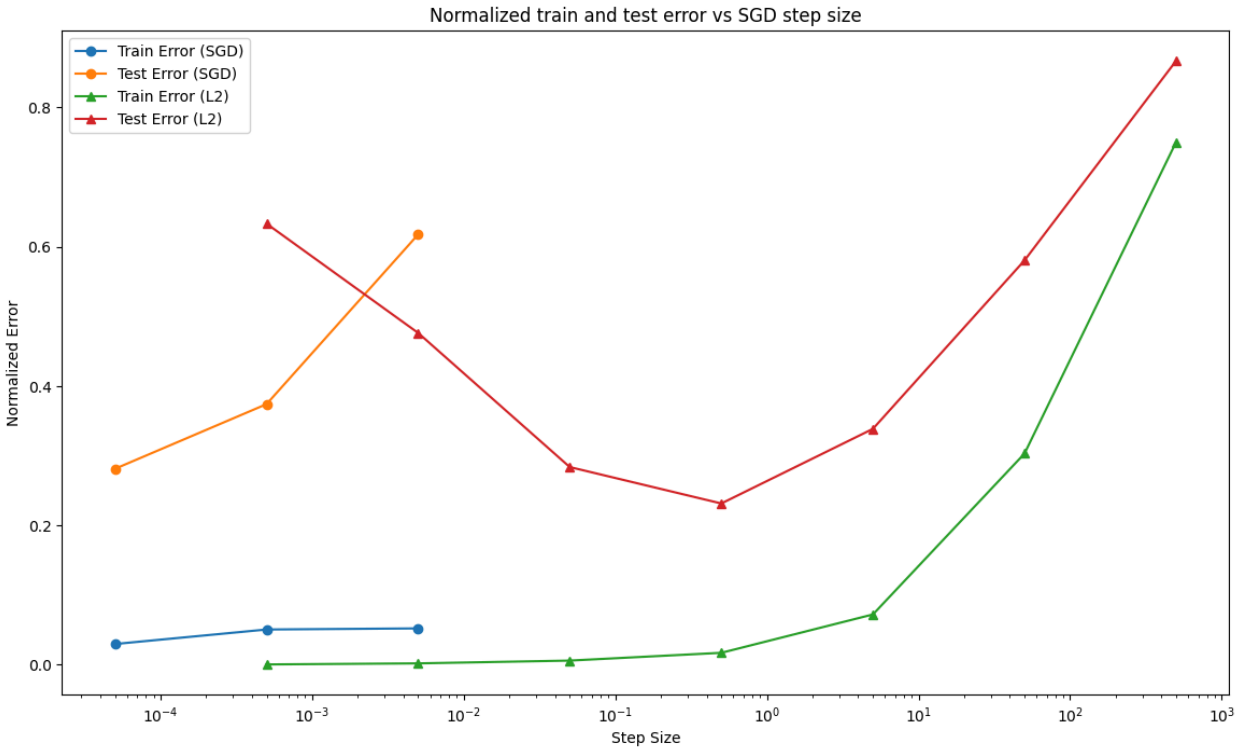
Due to the random generation of data for each run, the results may vary.

The errors for training and testing data are lower for smaller step sizes.

How do the training and test errors of the SGD solutions compare with those of the solution in (3.1)? Can you explain your observations?

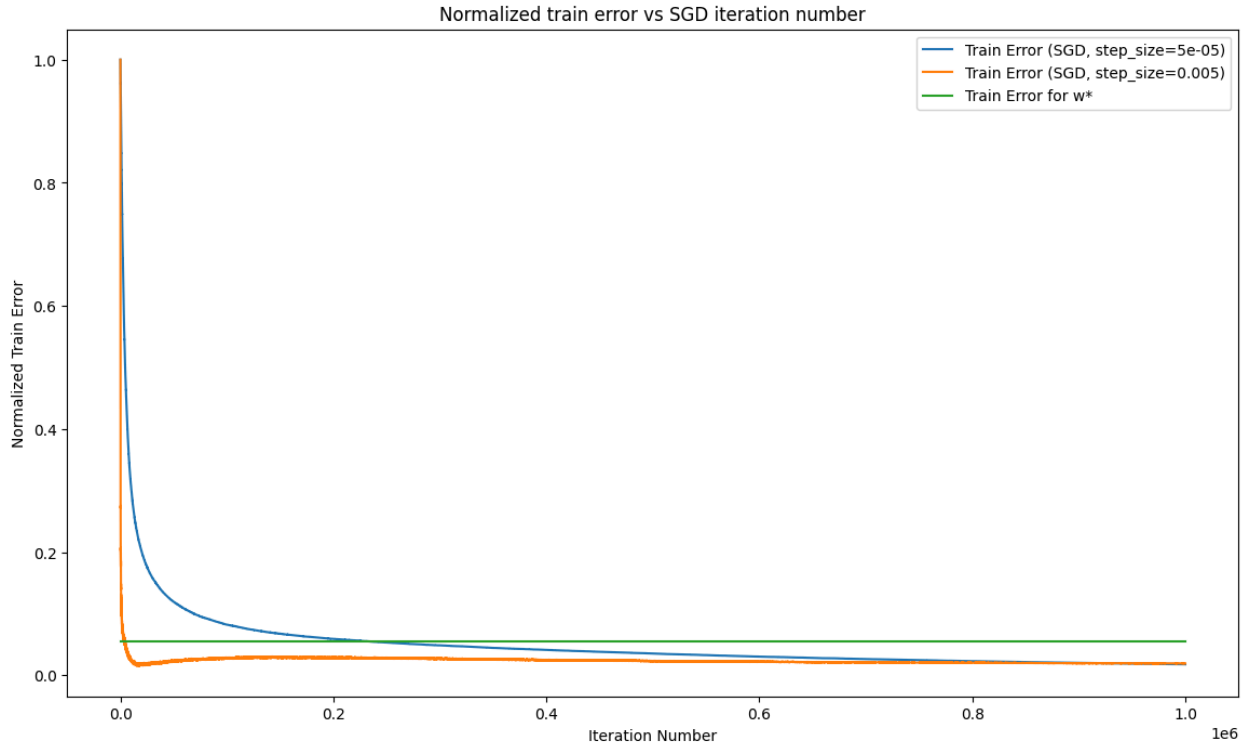
The best test error from SGD is much less than the test error in 3.1 because SGD can be used to optimize the regularized objective, thanks to its iterative nature and the way it updates model parameters.


Normalized train and test error vs SGD step size
Train Error (SGD)
Test Error (SGD)
Normalized Error
0.6
0.5
0.4
0.3
0.2
0.1
10^{-4}
10^{-3}
Step Size


Normalized train and test error vs SGD step size
Train Error (SGD)
Test Error (SGD)
Train Error (L2)
Test Error (L2)
Normalized Error
0.8
0.6
0.4
0.2
0.0
10^{-4}
10^{-3}
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}
10^{3}
Step Size

