CS 771 Minor Assignment-1

Chadhurbala R V (251110022)
Nishu Kumar (251110054)
Pranjal (251110058)

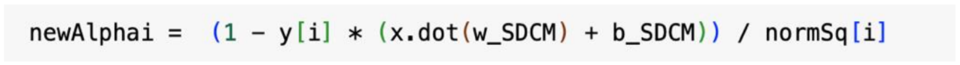
Shubham Rawat (251110070)

Department of Computer Science and Engineering
Indian Institute of Technology Kanpur, Kanpur, UP

Question 1: Discover all coding errors made by Melbo in implementing the SDCM algorithm. Recall that all coding errors are confined to cell number 5 of the Jupyter notebook. Describe the errors in your report along with a description of why you think it is an error.

Answer:

Error 1: Incorrect update of alpha

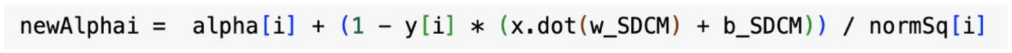


```
newAlphai =  (1 - y[i] * (x.dot(w_SDCM) + b_SDCM)) / normSq[i]
```

Figure 1: Incorrect update of alpha.

Here, the update expression directly assigns a new value to α_i, discarding the contribution of the previous α_i. This is incorrect because the optimization works iteratively, where each update should adjust and improve the current value of α_i instead of completely replacing it. The correct update expression for α_i is given as follows:

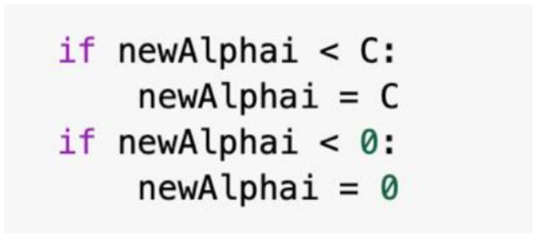
$$\alpha_i^{\text{new}} = \alpha_i + \frac{1 - y_i \left(w^\top x_i + b\right)}{\|x_i\|^2 + 1}$$



```
newAlphai =  alpha[i] + (1 - y[i] * (x.dot(w_SDCM) + b_SDCM)) / normSq[i]
```

Figure 2: Correct update of alpha.

Error 2: Box Constraints

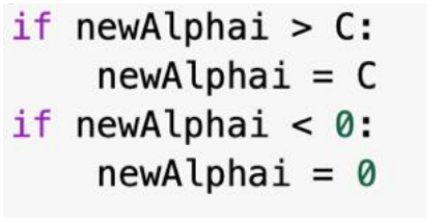


```
if newAlphai < C:
    newAlphai = C
if newAlphai < 0:
    newAlphai = 0
```

Figure 3: Incorrect application of box constraints.

The box constraints were not applied correctly. We need to clip the updated value of α_i^{new} such that it always lies within the range $[0, C]$. When $\alpha_i^{\text{new}} > C$, we set $\alpha_i^{\text{new}} = C$, and when $\alpha_i^{\text{new}} < 0$, then we set $\alpha_i^{\text{new}} = 0$.

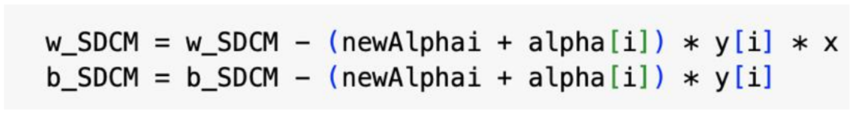
$$\alpha_i^{\text{new}} = \begin{cases} C & \text{if } \alpha_i^{\text{new}} > C, \\ 0 & \text{if } \alpha_i^{\text{new}} < 0, \\ \alpha_i^{\text{new}} & \text{otherwise.} \end{cases}$$



```
if newAlphai > C:
    newAlphai = C
if newAlphai < 0:
    newAlphai = 0
```

Figure 4: Correct application of box constraints.

