Pattern Recognition, Neural Networks and Deep Learning:

Ensemble Methods

Rakan Zabian - 1741706 - MSc Data Science

22/04/2021

1 Question 1

Class 1: y=-1	Class 2: y=+1
(-2.1, 0.1)	(0.1, -1.3)
(-0.8, 1.4)	(-3.4, -1.9)
(-1.9, 3.1)	(-0.7, 1.6)
(-2.4, -1)	(-1.8, -0.5)
(-0.4, 2.6)	(-5.3, -4.1)
(-1.7, 1.2)	(-3.6, -4.8)
(0.4, 2.5)	(-1.5, -6.4)
(-3.5, -1)	(-1. , 0.2)
(1.5, 3.7)	(0.2, -2.2)
(-4.5, -0.9)	(-4.7, 1.4)

Table 1: Dataset of two classes.

This dataset of 20 samples per class was generated using 'sklearn.datasets.make_classification'. The parameter 'random_state' was equated to the student ID (1741706) to avoid any plagiarism/collusion issues. The parameter class_sep was equated to 0.5 to minimize the separation between the classes so that they are non-linearly separable.

2 Question 2

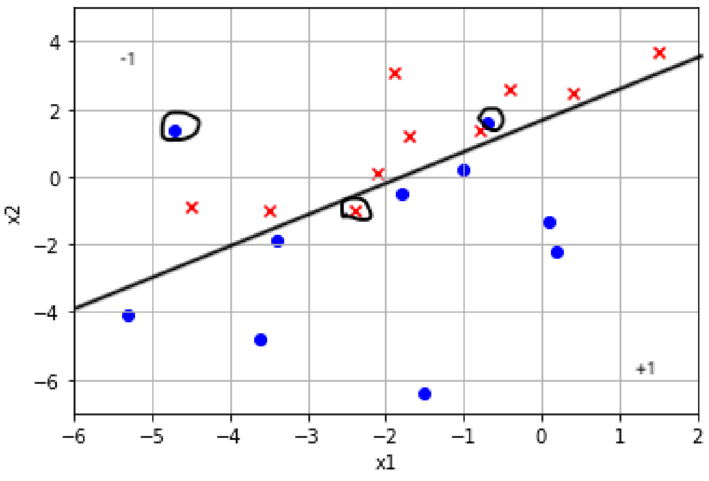


Figure 1: Separability

In figure 1, we observe that the two classes cannot be separated using a straight line. This is because, when we try to use a linear classifier, we will have missclassified points (circled in black). Hence, the dataset is non-linearly separable.