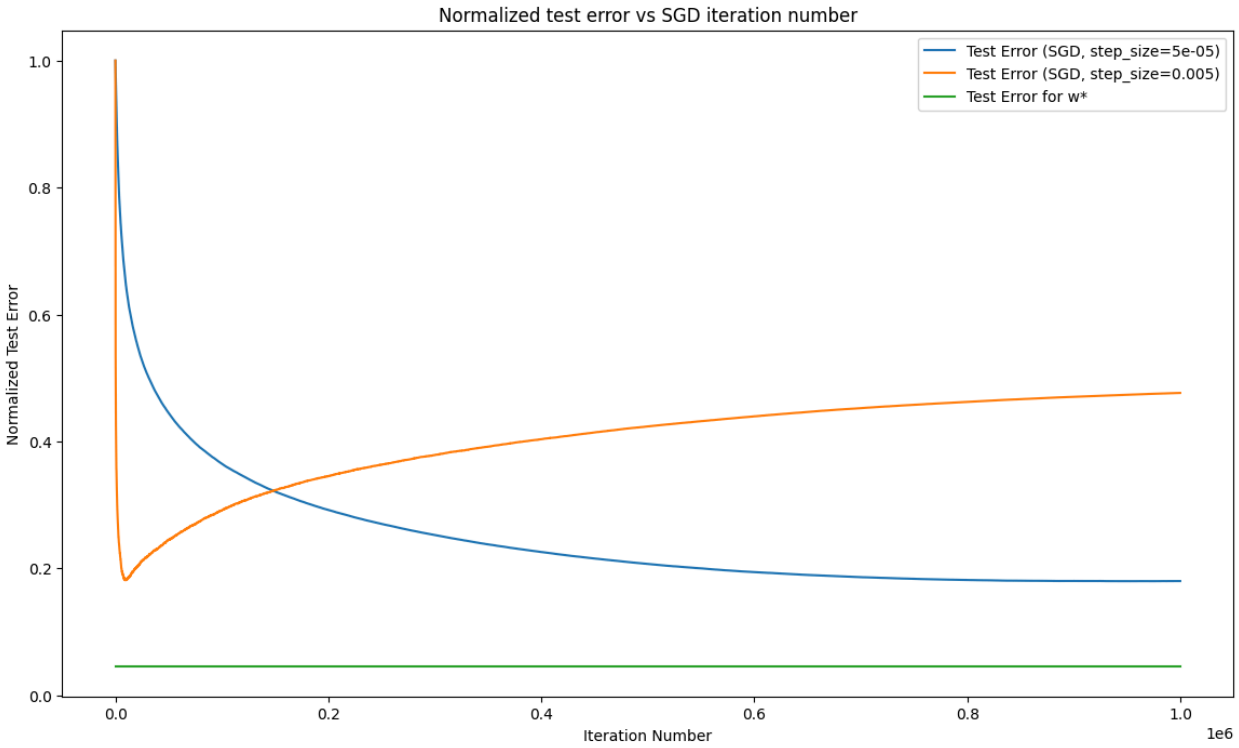
3.4 (i) Plot the normalized training error vs. the iteration number. On the plot of training error, draw a line parallel to the x-axis indicating the error ˆf (w∗) of the true model w∗.

The training error for a step size of 0.005 (orange) drops below the training error for w* at the beginning and remains below it. The training error for a step size of 5e-5 (blue) drops below the training error for w* after 2e5 iterations. Both trends indicate that the model is overfitting and adapting too well to the noise.



(ii) Plot the normalized test error vs. the iteration number. Your code might take a long time to run if you compute the test error after every SGD step—feel free to compute the test error every 100 iterations of SGD to make the plots.

After a steep drop, the test error for 0.005 (orange) rises and surpasses the test error for a step size of 5e-5 (blue), which can be a sign of overfitting. The test error for 5e-5 after around 6e5 iterations is still slightly decreasing, making it resemble a flat line. There are gaps between the SGD errors and the test error for w*.



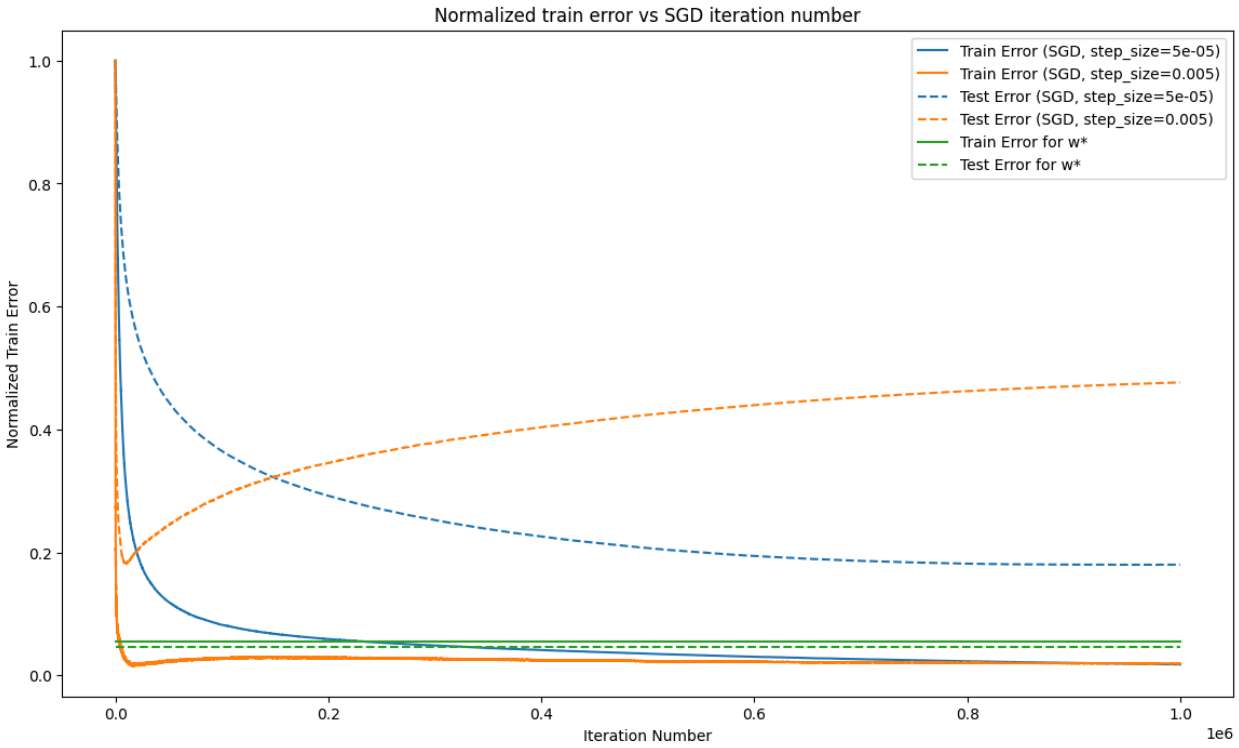
(iii) Plot the ℓ2 norm of the SGD solution vs. the iteration number.

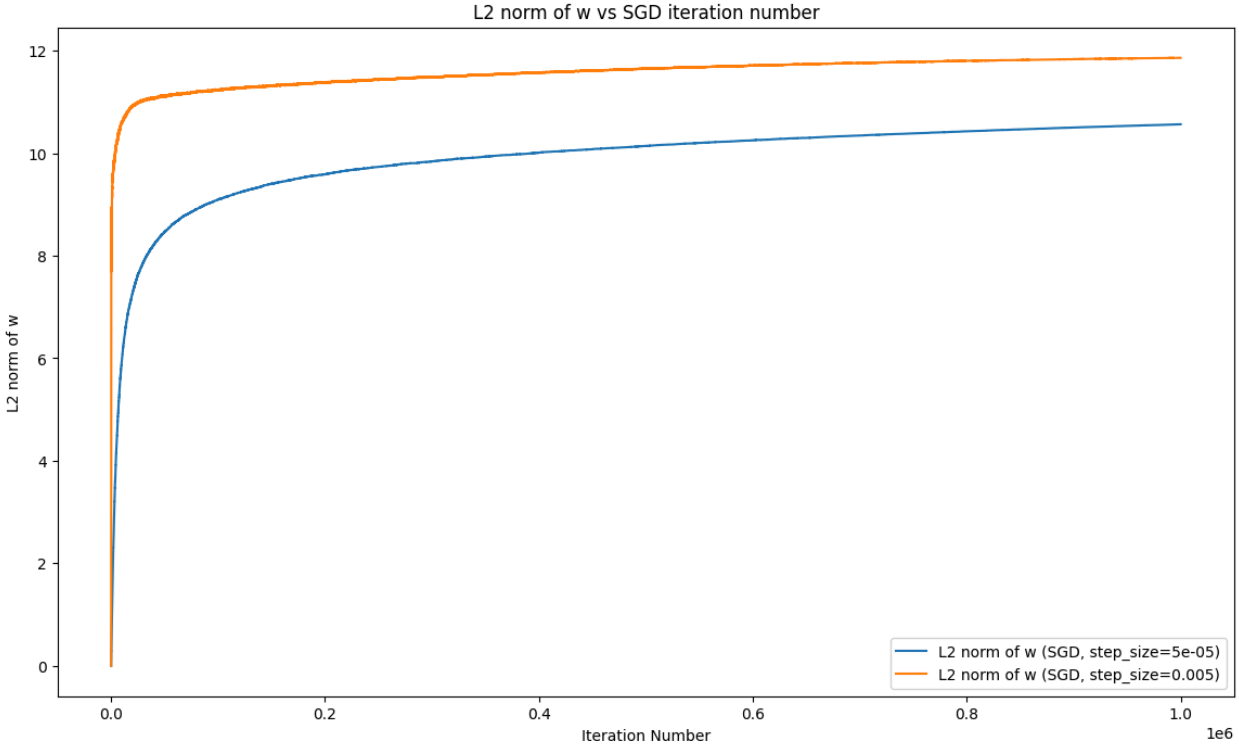
Both L2 norms increase rapidly at first and gradually become flat lines afterward and the norm for the step size 0.005 (orange) is larger. Although both norms are still increasing, the step size 5e-5 (blue) exhibits a greater increase in norm compared to the step size 0.005 (orange).