Error 3: Wrong update of w and b during Book-keeping Step



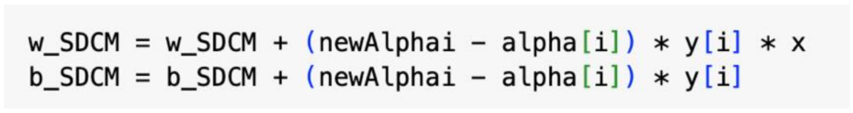
```
w_SDCM = w_SDCM - (newAlphai + alpha[i]) * y[i] * x
b_SDCM = b_SDCM - (newAlphai + alpha[i]) * y[i]
```

Figure 5: Incorrect update of w and b during the book-keeping step.

The above implementation of the book-keeping step is incorrect. Instead of adding the weighted change in α to the previous weights and bias, the code subtracts the weighted sum of the old and new α values from the previous weights and bias, which is logically incorrect. The standard update of w and b with respect to the change in α should be carried out using the following equations:

$$w \leftarrow w + (\alpha_i^{\text{new}} - \alpha_i)\, y_i\, x_i$$

$$b \leftarrow b + (\alpha_i^{\text{new}} - \alpha_i)\, y_i$$

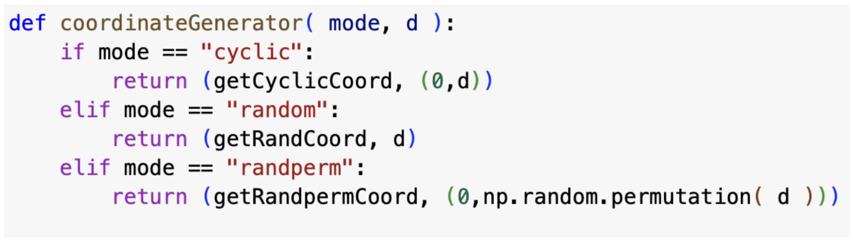


```
w_SDCM = w_SDCM + (newAlphai - alpha[i]) * y[i] * x
b_SDCM = b_SDCM + (newAlphai - alpha[i]) * y[i]
```

Figure 6: Correct Update of w and b during Book-keeping Step

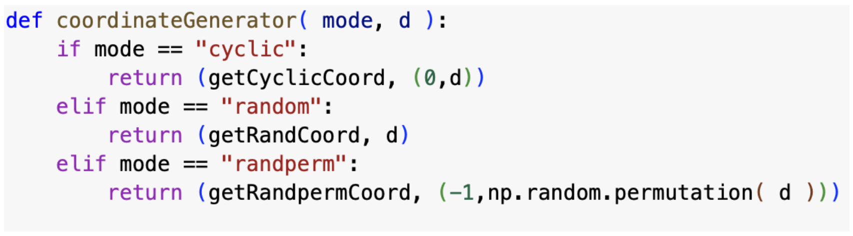
Error 4: Incorrect state initialization for `getRandpermCoord`:

In function `coordinateGenerator`, when calling the function `getRandpermCoord`, the state is initialized as `(0, np.random.permutation(d))`. When `getRandpermCoord` is called for the first time, it skips the permutation value at index zero. This can be fixed by initializing the state as `(-1, np.random.permutation(d))`.



```
def coordinateGenerator( mode, d ):
    if mode == "cyclic":
        return (getCyclicCoord, (0,d))
    elif mode == "random":
        return (getRandCoord, d)
    elif mode == "randperm":
        return (getRandpermCoord, (0,np.random.permutation( d )))
```

Figure 7: Incorrect initialization of state.



```
def coordinateGenerator( mode, d ):
    if mode == "cyclic":
        return (getCyclicCoord, (0,d))
    elif mode == "random":
        return (getRandCoord, d)
    elif mode == "randperm":
        return (getRandpermCoord, (-1,np.random.permutation( d )))
```

Figure 8: Correct initialization of state.

Question 2: Once you have corrected Melbo's mistakes, what test accuracy do you get? Mention this and show what primal and dual objective curve do you get. Do not change any hyper parameter values or settings (e.g. value of C, number of training points, initialization, horizon etc) in any cell other than cell number 5 while answering this question.

Answer:

Test Accuracy: 95.10%.