3 Question 3

3.1 Step 1: Dataset $\mathcal{D}$

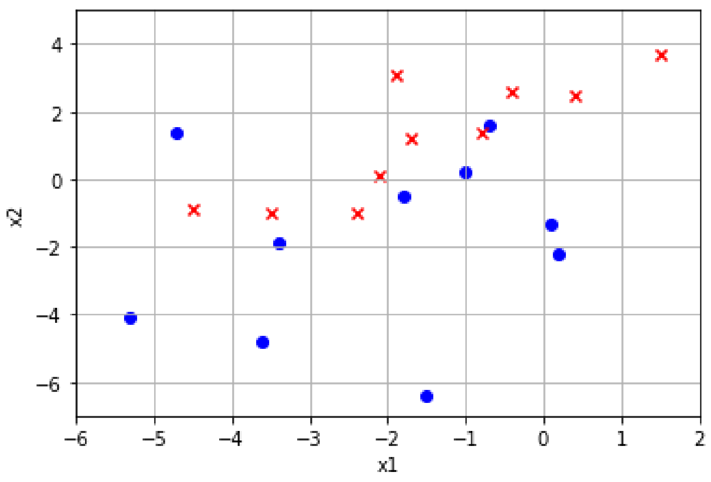
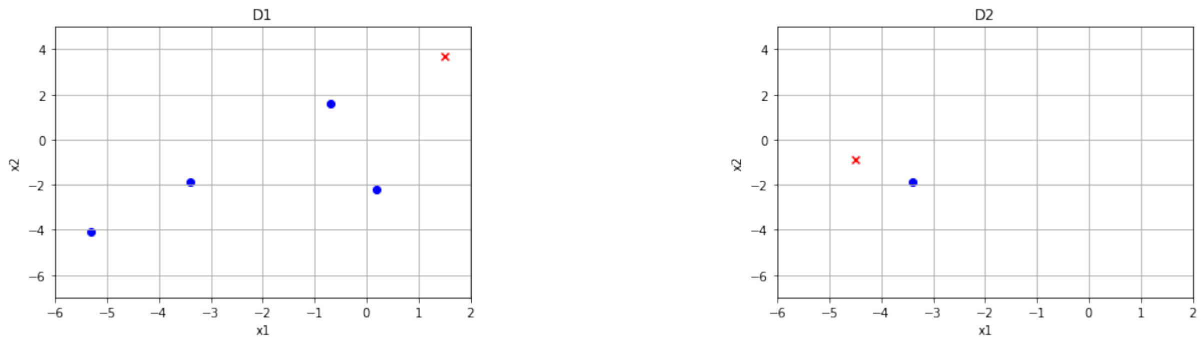
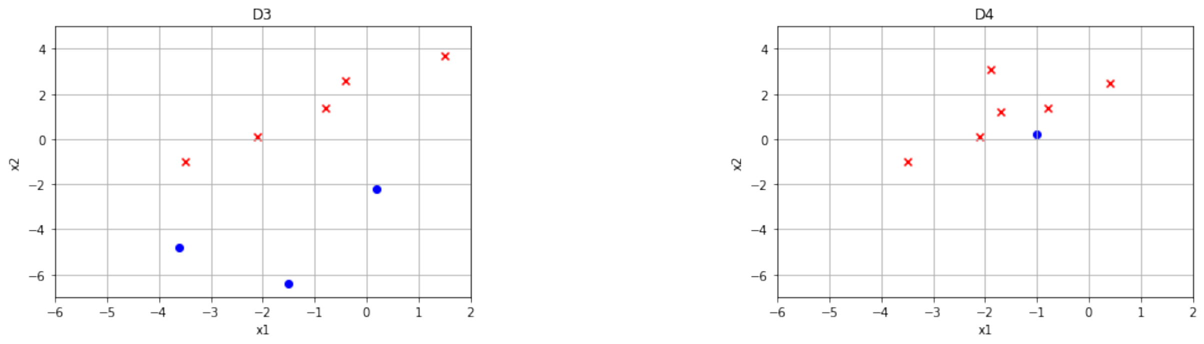


Figure 2: Dataset $\mathcal{D}$

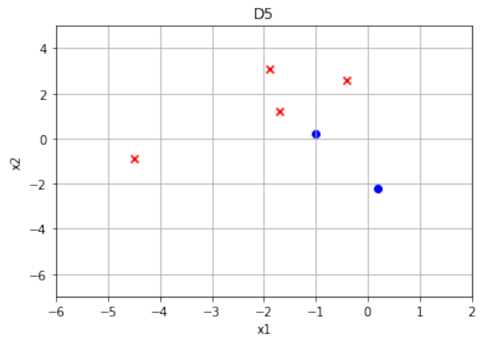
3.2 Step 2: 5 Sub-Datasets with replacement



(a)



(b)



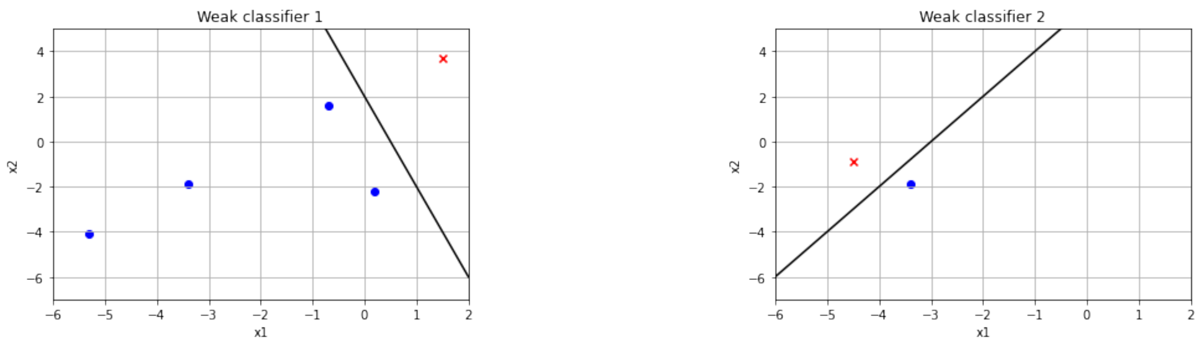
(c)