Comment on the plots. What can you say about the generalization ability of SGD with different step sizes? Does the plot correspond to the intuition that a learning algorithm starts to overfit when the training error becomes too small, i.e. smaller than the noise level of the true model so that the model is fitting the noise in the data? How does the generalization ability of the final solution depend on the ℓ2 norm of the final solution?

Although the SGD test error for the larger step size (orange) drops faster than smaller the step size (blue), the test error starts to increase afterward and surpasses the error for the smaller step size. On the other hand, the training error for the larger step size (orange) continues to decrease and stays below the error for w*, meaning that the model fits too well to the noise. Given these trends, we can conclude that the model for step size 0.005 (orange) is overfitting.

From the normalized error graph and the L2 norm graph, we observe that step size 5e-5 (blue) has a smaller norm and a lower test error after around 1.8e5 iterations. We can say smaller L2 norm might have a better generalization ability.


Normalized train error vs SGD iteration number
Train Error (SGD, step_size=5e-05)
Train Error (SGD, step_size=0.005)
Test Error (SGD, step_size=5e-05)
Test Error (SGD, step_size=0.005)
Train Error for w*
Test Error for w*
Normalized Train Error
Iteration Number
1e6


L2 norm of w vs SGD iteration number
L2 norm of w (SGD, step_size=5e-05)
L2 norm of w (SGD, step_size=0.005)
L2 norm of w
Iteration Number
1e6

3.5 Plot the average normalized training error and the average normalized test error over 10 trials vs r. Comment on the results, in relation to the results from part (3.2) where you explored different ℓ2 regularization coefficients. Can you provide an explanation for the behavior seen in this plot?

Normalized train error (SGD):
[0.5671845217864043, 0.5682738667530166, 0.5681697019072296, 0.5691155644850522, 0.6838982121877393, 0.9337740263739082, 1.232233815248266]

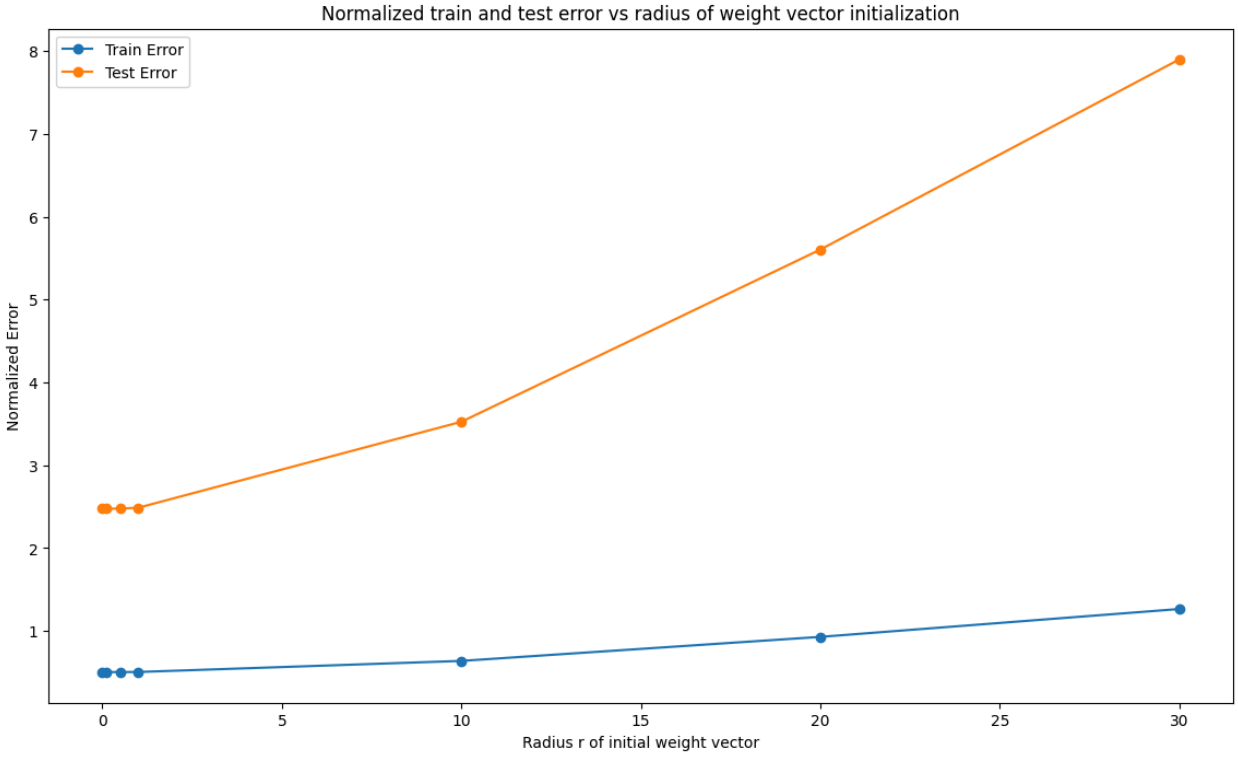
Normalized test error (SGD):
[3.7431243645040784, 3.7425923070178024, 3.7470914158790825, 3.7528951628033456, 4.8929737225077945, 7.283384055505208, 10.082784674547897]

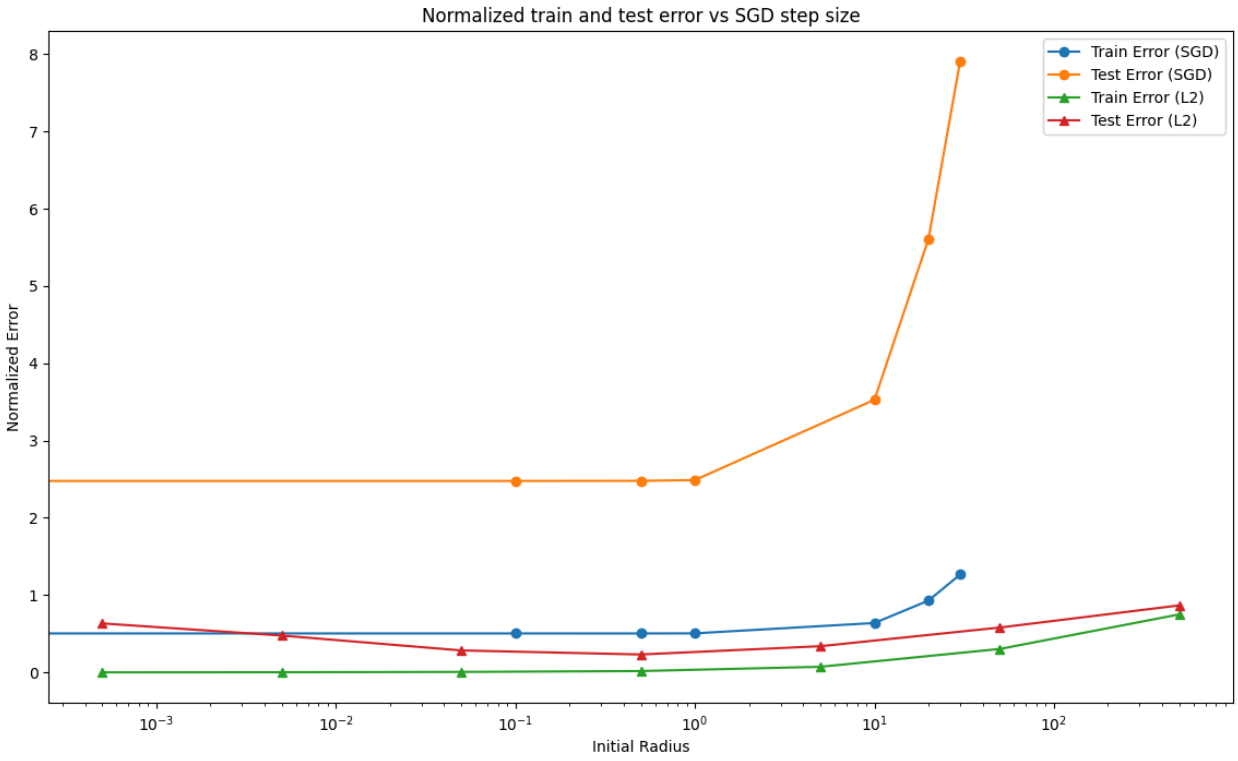
A larger radius leads to a higher normalized error. While the train error increases slowly when the radius is larger, the test error increases much more than the train error.