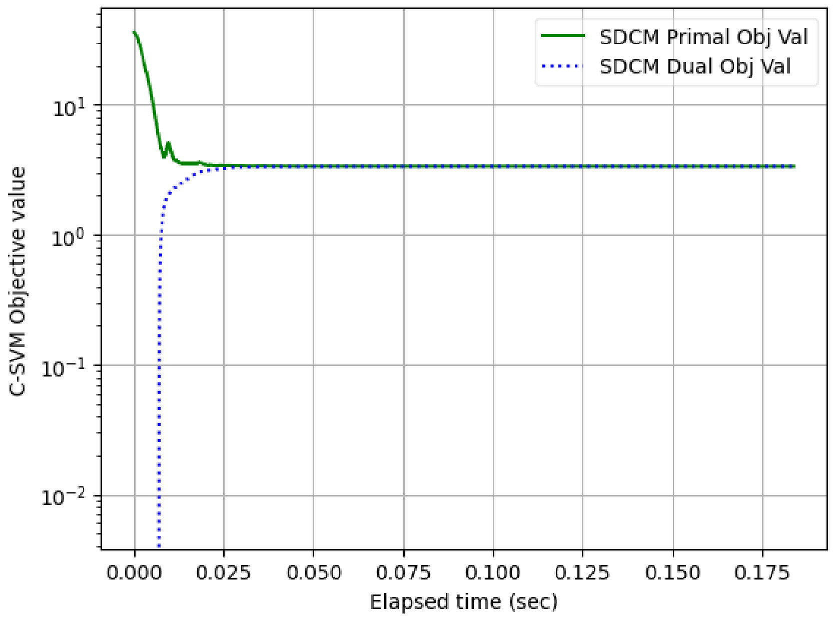


Figure 9: Primal and Dual Objective Curve

Question 3: Change the value of C between very small values (e.g. 0.00001) and very large values (e.g. 10000) without changing any of the other hyperparameters/settings. What changes do you observe in the test accuracy? What changes do you observe in the primal-dual objective curve? Please do these experiments after you have fixed Melbo's errors.

Answer:

C	Test Accuracy (%)	Final Primal Value	Final Dual Value	Duality Gap
0.00001	93.15	9.965×10^{-3}	9.965×10^{-3}	-3.469×10^{-18}
0.0001	93.15	9.650×10^{-2}	9.650×10^{-2}	-2.776×10^{-17}
0.001	93.52	6.988×10^{-1}	6.988×10^{-1}	7.550×10^{-15}
0.01	95.06	3.354×10^{0}	3.354×10^{0}	2.981×10^{-4}
0.1	96.48	1.479×10^{1}	1.460×10^{1}	1.936×10^{-1}
1	97.09	8.158×10^{1}	4.951×10^{1}	3.207×10^{1}
10	96.93	7.688×10^{2}	-1.337×10^{1}	7.822×10^{2}
100	96.55	6.176×10^{4}	-2.258×10^{4}	8.434×10^{4}
1000	95.90	7.387×10^{6}	-2.617×10^{6}	1.000×10^{7}
10000	96.92	4.405×10^{8}	-2.465×10^{8}	6.870×10^{8}

Table 1: Observed Test Accuracy, Final Primal, Final Dual, and Duality Gap values for different C.

A lower value of C allows slack (misclassification). For a higher value of C, misclassifications are penalized heavily.

As we increase the value of C from a very small value (C = 0.00001), the test accuracy increases to a maximum at C = 1, after which it decreases slightly with increasing value of C. As we increase the value of C, the difference between the final primal and dual value (duality gap) increases. For small values of C, up to C = 0.01, the duality gap is much smaller, which is almost 0. For a larger value of C, this duality gap becomes larger, which shows that the solver did not reach the optimal value with the given iteration of 20000.

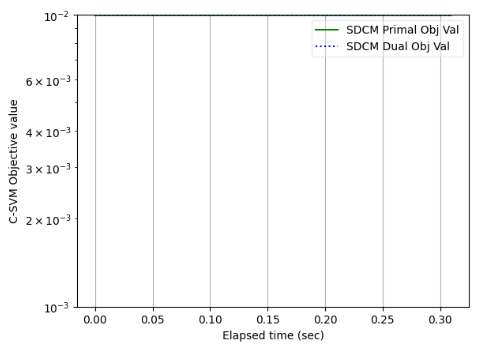


Figure 10: Primal and Dual Objective Curve (C = 0.00001)