Figure 3: Datasets $\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3, \mathcal{D}_4, \mathcal{D}_5$

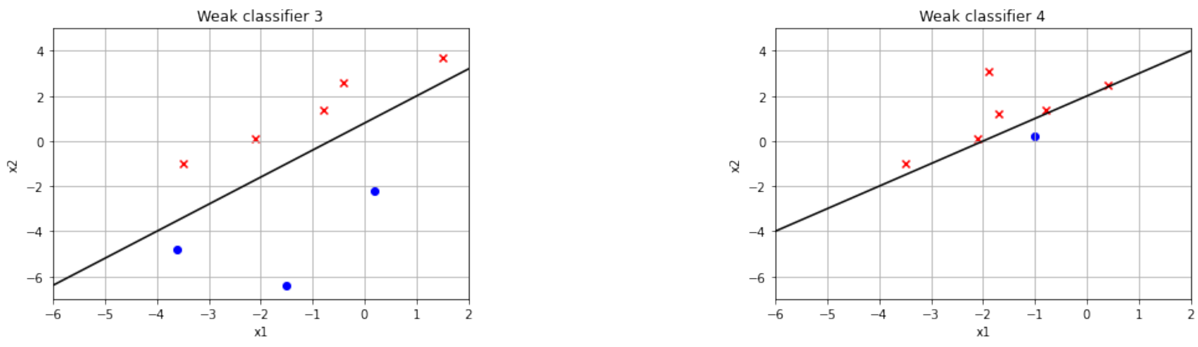
The above distributions were generated randomly by drawing $n' < n$ samples from $\mathcal{D}$ with replacement. Some samples appear more than once, while others do not appear at all.

3.3 Step 3: Weak classifiers

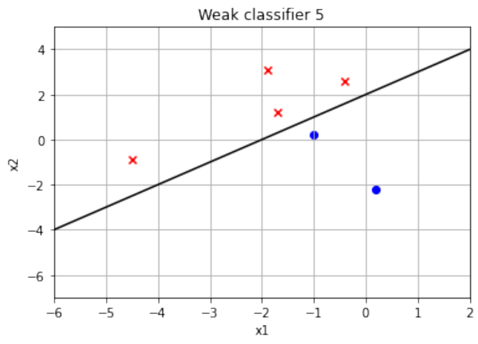
3.3.1 Obtaining weak classifiers



(a)



(b)



(c)

Figure 4: Weak classifiers 1,2,3,4,5

The above images display the designs of the 5 weak classifiers. All 5 classifiers were obtained by observation. The Batch Perceptron Learning algorithm, explored in the next section, is used to justify the observations. The solutions weren't changed by the algorithm as all initial solutions (obtained by observation) classified the datasets with 100% accuracy.

Below, we will explain the design of the hyperplane for the first weak classifier. This design will show how the hyperparamaters used to classify all samples in $\mathcal{D}_1$ are obtained. Once the hyperparameters are justified by the algorithm, the hyperplane is used as a weak classifier.

Sample	Class: (0,1)
(1.5, 3.7)	-1
(-3.4, -1.9)	+1
(-5.3, -4.1)	+1
(-0.7, 1.6)	+1
(0.2, -2.2)	+1

Table 2: Dataset $\mathcal{D}_1$

The above table displays all samples in $\mathcal{D}_1$. To classify the samples, we observed that a line intersecting points (0,2) & (1,-2) can linearly separate class 1 (red x, y = -1) from class 2 (blue o, y = +1). So, we begin by finding the equation of the line using the two points.

Find slope m:

$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{2-(-2)}{0-1} = -4$$

Find equation of line:

$$y - y_1 = m(x - x_1)$$

$$y - 2 = -4(x - 0)$$

$$y = -4x + 2$$

So, the hyperparameters of the above line are: $\begin{bmatrix} 2 \\ -4 \\ -1 \end{bmatrix}$

Now, we find the hyperplane using the Batch Perceptron Learning algorithm by hand.