We also notice that for a larger radius, the norm is larger as well. Given the observation in 3.4, a smaller norm leads to lower test error with a better generalization.

Looking at this graph and the one obtained in the earlier section, the patterns of training errors and test errors can be compared.

From problem 3.2, the best lambda with the lowest test error is 0.5. In problem 3.5, the test error is also close to the lowest error for radius 0.5.




Normalized train and test error vs SGD step size
Train Error (SGD)
Test Error (SGD)
Train Error (L2)
Test Error (L2)
Normalized Error
Initial Radius
0
1
2
3
4
5
6
7
8
10^{-3}
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}

4.1 (7pts) **How do the objective function values on the train and test data relate with each other for different step sizes? Comment in 3-4 sentences.**

Due to the random generation and selection of train and test datasets, results may be very slightly different after each run.

Blue graph or 0.0005

The mean logistic losses for the training and test dataset continue to decrease for all iterations and have a converging trend.

The gap between the training MLL and test MLL starts to increase after ~1200 iterations, showing there might be a potential overfitting.

Orange graph or 0.005

It has the lowest training loss after ~1000 iterations.

The training Mean Logistic Loss (MLL) after ~2500 iterations stays flat.

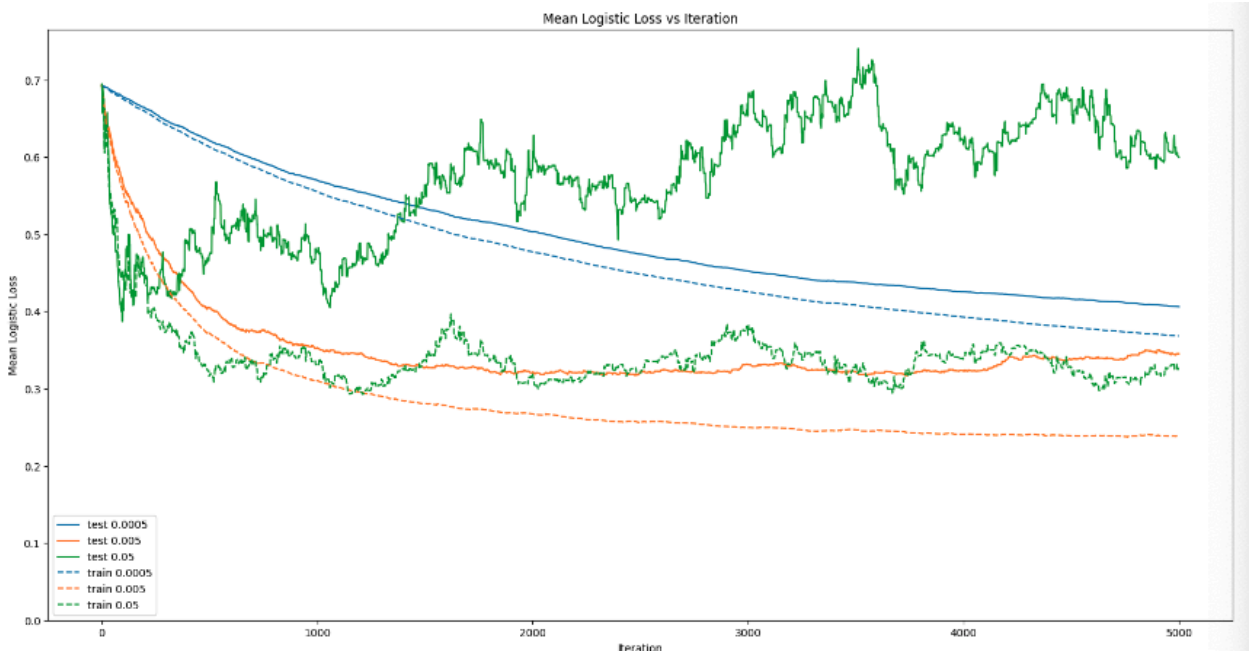
Although the test MLL decreases first (before ~1400 iterations), the gap between training and test MLL increases (after ~1400 iterations) which shows the model is overfitting,

Green graph or 0.05

The training loss drops rapidly before 500 iterations. On the other hand, the test loss first drops and quickly increases after 200 iterations.

The test MLL is always bounded between 0.4 and 0.7, whereas the training MLL is bounded between 0.3 and 0.4.

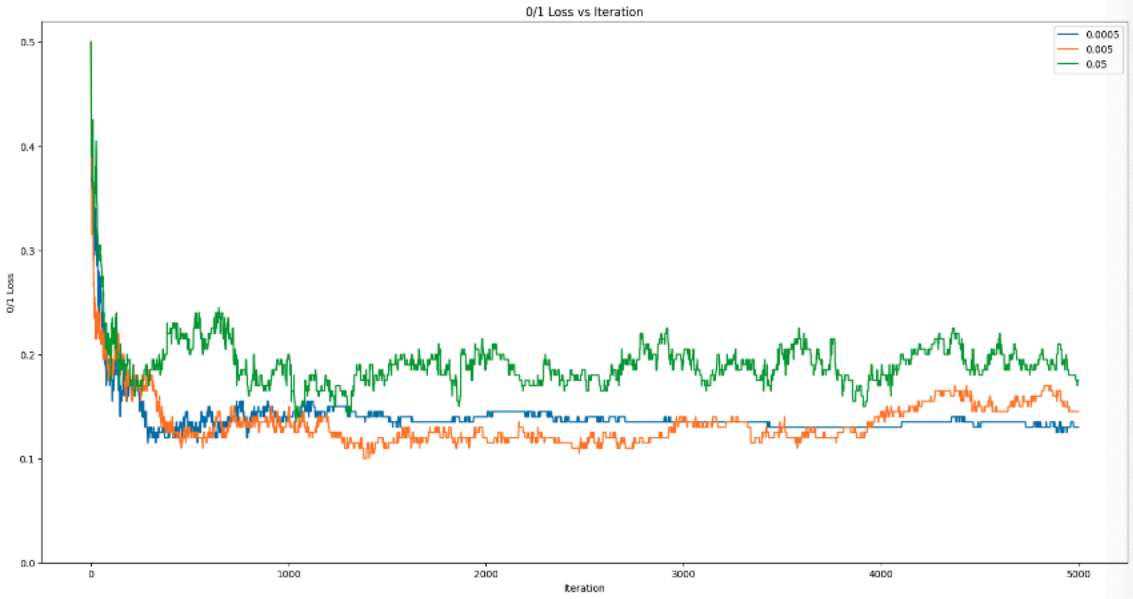
Compared to other graphs, the magnitude of green line oscillation is greater.



