**Instructions**: Please complete and submit your work to the appropriate folder in LumiNUS. You may work in study groups, but each student must be responsible for their own submission.

Please submit all the following documents as a single zip file named StudentID-Name-H3.zip:

1. Completed Word file named as StudentID-Name-H3.docx (with all results)
2. Print preview of ipynb file named as StudentID-Name-H3.pdf (with results)
3. Working ipynb file named as StudentID-Name-H3.ipynb

1. Consider building an SVM classifier for the following two-class training data:

Positive class: { (-1, 3) (0, 2) (0, 1) (0, 0) }; Negative class: { (1, 5) (1, 6) (3, 3) }

1. Plot the training points. Use ‘+’ for positive class and ‘o’ for the negative class.
2. By inspection, draw a linear classifier that separates the data with maximum margin.
3. The linear SVM is parameterized by h(x) = (**w**^t)(x) + b. What are the parameters **w** and b for this problem?
4. Suppose you observe an additional set of points, all from the positive class.

Additional data points in positive class: { (−2, 0) (−2, 1) (−2, 3) (−1, 0) (−1, 1) }

What is the linear SVM (in terms of **w** and b) now?

**Ans 1:**

1. The plot for part (a) is shown below and the linear classifier for part (b) is shown below as well.

Diagram

Description automatically generated

1. The linear classifier is shown above.
2. The parameters w and b is shown below – if I follow the mathematical formula:

Text, letter

Description automatically generated

However, if I solve purely by inspection, the **w** will be the gradient of the drawn SVM line which is

and the value of **b** as **3.5** as b is the y-intercept, given that the drawn SVM line follows to that of an equation of a straight line. For the purposes of this problem (through inspection), I will just take **w** as and **b** as **3.5.**

1. Since the new data points from the positive class all lie further than the original “+” support vectors, they do **not** affect the previous linear SVM linear classifier as drawn in part (a). Hence, the values for the parameters **w** and b remains at and **3.5.**

Diagram

Description automatically generated

1. A picture containing clock

   Description automatically generatedConsider the dataset on the right. Consider using the SVM with soft margin classifier with parameter C.
   1. Draw the linear classifier when C is large.
   2. Draw the linear classifier when C is small.
   3. Which value of C yields the classifier most closely resembling the hard margin SVM solution?
   4. Using your two examples, explain how the C parameter helps with overfitting in SVMs.

**Ans 2:**

1. Linear Classifier When C Is **Large**:

A picture containing clock

Description automatically generated

1. Linear Classifier When C Is **Small**:

A picture containing clock

Description automatically generated

1. When C is large i.e., as mentioned in part (a), it closely resembles to the large margin SVM solution as it linearly separates the data into 2 distinct classes and prevents wrong misclassifications at all, so the penalty for misclassification is high.
2. In part (a), when the value of C is high, the classifier will do extremely well in the training data as there is a high penalty for misclassifications, so the SVM solution linearly separates the training data well into distinct classes. However, in part (b), when the value of C is low, there is a smaller penalty for misclassifications, hence, it is prone to anomalies in classification as some classes can be defined wrongly as shown in the misclassified red circle in part (b). The red circle in the bottom in part (b) is an outlier when C is small but not when C is high. These outlier data might make the linear separators go closer to the wrong class, and in the above case, it is the red circle being misclassified as a + class instead of a circle. As a result, you might see that the SVM classifiers with the C parameters in place might misclassify the unseen test data poorly just like how it did for the red circle as shown in part (b) and result in inaccuracies. Thus, this could result in overfitting as training data would be generalized or classified well but not very well for unseen test data where the C parameter variation is prone to misclassify the data. So when the C is large, the SVM separators is prone to overfitting but not too much when C is small. Hence, when the C value is lowered, we can prevent the classifier from overfitting the training data by allowing for some misclassifications in the training data, with a larger margin than when C is large, so more confidence in the model.
3. In this problem, we will look at the Breast Cancer Wisconsin (Diagnostic) Data Set available UCI Machine Learning Repository. Please use the wdbc.data dataset from:

https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29

* Compute the performance of the SVM algorithm on this dataset for predicting the whether the cancer is malignant or benign. Use a random train/test data split of 70%/30%. Repeat this process 20 times and compute the average performance.
* Please evaluate the following algorithms:
* SVM1: SVM with linear kernel
* SVM2: SVM with RBF kernel
* SVM3: Same as SVM2 but with regularization (soft margin), vary C and report your best results.
* Please compute the following metrics and fill in the table below.
* Training Accuracy and Test Accuracy
* Precision and Recall (which are important metrics that complement Accuracy)
* You can read about performance metrics at: <https://en.wikipedia.org/wiki/Confusion_matrix>
* SKLearn contains functions to compute these metrics:

<https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Accuracy | | Precision | Recall |
|  | Train | Test |  |  |
| SVM1 | **0.9673366834170851** | **0.9514619883040936** | **0.9479821811347918** | **0.921875** |
| SVM2 | **0.913316582914573** | **0.9225146198830411** | **0.9635462225815864** | **0.825** |
| SVM3  C = **18000** | **0.9701005025125626** | **0.9578947368421051** | **0.9601367362398697** | **0.9265625** |