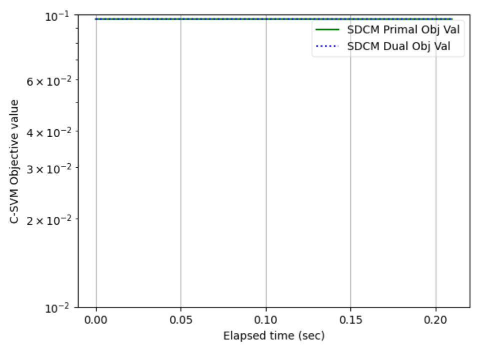


Figure 11: Primal and Dual Objective Curve (C = 0.0001)

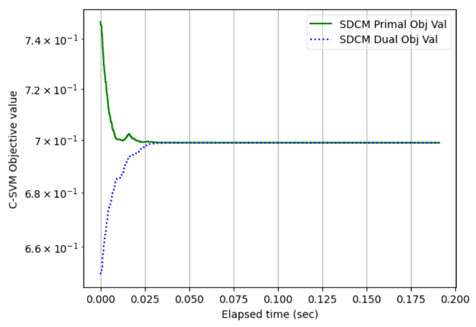


Figure 12: Primal and Dual Objective Curve (C = 0.001)

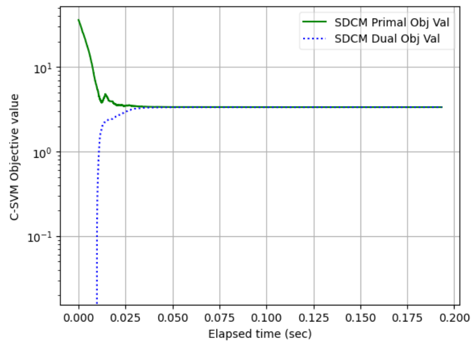


Figure 13: Primal and Dual Objective Curve (C = 0.01)

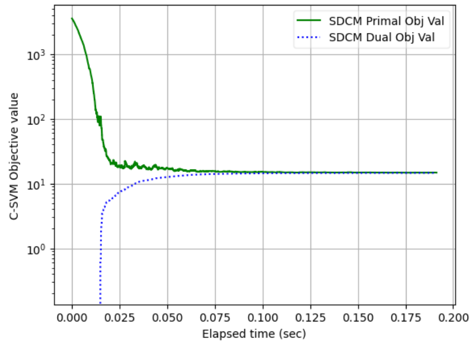


Figure 14: Primal and Dual Objective Curve (C = 0.1)

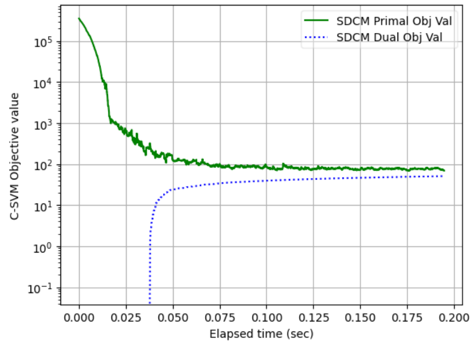


Figure 15: Primal and Dual Objective Curve (C = 1)

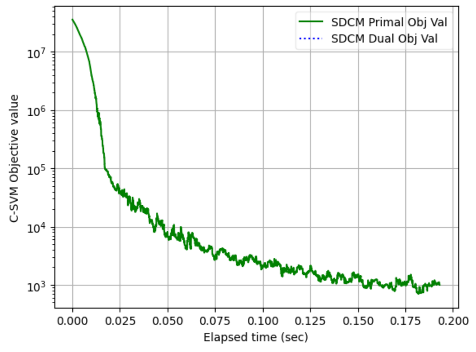


Figure 16: Primal and Dual Objective Curve (C = 10)

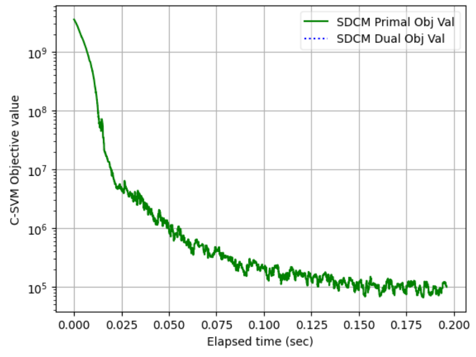


Figure 17: Primal and Dual Objective Curve (C = 100)