4.2 (2pts) **report the step size that had the lowest final 0-1 loss on the test set and the corresponding value of the 0-1 loss.**

The step size of 0.0005 has the lowest 0-1 loss equal to 0.13 (not rounding)

```
[25]: <matplotlib.legend.Legend at 0x10c00a7a0>
```

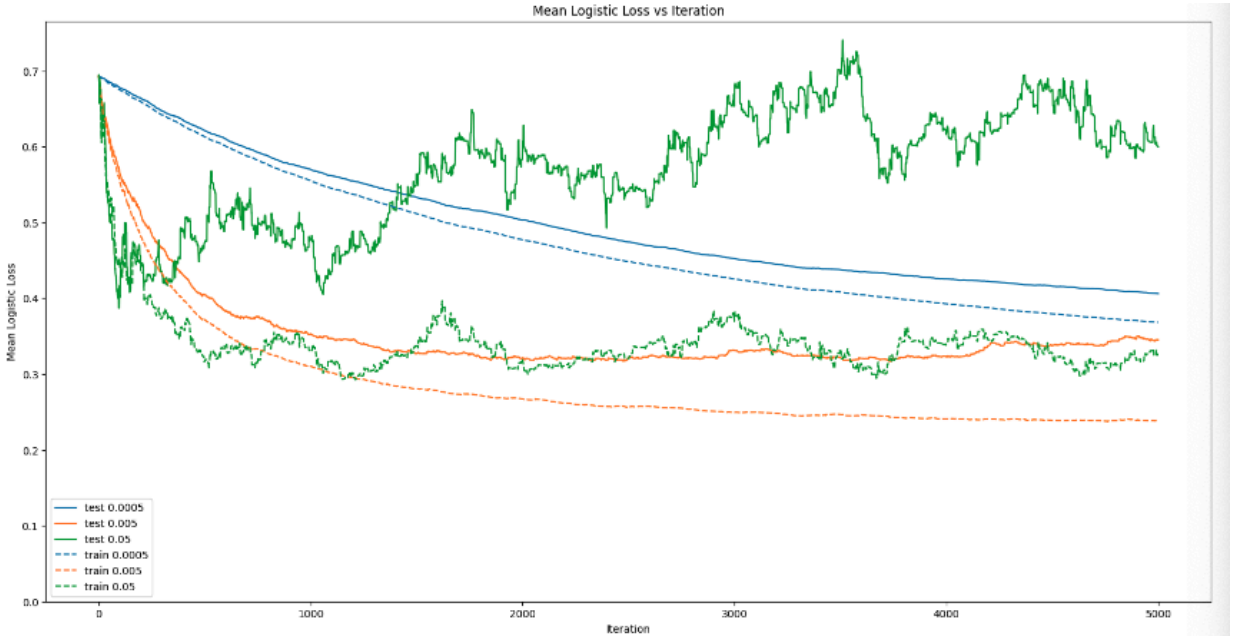


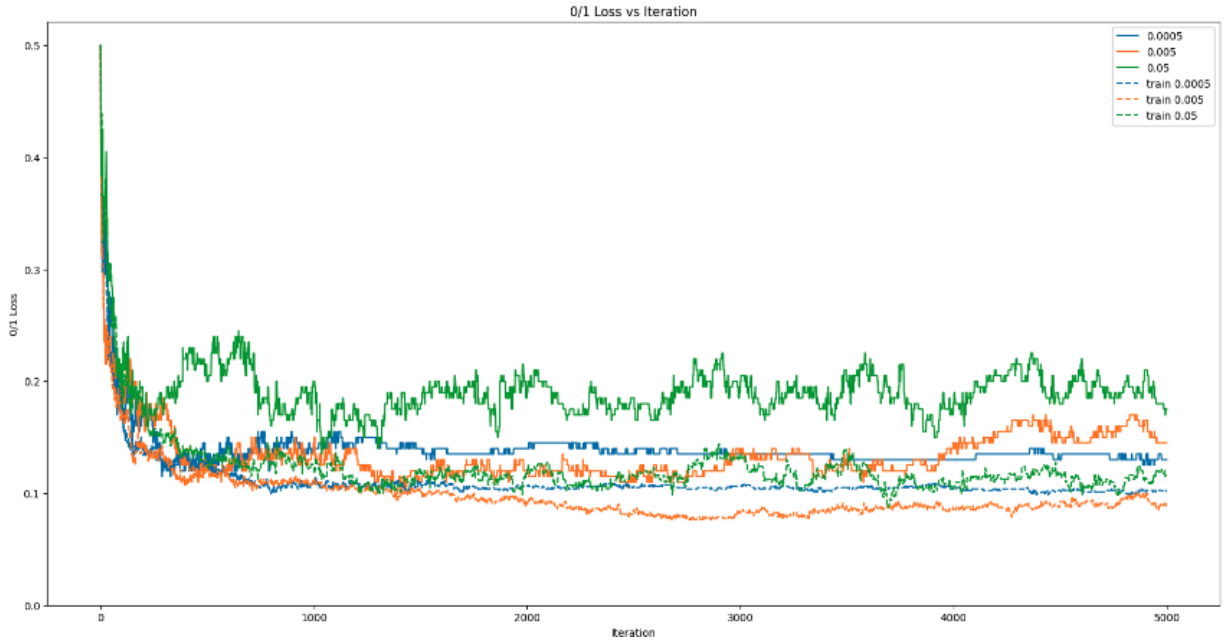

```
[24]: loss01_vals1[-1], loss01_vals2[-1], loss01_vals3[-1]
[24]: (0.13, 0.145, 0.175)
```

4.3 (2pts) Comment on how well the logistic loss act as a surrogate for the 0-1 loss.

The logistic losses in general tend to be greater than 0-1 losses. The logistic losses almost never fall below 0.3, while the 0-1 losses after approximately 200 iterations fall between around 0.1 and 0.2.

When we want to choose the best step size, the answers for logistic loss and 0-1 loss are different. While logistic loss is necessary for optimizing the model, it is more suitable to use 0-1 loss to choose the best model.





5.1 (2pts) Notice that MOON and CIRCLES are not linearly separable. Do linear classifiers do well on these? How does SVM with RBF kernel do on these? Comment on the difference.

No, Since Moon and circle are not linearly separable, Linear SVM and Logistic regression with and without regularization parameter are not doing a good job. These two methods are not suitable for this kind of data. However, since LINEARLY_SEPERABALE data is linearly separable, we see Linear SVM and Logistic regression performs much better.

Since the RBF SVM kernel maps the input data into a higher-dimensional space, we see better non-linear decision boundaries separating the two classes. RBF SVM does a good job on all three datasets separating the classes.

On LINEARLY_SEPERABALE data logistic regression does a little better job in separating classes with 95% accuracy. Linear SVM and RBF SVM accuracy scores are very close.

5.2 (2pts) Try various values of the penalty term C for the SVM with linear kernel. On the LINEARLY SEPARABLE dataset, how does the train and test set accuracy relate to C? On the LINEARLY SEPARABLE dataset, how does the decision boundary change?

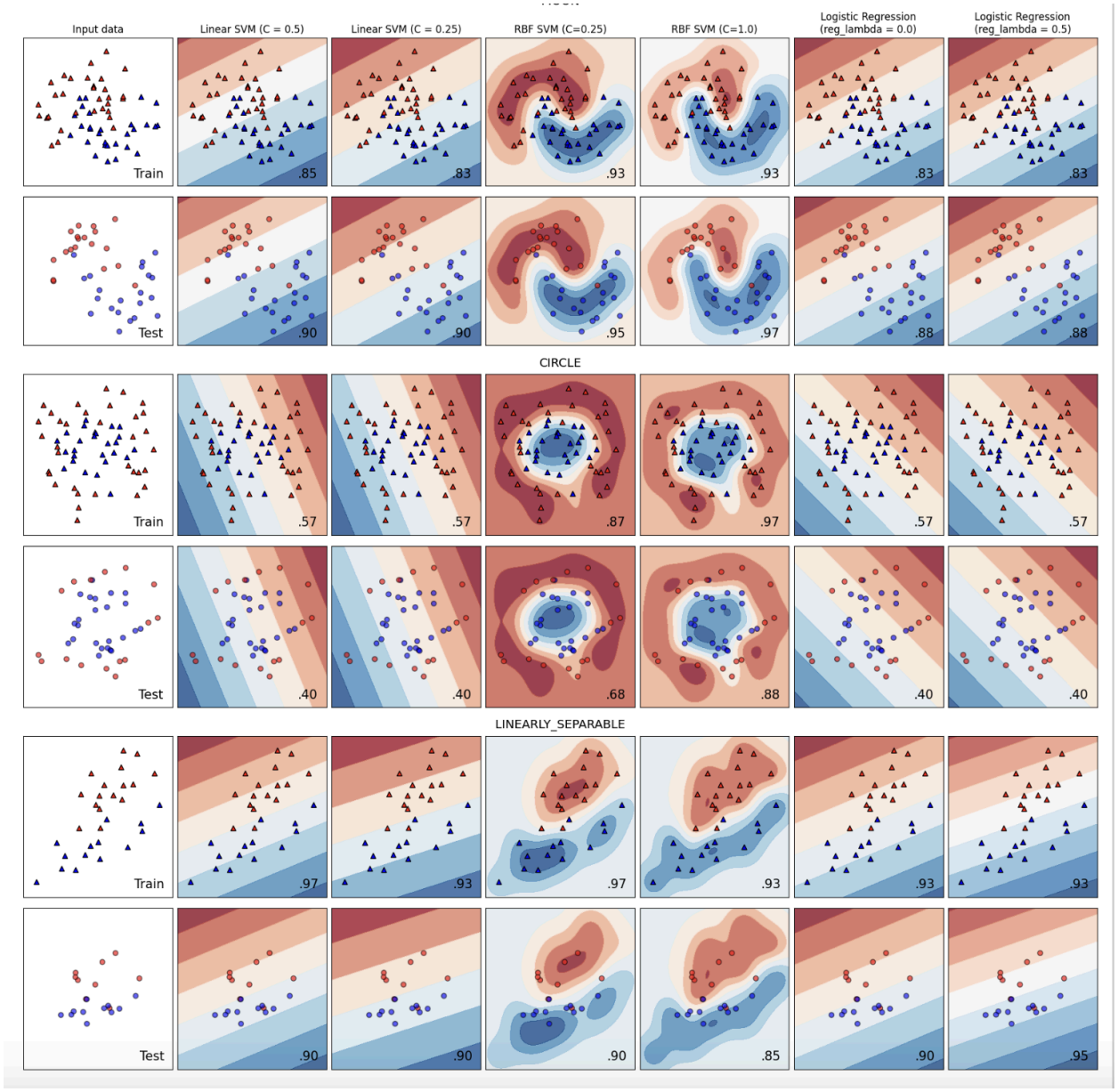
Column # 2 shows our experiments.