Assume initial values of a = $(w_0, \boldsymbol{w})^t = (2, -4, -1)^t$, and use a learning rate of 1.

Using Augmented notation, dataset is:

Sample	$\boldsymbol{y}$	Class: y=(-1,+1)
(1.5, 3.7)	(1, 1.5, 3.7)	-1
(-3.4, -1.9)	(1, -3.4, -1.9)	+1
(-5.3, -4.1)	(1, -5.3, -4.1)	+1
(-0.7, 1.6)	(1, -0.7, 1.6)	+1
(0.2, -2.2)	(1, 0.2, -2.2)	+1

Table 3: Dataset $\mathcal{D}_1$ Augmentation

Epoch 1: initial $\boldsymbol{a} = (2, -4, -1)^t$

$\boldsymbol{y}$	g(x) = $\boldsymbol{a}^t\boldsymbol{y}$	Misclassified?(i.e g(x) ≥ 0 (class 1) or $g(x) \leq 0$ (class 2)
(1, 1.5, 3.7)	$(2 \times 1) + (-4 \times 1.5) + (-1 \times 3.7) = -7.7$	No
(1,-3.4, -1.9)	$(2 \times 1) + (-4 \times -3.4) + (-1 \times -1.9) = 17.5$	No
(1,-5.3, -4.1)	$(2 \times 1) + (-4 \times -5.3) + (-1 \times -4.1) = 27.3$	No
(1,-0.7, 1.6)	$(2 \times 1) + (-4 \times -0.7) + (-1 \times 1.6)) = 3.2$	No
(1,0.2, -2.2)	$(2 \times 1) + (-4 \times 0.2) + (-1 \times -2.2) = 3.4$	No

Table 4: Batch Perceptron Learning

Learning has converged, so required parameters are a = (2,-4,-1) .

Using the above process, we calculated the hyperparameters for each of the 5 weak classifiers.

Weak classifier	Hyperparameters
1	(2, -4, -1)
2	(6, 2, -1)
3	(0.8, 1.2, -1)
4	(2, 1, -1)
5	(2, 1, -1)

Table 5: Weak classifiers - Hyperparameters

3.3.2 Design of algorithm used to support observation

To learn the weak classifiers for each dataset, we opted to use the Batch Perceptron Learning Algorithm. This is because the algorithm will allow us to begin with an arbitrary solution, which we will find by observation. If the solution correctly classifies all samples, the algorithm will output the initial solution that we input by observation. The inputs & outputs of the algorithm are explained below:

Input:

1. Set value of hyper-parameter (η)
2. Using augmented notation, input feature vector
3. Input class labels
4. Initialise $\boldsymbol{a}$ to arbitrary solution

Output:

1. Output linear classifier hyperparameters

The algorithm pseudo-code is as follows:

Algorithm 1 Batch Perceptron Learning

```
Input: 1. (learning_rate) 2. (dataset) 3. (class_labels) 4. ($\boldsymbol{a}$_initial)
Output: 1. ($\boldsymbol{a}$_final)
while # of missclassified samples >0 do
    Compute gradient vector at $\boldsymbol{a}$: $\nabla J_p(\boldsymbol{a}) = \sum_{y \in \mathcal{X}} (-\boldsymbol{y})$
    Move solution in direction of steepest descent: $\boldsymbol{a} \leftarrow \boldsymbol{a} + \eta \sum_{y \in \mathcal{X}} \boldsymbol{y}$
end
return $\boldsymbol{a}$_final
```

In words:

The new weight vector is obtained by summing the feature vectors associated with all misclassified samples, and adding some multiple of this to the current weight vector.

