CSCE 5380.401 – Data Mining

Homework No. 3 - ANN, SVM & Imbalanced Classes

Uday Bhaskar Valapadasu - 11696364 | Sweatha Subramanian - 11655058 |Sapthagiri Naik Bhukya - 11699072

1Ans:

The AND function for two Boolean variables is defined as:

x_1	x_2	AND
0	0	0
0	1	0
1	0	0
1	1	1

To represent this using a perceptron, we need to determine appropriate weights (w_1 and w_2) and a bias (b) that will correctly classify all input combinations. Let's set up the perceptron:

1. Inputs: x_1 and x_2 (each can be 0 or 1)
2. Weights: w_1 and w_2
3. Bias: b
4. Activation function: Step function

The perceptron computes: $y = f(w_1x_1 + w_2x_2 + b)$

Where f is the step function:
$f(z) = 1$ if $z \geq 0$
$f(z) = 0$ if $z < 0$

For the AND function, we can use the following parameters:
$w_1 = 1$, $w_2 = 1$, $b = -1.5$

Let's verify this works for all input combinations:

1. $x_1 = 0$, $x_2 = 0$:
   $z = 1(0) + 1(0) + (-1.5) = -1.5$
   $f(-1.5) = 0$, because $-1.5 < 0$

2. $x_1 = 0$, $x_2 = 1$:
   $z = 1(0) + 1(1) + (-1.5) = -0.5$
   $f(-0.5) = 0$, because $-0.5 < 0$

3. $x_1 = 1$, $x_2 = 0$:
   $z = 1(1) + 1(0) + (-1.5) = -0.5$
   $f(-0.5) = 0$, because $-0.5 < 0$

4. $x_1 = 1$, $x_2 = 1$:
   $z = 1(1) + 1(1) + (-1.5) = 0.5$
   $f(0.5) = 1$, because $0.5 \geq 0$

As we can observe, this perceptron correctly represents the AND function for all input combinations. The key points are:

1. Both inputs need to be 1 to overcome the bias of -1.5.
2. The weights are equal (1 and 1), reflecting that both inputs are equally important in an AND operation.
3. The bias (-1.5) ensures that the output is 1 only when both inputs are 1.

This example demonstrates how a simple perceptron can learn a linear decision boundary that perfectly separates the positive class (1,1) from the negative classes (0,0), (0,1), and (1,0) for the AND function.

2Ans:

(a) A AND B AND C

This function is **linearly separable.** Because of the following reasons:

- The AND function is linearly separable, as demonstrated in the perceptron example for two variables.
- We can extend this to three variables by using a similar approach.
- The decision boundary would be a plane in 3D space that separates the single positive case (1,1,1) from all other combinations.
- We can represent this with a perceptron using weights $w_1 = w_2 = w_3 = 1$ and a bias $b = -2.5$.
- The function would be: $f(w_1A + w_2B + w_3C + b)$ where f is the step function.
- This will only output 1 when A=B=C=1, which is the definition of A AND B AND C.

(b) NOT A AND B

This function is **linearly separable.** Because of the following reasons:

- This is a two-variable function that can be represented by a straight line in 2D space.
- We can use a perceptron with weights $w_1 = -1$ (for NOT A), $w_2 = 1$ (for B), and a bias $b = 0$.
- The function would be: f(-A + B + 0) where f is the step function.
- This will output 1 only when A=0 and B=1, which correctly represents NOT A AND B.

(c) (A OR B) AND (A OR C)

This function is **linearly separable.** Because of the following reasons:

- Although this function seems more complex, it can still be represented by a plane in 3D space.
- We can break it down: it's true when either A is true, or when both B and C are true.
- A perceptron can represent this with weights $w_1 = 2$ (for A), $w_2 = 1$ (for B), $w_3 = 1$ (for C), and a bias $b = -1.5$.
- The function would be: f(2A + B + C - 1.5) where f is the step function.
- This will output 1 in the following cases: (1,0,0), (1,0,1), (1,1,0), (1,1,1), (0,1,1), which correctly represents (A OR B) AND (A OR C).

Key Points:
1. Linear separability means that a single linear decision boundary (line in 2D, plane in 3D, hyperplane in higher dimensions) can separate the true cases from the false cases.
2. The perceptron model and linear SVMs can only represent linearly separable functions.
3. For Boolean functions, if we can find weights and a bias that correctly classify all input combinations, the function is linearly separable.
4. The complexity of the function doesn't always determine its linear separability. Even some complex-looking functions (like the last one) can be linearly separable.

In this case, all three functions are linearly separable, demonstrating that many common Boolean functions can be represented by simple perceptron models or linear SVMs.