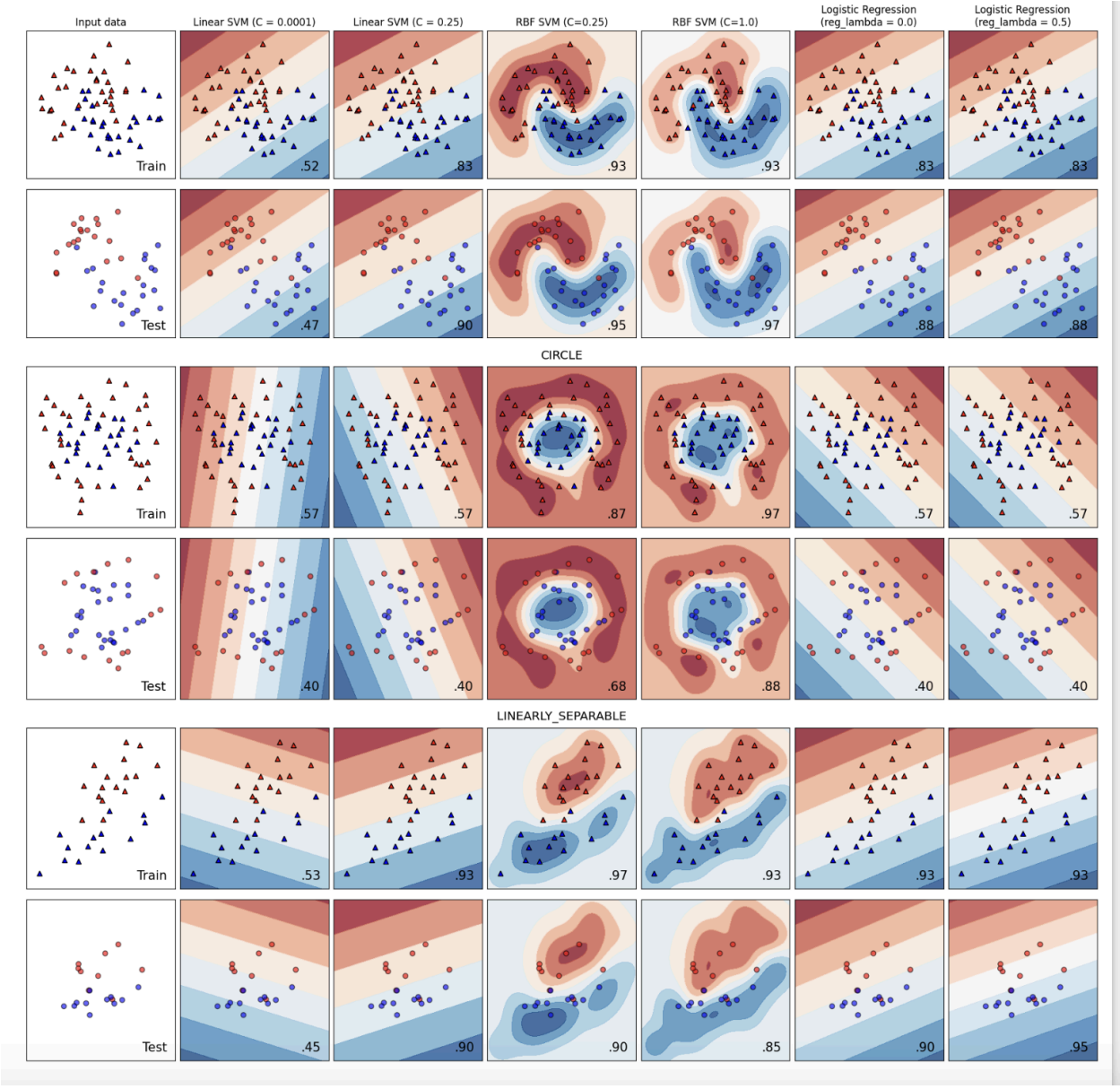
C is referring to the penalty term for misclassifying training data.

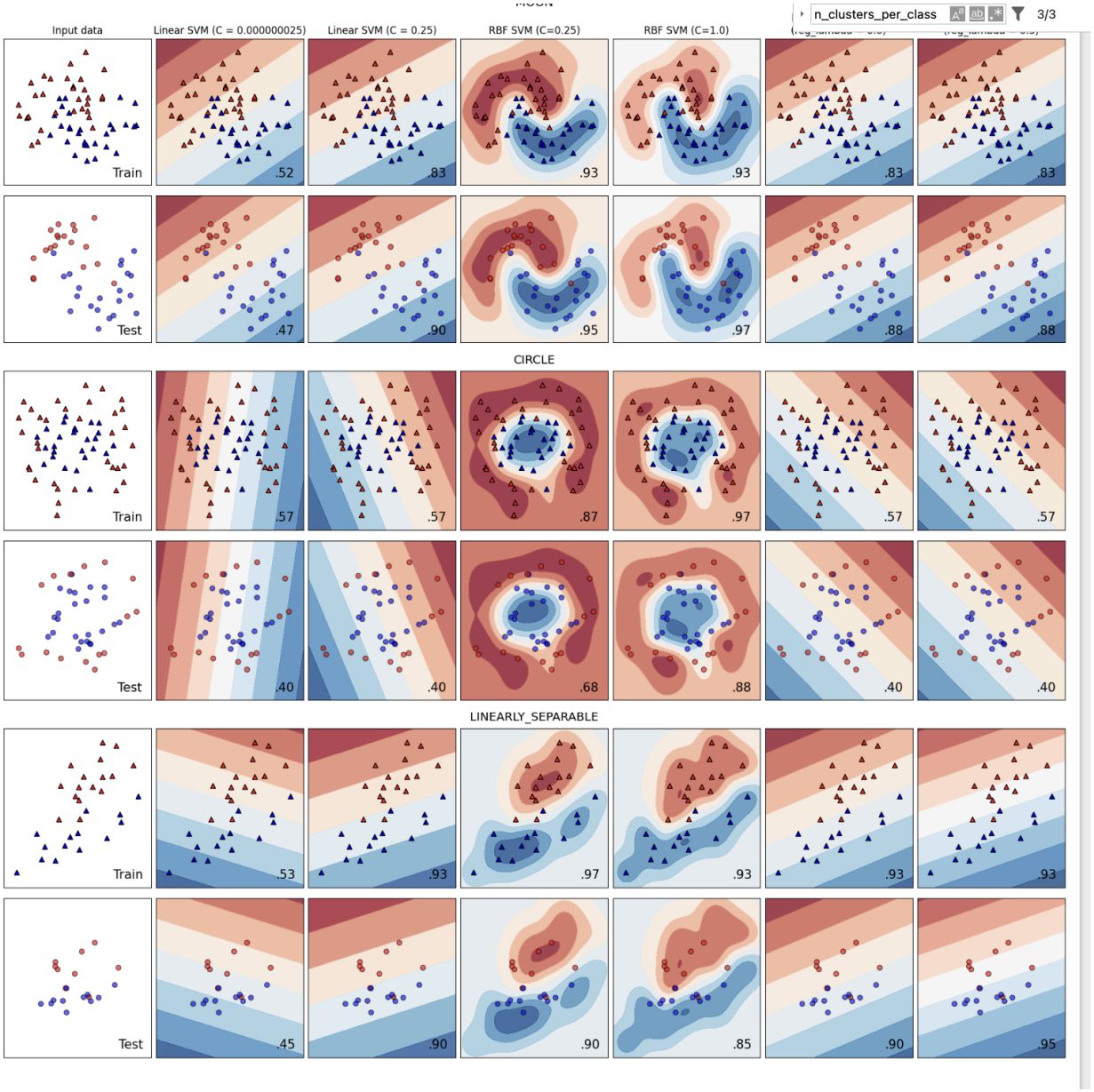
We tried C=0.0001 and 0.000000025 and the accuracy of training dropped to 53% and test dropped to 45%.