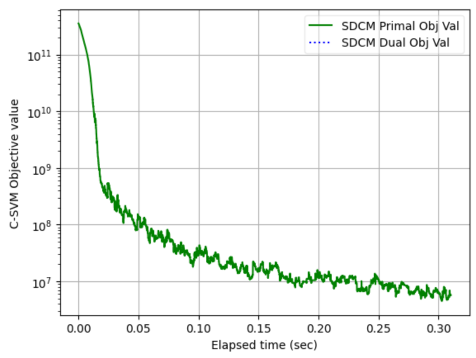


Figure 18: Primal and Dual Objective Curve (C = 1000)

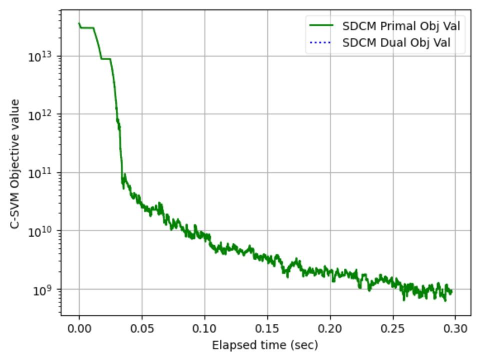


Figure 19: Primal and Dual Objective Curve (C = 10000)

Question 4: Experiment with at least two of the hyperparameters mentioned below and comment on their effect on test accuracy and primal-dual convergence curve. You must report the test accuracies and the convergence curves in your report after changing these hyperparameters. Please do these experiments after you have fixed Melbo's errors.

(c) Effect of horizon (the number of iterations) – note that this is passed as an argument to the function doSDCM. Currently it is set to 20,000. Does increasing it by a lot get a much better solution? Does decreasing it a lot make the solution much worse?

Answer:

Horizon	Test Accuracy (%)	Final Primal Value	Final Dual Value	Duality Gap
20	93.31	34.787467	-23.848933	58.636400
200	93.22	23.913502	-14.632440	38.545942
2000	93.18	3.547235	3.097908	0.449328
20000	95.06	3.353955	3.353895	0.000060
100000	95.05	3.353906	3.353899	0.000007
200000	95.06	3.353900	3.353899	0.000001

Table 2: Observed Test Accuracy, Final Primal and Dual Values for different number of iterations.

As we decrease the horizon (number of iterations), the test accuracy drops when compared to horizon = 20000, and the difference between the final primal and dual values (duality gap) increases. This indicates that the primal and dual values did not converge to the optimal solution. When the horizon is set to 2000, the duality gap reduces drastically compared to when the horizon is 200. As we increase the horizon from 20000 to 200,000 (very high value), the test accuracy remains almost the same, and the duality gap almost becomes 0.

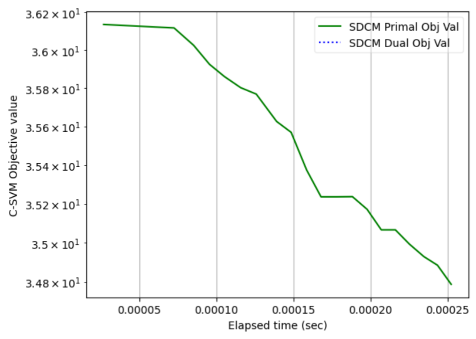


Figure 20: Primal and Dual Objective Curve (Horizon = 20)

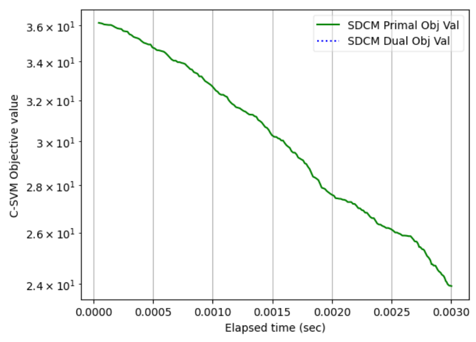


Figure 21: Primal and Dual Objective Curve (Horizon = 200)