3.4 Overall classifier

3.4.1 3 weak classifiers

Data	Weak classifier 1	Weak classifier 2	Weak classifier 3	Overall classifier
(0.1, -1.3)	+1	+1	+1	+1
(-2.1, 0.1)	+1	+1	-1	+1
(-3.4, -1.9)	+1	+1	-1	+1
(-0.7, 1.6)	+1	+1	-1	+1
(-0.8, 1.4)	+1	+1	-1	+1
(-1.8, -0.5)	+1	+1	-1	+1
(-1.9, 3.1)	+1	-1	-1	-1
(-2.4, -1.0)	+1	+1	-1	+1
(-5.3, -4.1)	+1	-1	-1	-1
(-0.4, 2.6)	+1	+1	-1	+1
(-3.6, -4.8)	+1	+1	+1	+1
(-1.5, -6.4)	+1	+1	+1	+1
(-1.0, 0.2)	+1	+1	-1	+1
(-1.7, 1.2)	+1	+1	-1	+1
(0.4, 2.5)	-1	+1	-1	-1
(0.2, -2.2)	+1	+1	+1	+1
(-4.7, 1.4)	+1	-1	-1	-1
(-3.5, -1.0)	+1	0	-1	Undefined
(1.5, 3.7)	-1	+1	-1	-1
(-4.5, -0.9)	+1	-1	-1	-1
Accuracy (%)	0.6	0.5	0.7	0.6

Table 6: Bagging classifier using 3 weak classifiers

3.4.2 4 weak classifiers

Data	Weak classifier 1	Weak classifier 2	Weak classifier 3	Weak classifier 4	Overall classifier
(0.1, -1.3)	+1	+1	+1	+1	+1
(-2.1, 0.1)	+1	+1	-1	-1	Undefined
(-3.4, -1.9)	+1	+1	-1	+1	+1
(-0.7, 1.6)	+1	+1	-1	-1	Undefined
(-0.8, 1.4)	+1	+1	-1	-1	Undefined
(-1.8, -0.5)	+1	+1	-1	+1	+1
(-1.9, 3.1)	+1	-1	-1	-1	-1
(-2.4, -1.0)	+1	+1	-1	+1	+1
(-5.3, -4.1)	+1	-1	-1	+1	Undefined
(-0.4, 2.6)	+1	+1	-1	-1	Undefined
(-3.6, -4.8)	+1	+1	+1	+1	+1
(-1.5, -6.4)	+1	+1	+1	+1	+1
(-1.0, 0.2)	+1	+1	-1	+1	+1
(-1.7, 1.2)	+1	+1	-1	-1	Undefined
(0.4, 2.5)	-1	+1	-1	-1	-1
(0.2, -2.2)	+1	+1	+1	+1	+1
(-4.7, 1.4)	+1	-1	-1	-1	-1
(-3.5, -1.0)	+1	0	-1	-1	-1
(1.5, 3.7)	-1	+1	-1	-1	-1
(-4.5, -0.9)	+1	-1	-1	-1	-1
Accuracy (%)	0.6	0.5	0.7	0.6	0.6

Table 7: Bagging classifier using 4 weak classifiers

3.4.3 5 weak classifiers

Data	Weak classifier 1	Weak classifier 2	Weak classifier 3	Weak classifier 4	Weak classifier 5	Overall classifier
(0.1, -1.3)	+1	+1	+1	+1	+1	+1
(-2.1, 0.1)	+1	+1	-1	-1	-1	-1
(-3.4, -1.9)	+1	+1	-1	+1	+1	+1
(-0.7, 1.6)	+1	+1	-1	-1	-1	-1
(-0.8, 1.4)	+1	+1	-1	-1	-1	-1
(-1.8, -0.5)	+1	+1	-1	+1	+1	+1
(-1.9, 3.1)	+1	-1	-1	-1	-1	-1
(-2.4, -1.0)	+1	+1	-1	+1	+1	+1
(-5.3, -4.1)	+1	-1	-1	+1	+1	+1
(-0.4, 2.6)	+1	+1	-1	-1	-1	-1
(-3.6, -4.8)	+1	+1	+1	+1	+1	+1
(-1.5, -6.4)	+1	+1	+1	+1	+1	+1
(-1.0, 0.2)	+1	+1	-1	+1	+1	+1
(-1.7, 1.2)	+1	+1	-1	-1	-1	-1
(0.4, 2.5)	-1	+1	-1	-1	-1	-1
(0.2, -2.2)	+1	+1	+1	+1	+1	+1
(-4.7, 1.4)	+1	-1	-1	-1	-1	-1
(-3.5, -1.0)	+1	0	-1	-1	-1	-1
(1.5, 3.7)	-1	+1	-1	-1	-1	-1
(-4.5, -0.9)	+1	-1	-1	-1	-1	-1
Accuracy (%)	0.6	0.5	0.7	0.6	0.85	0.85

Table 8: Bagging classifier using 5 weak classifiers

3.5 Sample 2

Below, we will explain how to determine the class (for each weak classifier and overall classifier) using one test sample (-2.1,0.1).