In the next experiment, we increased the value of C to 0.5 and the accuracy of training dropped to 97% and test dropped to 90%.

We can conclude: the lower value for C => the lower accuracy rate for both train and test => weaker decision boundary selection by the model


Input data
Linear SVM (C = 0.5)
Linear SVM (C = 0.25)
RBF SVM (C=0.25)
RBF SVM (C=1.0)
Logistic Regression (reg_lambda = 0.0)
Logistic Regression (reg_lambda = 0.5)
Train
Test
.85
.83
.93
.93
.83
.83
.90
.90
.95
.97
.88
.88
CIRCLE
Train
Test
.57
.57
.87
.97
.57
.57
.40
.40
.68
.88
.40
.40
LINEARLY_SEPARABLE
Train
Test
.97
.93
.97
.93
.93
.93
.90
.90
.90
.85
.90
.95


Input data
Linear SVM (C = 0.0001)
Linear SVM (C = 0.25)
RBF SVM (C=0.25)
RBF SVM (C=1.0)
Logistic Regression (reg_lambda = 0.0)
Logistic Regression (reg_lambda = 0.5)
Train
.52
.83
.93
.93
.83
.83
Test
.47
.90
.95
.97
.88
.88
CIRCLE
Train
.57
.57
.87
.97
.57
.57
Test
.40
.40
.68
.88
.40
.40
LINEARLY_SEPARABLE
Train
.53
.93
.97
.93
.93
.93
Test
.45
.90
.90
.85
.90
.95


n_clusters_per_class
3/3
Input data
Linear SVM (C = 0.000000025)
Linear SVM (C = 0.25)
RBF SVM (C=0.25)
RBF SVM (C=1.0)
Train
Test
.52
.83
.93
.93
.83
.83
.47
.90
.95
.97
.88
.88
CIRCLE
Train
Test
.57
.57
.87
.97
.57
.57
.40
.40
.68
.88
.40
.40
LINEARLY_SEPARABLE
Train
Test
.53
.93
.97
.93
.93
.93
.45
.90
.90
.85
.90
.95

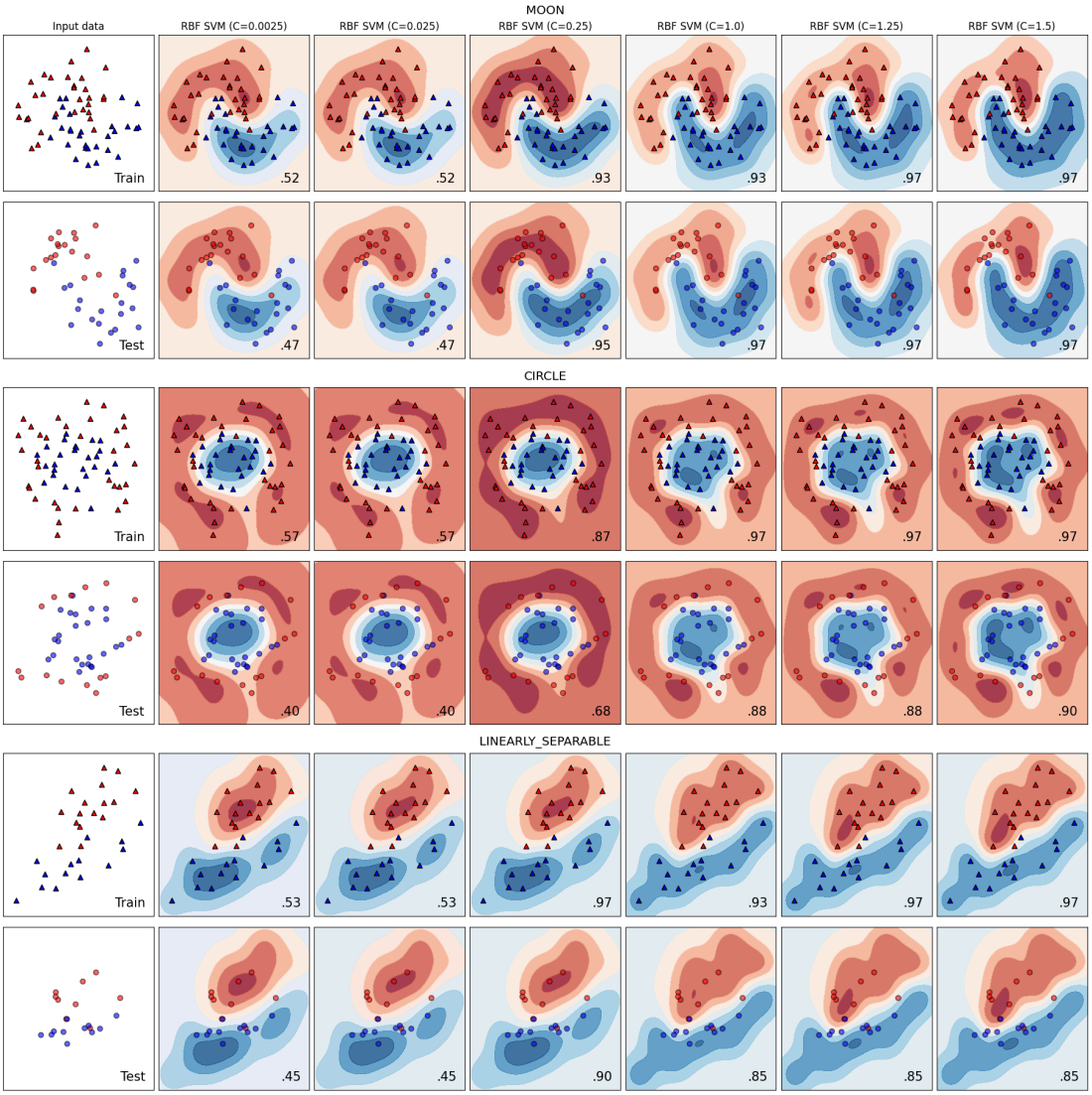
5.3 (2pts) Try various values of the penalty term C for the SVM with the RBF kernel. How does the train and test set accuracy relate to C? How does the decision boundary change?

Going from left to the right columns, we show the decision boundaries as C increases.

The smaller or weaker C (regularization parameter) has a weaker decision boundary and accuracy sores for both training and test.

As C become larger (from left to right) the accuracy score increases resulted from having a better decision boundary.

This can be more useful for the MOON and CIRCLE datasets.