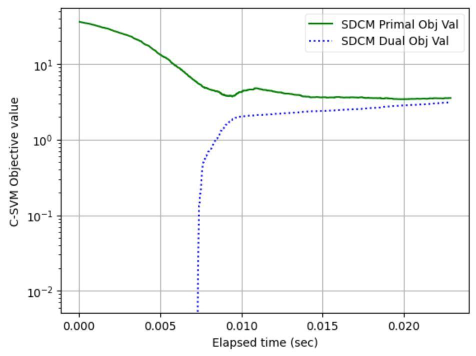


Figure 22: Primal and Dual Objective Curve (Horizon = 2000)

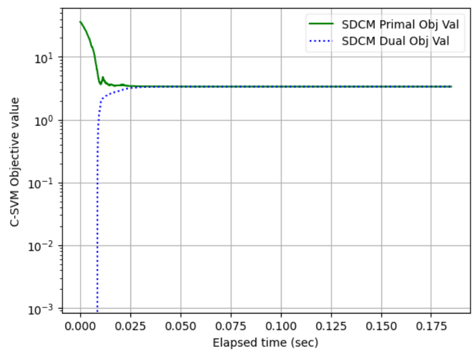


Figure 23: Primal and Dual Objective Curve (Horizon = 20000)

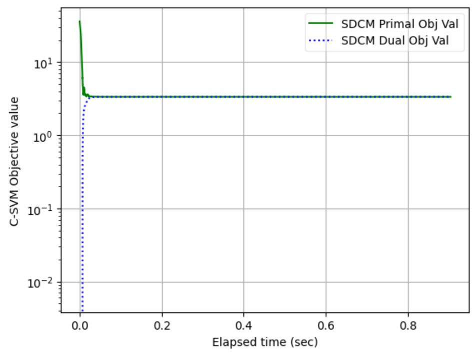


Figure 24: Primal and Dual Objective Curve (Horizon = 100000)

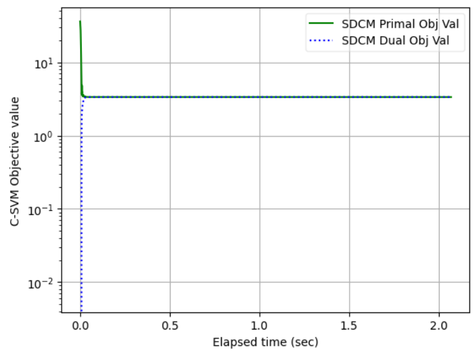


Figure 25: Primal and Dual Objective Curve (Horizon = 200000)

(e) Random chance – even if you keep all hyperparameters and settings fixed (say to their default values), does running the entire Jupyter notebook from first-to-last-cell again and again change the test accuracy? What could be the reason behind this?

Answer:

The test accuracy changes when running the entire Jupyter notebook from the first cell to the last cell again and again, even if all hyperparameters are kept fixed. This is due to sources of randomness in the code, and no fixed random seed is set to control them. As a result, the model behaves slightly different across runs.

There are two main sources of randomness:

1. **Train-Test Split:**

   ```
   train_test_split(data, train_size = 1000)
   ```

   The command above internally shuffles the dataset before splitting. Since the random state is not fixed, the training and testing sets are different each time.

2. **Random Coordinate Selection in SDCM:**

   ```
   np.random.permutation(d)
   ```

   In the coordinate descent implementation, the above function is used to generate a random permutation of coordinates. This permutation controls the order in which the coordinates are updated. Since the random seed is not fixed, every run produces a different permutation. This leads to a different optimization path and, consequently, the final weights differ slightly across runs, resulting in variations in accuracy.

References

[1] “Duality gap,” *Wikipedia*, Apr. 08, 2021. [Online]. Available: `https://en.wikipedia.org/wiki/Duality_gap`

[2] “CS771: Lecture Material,” *IIT Kanpur*, 2025. [Online]. Available: `https://www.cse.iitk.ac.in/users/purushot/courses/ml/2025-26-a/lectures.html`

[3] ML Conversations, “How do ML libraries like scikit-learn solve SVM problems? | Introduction to Machine Learning | CS771,” *YouTube*, Feb. 21, 2023. [Online]. Available: `https://www.youtube.com/watch?v=7xj4puUGJEc`

[4] ML Conversations, “To b or not to b | Introduction to Machine Learning | CS771,” *YouTube*, Mar. 04, 2023. [Online]. Available: `https://www.youtube.com/watch?v=QbSVRFTFt-8`