Classifier	g(x) = $\boldsymbol{a}^t\boldsymbol{y}$	Class (-1,+1)
1	$(2\ \times 1) + (-4 \times -2.1) + (-1 \times 0.1) = 10.3$	+1
2	$(6\ \times 1) + (2 \times -2.1) + (-1 \times 0.1) = 1.7$	+1
3	$(0.8\ \times 1) + (1.2 \times -2.1) + (-1 \times 0.1) = -1.82$	-1
4	$(2\ \times 1) + (1 \times -2.1) + (-1 \times 0.1) = -0.2$	-1
5	$(2\ \times 1) + (1 \times -2.1) + (-1 \times 0.1) = -0.2$	-1

Table 9: Sample 2 classification

By majority voting, sample 2 (-2.1,0.1) is classified as y = -1.

3.6 Discussion

When observing the performance of each of the Bagging classifiers, we see that it begins at a low accuracy (0.6) and ends at a high accuracy (0.85).The performance surprisingly remained constant when integrating a 4th weak classifier (0.6 accuracy maintained). This is because when using an even number of weak classifiers, ties may occur, hence the 'undefined' labels. These samples were considered misclasified. However, after adding the 5th weak classifier, ties could no longer occur, and the majority vote lead us to a strong 85% accuracy. This shows us that increasing the number of weak classifiers can help increase our chances of improving the classification performance of our overall classifier.

4 Question 4

4.1 Dataset $\mathcal{D}$

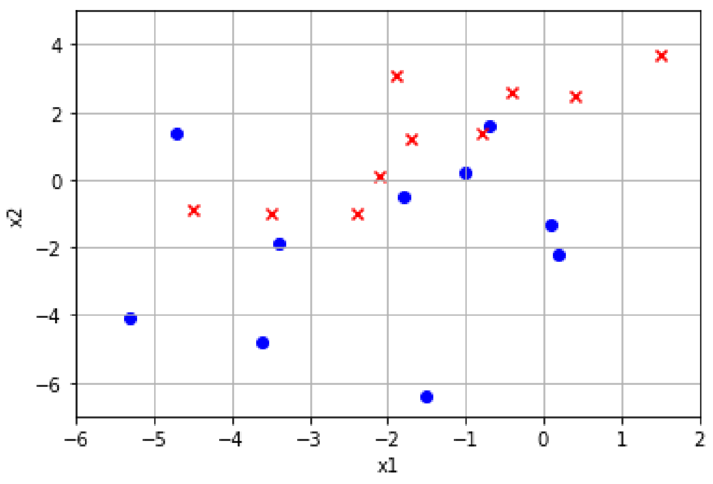


Figure 5: Dataset $\mathcal{D}$

4.2 Step 1: Dataset $\mathcal{D}_1$

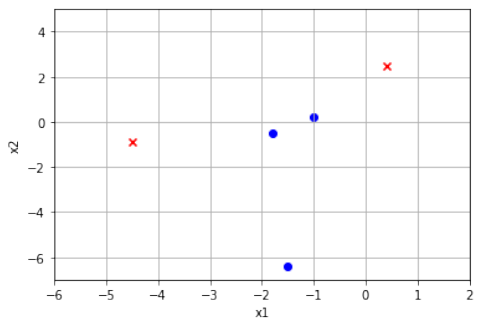


Figure 6: Dataset $\mathcal{D}_1$

The above distribution was generated randomly by drawing $n' < n$ samples from $\mathcal{D}$. The samples were then removed from $\mathcal{D}$ so that the following sub datasets are generated without replacement. Samples that appear in this dataset will not appear again.

4.2.1 Weak classifier 1

To learn the weak classifier for each dataset, the same design used in Q3 was implemented.