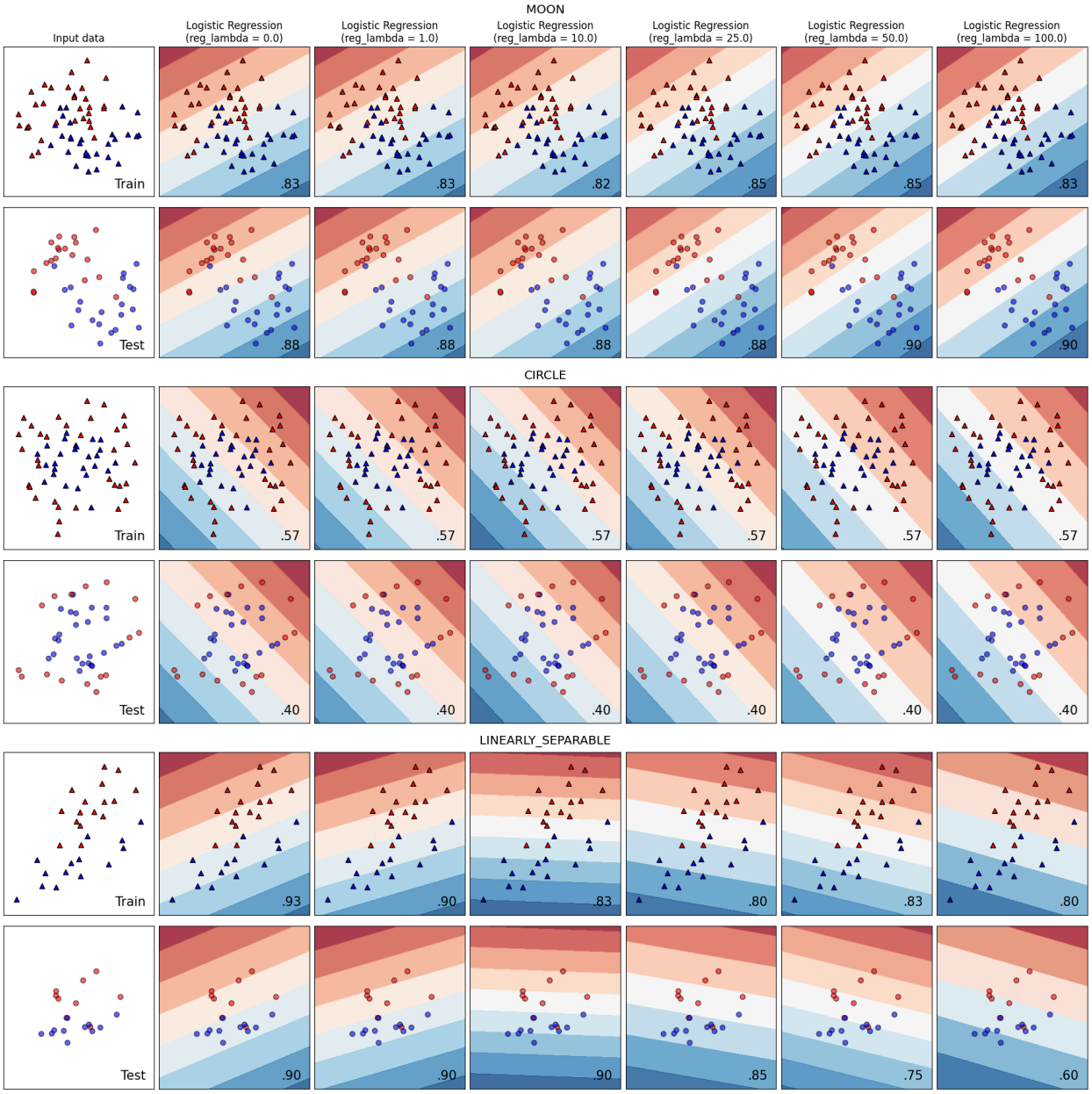


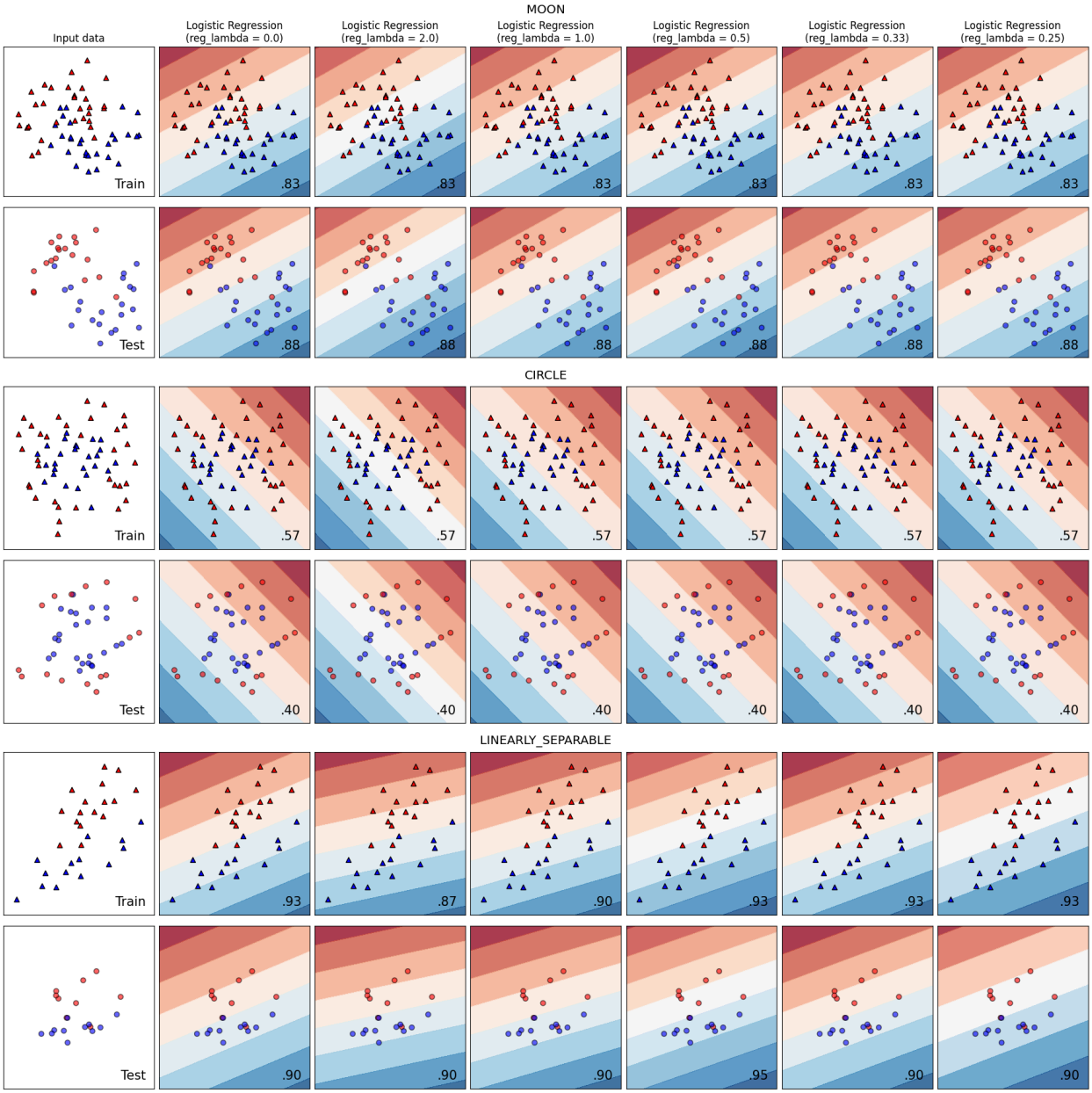
5.4 (2pts) Try various values of C for Logistic Regression (Note: C is the inverse regularization strength). Do you see any effect of regularization strength on Logistic Regression? Hint: Under what circumstances do you expect regularization to affect the behavior of a Logistic Regression classifier?

After updating the reg_labmda from a small to larger number and vice versa, the accuracy stayed the same or the change was very small.

We can say since the dataset size for MOON, CIRCLE, and LINEARLY_SEPERABALE are relatively small, reg_labmda does not play an important role here.

According to our research, if data has more dimensions and the size is relatively larger, reg_labmda can be useful in improving the accuracy




MOON
Input data
Logistic Regression (reg_lambda = 0.0)
Logistic Regression (reg_lambda = 2.0)
Logistic Regression (reg_lambda = 1.0)
Logistic Regression (reg_lambda = 0.5)
Logistic Regression (reg_lambda = 0.33)
Logistic Regression (reg_lambda = 0.25)
Train
.83
.83
.83
.83
.83
.83
Test
.88
.88
.88
.88
.88
.88
CIRCLE
Train
.57
.57
.57
.57
.57
.57
Test
.40
.40
.40
.40
.40
.40
LINEARLY_SEPARABLE
Train
.93
.87
.90
.93
.93
.93
Test
.90
.90
.90
.95
.90
.90