3Ans:

Part (a): A data set with 4 continuous-valued features x_1, x_2, x_3, and x_4

2D:
In 2D, we can't directly represent all four features. However, we can transform the problem into 2D by using the products x_1x_2 and x_3x_4 as our two dimensions:

- Dimension 1: $\Phi_1(x) = x_1x_2$
- Dimension 2: $\Phi_2(x) = x_3x_4$

In this 2D space, the problem is linearly separable. The decision boundary is the line $x_1x_2 = x_3x_4$.

Linear classifier: $f(x) = x_1x_2 - x_3x_4$

3D:
In 3D, we still don't need all four original dimensions. We can use the same two dimensions as in 2D, and add a constant third dimension:

- Dimension 1: $\Phi_1(x) = x_1x_2$
- Dimension 2: $\Phi_2(x) = x_3x_4$
- Dimension 3: $\Phi_3(x) = 1$ (constant)

The problem remains linearly separable in 3D, with the decision boundary being a plane instead of a line.

Linear classifier: $f(x) = x_1x_2 - x_3x_4 + 0$ (The last term is $0 * \Phi_3(x)$, which doesn't affect the classification)

Part (b): A data set with 2 binary features, x_1 and x_2, similar to XOR using -1 instead of 0

2D:
In the original 2D space (x_1, x_2), the problem is not linearly separable. The four points are:

- (1, 1) → y = -1
- (1, -1) → y = 1
- (-1, 1) → y = 1
- (-1, -1) → y = -1

No straight line can separate the positive and negative cases in this 2D space.

3D:
By adding a third dimension $\Phi_3(x) = x_1x_2$, we transform the problem into a 3D space where it becomes linearly separable:

- Dimension 1: $\Phi_1(x) = x_1$
- Dimension 2: $\Phi_2(x) = x_2$
- Dimension 3: $\Phi_3(x) = x_1x_2$

In this 3D space, the points become:

- (1, 1, 1) → y = -1
- (1, -1, -1) → y = 1
- (-1, 1, -1) → y = 1
- (-1, -1, 1) → y = -1

Linear classifier: $f(x) = x_1 + x_2 - 2x_1x_2$

This creates a plane in 3D space that perfectly separates the positive and negative cases.

In summary:
- Part (a) is linearly separable in both 2D and 3D after appropriate feature transformation.
- Part (b) is not linearly separable in 2D, but becomes linearly separable in 3D with the addition of an interaction term as the third dimension.

4Ans: XOR Problem Solution Using SVM

1. Problem Statement

The XOR (exclusive OR) problem is a classic example of a non-linearly separable classification task.

Given points:
- (0,0) → Negative class
- (1,0) → Positive class
- (0,1) → Positive class
- (1,1) → Negative class

Goal: Find a decision boundary that correctly classifies these points.

2. Approach

We'll use SVM with a custom kernel to transform the problem into a linearly separable one in a higher-dimensional space.

3. Feature Transformation

Transform function $\varphi(x1, x2) = (1, \sqrt{2}x1, \sqrt{2}x2, \sqrt{2}x1x2, x1^2, x2^2)$

Transformed points:
1. $\varphi(0,0) = (1, 0, 0, 0, 0, 0)$ → Negative
2. $\varphi(1,0) = (1, \sqrt{2}, 0, 0, 1, 0)$ → Positive
3. $\varphi(0,1) = (1, 0, \sqrt{2}, 0, 0, 1)$ → Positive
4. $\varphi(1,1) = (1, \sqrt{2}, \sqrt{2}, \sqrt{2}, 1, 1)$ → Negative

4. SVM in Transformed Space

In this 6D space, we seek a hyperplane:
$w0 + w1(1) + w2(\sqrt{2}x1) + w3(\sqrt{2}x2) + w4(\sqrt{2}x1x2) + w5(x1^2) + w6(x2^2) = 0$

5. Key Observation

The term $\sqrt{2}x1x2$ (4th component) perfectly separates the classes:
- For positive points: $\sqrt{2}x1x2 = 0$
- For negative points: $\sqrt{2}x1x2 \neq 0$

6. Optimal Decision Boundary

Based on this observation, the optimal decision boundary is:

$x1x2 = 0$

This is equivalent to setting $w4 = 1$ and all other $wi = 0$ in our hyperplane equation.

7. Verification

Let's verify this boundary for each point:

1. (0,0): $0 * 0 = 0$ → On boundary (classified as negative)
2. (1,0): $1 * 0 = 0$ → On boundary (classified as positive)
3. (0,1): $0 * 1 = 0$ → On boundary (classified as positive)
4. (1,1): $1 * 1 \neq 0$ → Not on boundary (classified as negative)

8. Interpretation

In the original 2D space:
- The boundary $x1x2 = 0$ represents two intersecting lines: $x1 = 0$ and $x2 = 0$
- This creates four quadrants that correctly separate the XOR classes

9. Significance

1. Non-linear to Linear: We transformed a non-linearly separable problem in 2D to a linearly separable one in 6D.
2. Kernel Trick: Although we explicitly showed the transformation, in practice, SVM would use the kernel trick to implicitly work in this higher-dimensional space.
3. Simplicity: The final decision boundary ($x1x2 = 0$) is remarkably simple, capturing the essence of XOR.