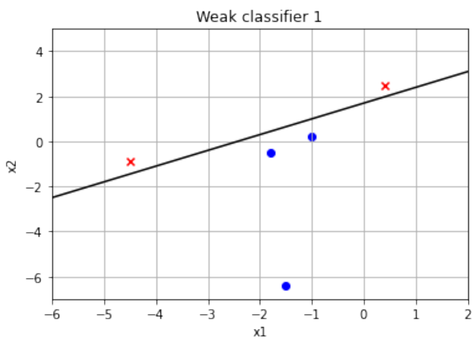


Figure 7: Weak classifier 1

Weak classifier	Hyperparameters	Accuracy
1	(1.7, 0.7, -1)	100%

Table 10: Weak classifier 1 performance on $\mathcal{D}_1$

4.3 Step 2: Dataset $\mathcal{D}_2$

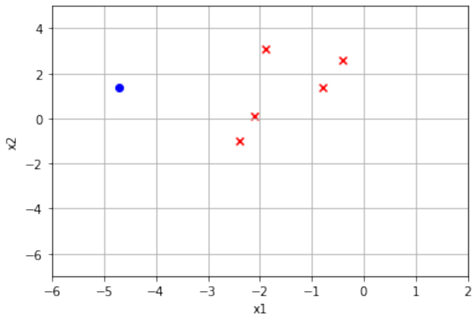


Figure 8: Dataset $\mathcal{D}_2$

The above distribution was generated randomly by drawing $n' < n$ samples from the modified version of $\mathcal{D}$ (excluding samples in $\mathcal{D}_1$). The samples were then removed from $\mathcal{D}$ so that the following sub dataset is generated without replacement. Samples that appear in this dataset will not appear again. Multiple seeds were tried when generating the dataset, until one was output such that weak classifier 1 correctly classified roughly 50% of its samples.

4.3.1 Weak classifier 2

To learn the weak classifier for each dataset, the same design used in Q3 was implemented.

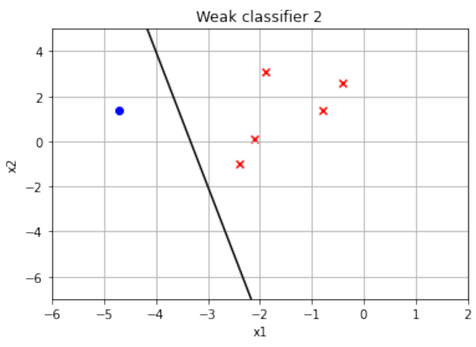


Figure 9: Weak classifier 2

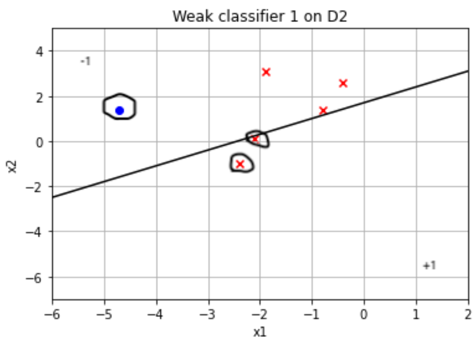


Figure 10: Weak classifier 1 on $\mathcal{D}_2$

Weak classifier	Hyperparameters	Accuracy
1	(1.7, 0.7, -1)	50%
2	(-20, -6, -1)	100%

Table 11: Weak classifiers 1&2 performance on $\mathcal{D}_2$

4.4 Step 3: Dataset $\mathcal{D}_3$

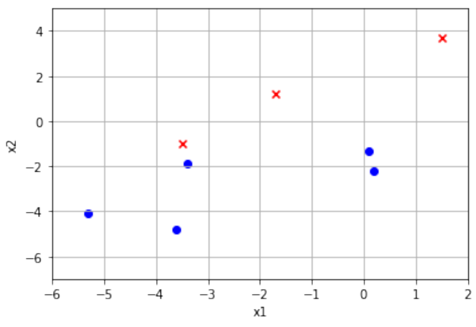


Figure 11: Dataset $\mathcal{D}_3$

The above distribution was generated randomly by drawing $n' < n$ samples from the modified version of $\mathcal{D}$ (excluding samples in $\mathcal{D}_1$& $\mathcal{D}_2$). Multiple seeds were tried when generating the dataset, until one was output such that weak classifiers 1&2 disagreed frequently (25% of all samples).

4.4.1 Weak classifier 3

To learn the weak classifier for each dataset, the same design used in Q3 was implemented.

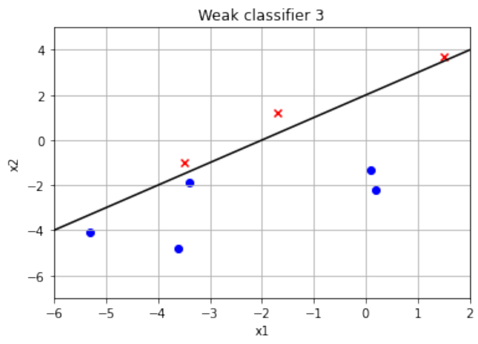


Figure 12: Weak classifier 3

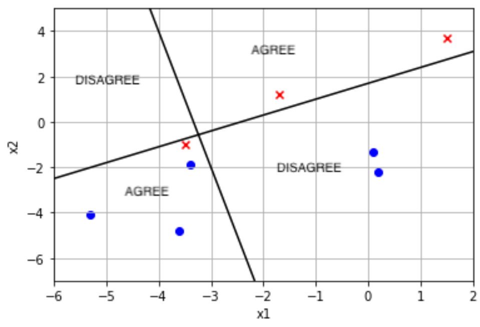


Figure 13: Weak classifiers 1,2 on $\mathcal{D}_3$ - disagreement regions

Weak classifier	Hyperparameters	Accuracy
3	(2, 1, -1)	100%

Table 12: Weak classifier 3 performance on $\mathcal{D}_3$

4.5 Overall classifier

Data	Weak classifier 1	Weak classifier 2	Weak classifier 3	Overall classifier
(0.1, -1.3)	+1	-1	+1	+1
(-2.1, 0.1)	+1	-1	-1	-1
(-3.4, -1.9)	+1	+1	+1	+1
(-0.7, 1.6)	-1	-1	-1	-1
(-0.8, 1.4)	-1	-1	-1	-1
(-1.8, -0.5)	+1	-1	+1	+1
(-1.9, 3.1)	-1	-1	-1	-1
(-2.4, -1.0)	+1	-1	+1	+1
(-5.3, -4.1)	+1	+1	+1	+1
(-0.4, 2.6)	-1	-1	-1	-1
(-3.6, -4.8)	+1	+1	+1	+1
(-1.5, -6.4)	+1	-1	+1	+1
(-1.0, 0.2)	+1	-1	+1	+1
(-1.7, 1.2)	-1	-1	-1	-1
(0.4, 2.5)	-1	-1	-1	-1
(0.2, -2.2)	+1	-1	+1	+1
(-4.7, 1.4)	-1	+1	-1	-1
(-3.5, -1.0)	+1	+1	-1	+1
(1.5, 3.7)	-1	-1	-1	-1
(-4.5, -0.9)	-1	+1	-1	-1
Accuracy (%)	0.75	0.6	0.85	0.8

Table 13: Boosting classifier using 3 weak classifiers

4.6 Sample 2

Below, we will explain how to determine the class (for each weak classifier and overall classifier) using one test sample (-2.1,0.1).

Classifier	g(x) = $\boldsymbol{a}^t\boldsymbol{y}$	Class (-1,+1)
1	$(1.7 \times 1) + (0.7 \times -2.1) + (-1 \times 0.1) = 0.13$	+1
2	$(-20\times 1) + (-6 \times -2.1) + (-1 \times 0.1) = -7.5$	-1
3	$(2 \times 1) + (1 \times -2.1) + (-1 \times 0.1) = -0.2$	-1

Table 14: Sample 2 classification

By majority voting, sample 2 (-2.1,0.1) is classified as y = -1.

4.7 Discussion

The overall classifier performs significantly better than both the first and second weak classifiers, but slightly worse than the third. This is because the first two sub datasets, primarily the second, observed some samples that could be considered outliers. Hence, the weak classifiers may have been slightly overfitted (weak classifier 2). However, using 3 weak classifiers helped boost the performance significantly. This is because the third dataset observed samples that weren't in the first two datasets, enabling the overall classifier to build a model fitted well to the original dataset.

Generating the datasets using the boosting method (without replacement) was essential to obtaining the significantly improved performance of the boosting classifier when compared to the bagging classifier (3 weak classifiers). This is because, the boosting datasets targeted different parts of the original dataset, hence, the overall model was well fitted to the original dataset, unlike the bagging datasets, were they may have targeted similar parts of the original dataset. In the case of bagging, a higher number of weak classifiers was needed so that we could attain a good performance.