10. Conclusion

This solution demonstrates the power of SVM with appropriate kernels to solve complex, non-linear classification problems by implicitly working in higher-dimensional spaces where linear separation becomes possible.

5Ans:

Step 1: Organize the Data

True Labels:

- True class: [0, 0, 1, 0, 1, 0, 0, 1, 1, 1]

Probabilities for Classifiers:

- **P(y=+|C1):** [0.15, 0.20, 0.25, 0.37, 0.41, 0.55, 0.65, 0.80, 0.92, 0.99]
- **P(y=+|C2):** [0.33, 0.22, 0.10, 0.41, 0.68, 0.59, 0.72, 0.75, 0.64, 0.95]

Step 2: Calculate TPR and FPR

For each threshold, we calculate the True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN). Then, we compute TPR and FPR as follows:

- TPR=TP/(TP+FN)
- FPR=FP/(FP+TN)

Classifier C1:

Sorted Probabilities and Corresponding True Labels:

Threshold	True Label
0.15	0
0.20	0
0.25	1
0.37	0
0.41	1
0.55	0
0.65	0
0.80	1
0.92	1
0.99	1

Calculation Table for C1:

Threshold	TP	FP	TN	FN	TPR	FPR
0.15	0	1	4	5	0/5	1/5
0.20	0	2	3	5	0/5	2/5
0.25	1	2	3	4	1/5	2/5
0.37	1	3	2	4	1/5	3/5
0.41	2	3	2	3	2/5	3/5
0.55	2	4	1	3	2/5	4/5
0.65	2	5	0	3	2/5	5/5
0.80	3	5	0	2	3/5	5/5
0.92	4	5	0	1	4/5	5/5
0.99	5	5	0	0	5/5	5/5

Classifier C2:

Sorted Probabilities and Corresponding True Labels:

Threshold	True Label
0.10	1
0.22	0
0.33	0
0.41	0
0.59	1
0.64	0
0.68	1
0.72	1
0.75	1
0.95	1

Calculation Table for C2:

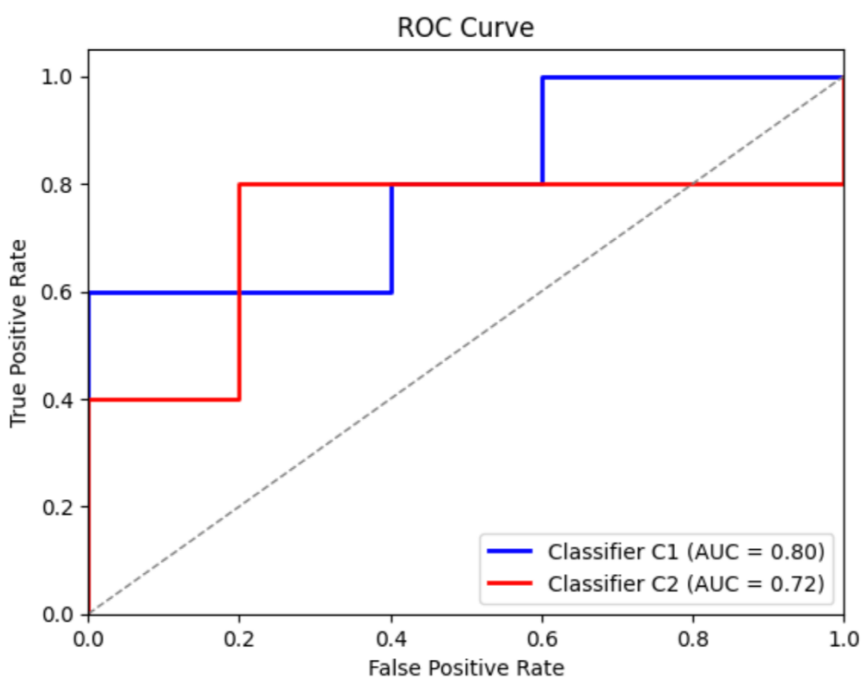
Threshold	TP	FP	TN	FN	TPR	FPR
0.10	1	0	5	4	1/5	0/5
0.22	1	1	4	4	1/5	1/5
0.33	1	2	3	4	1/5	2/5
0.41	1	3	2	4	1/5	3/5
0.59	2	3	2	3	2/5	3/5
0.64	2	4	1	3	2/5	4/5
0.68	3	4	1	2	3/5	4/5
0.72	4	4	1	1	4/5	4/5
0.75	4	5	0	1	4/5	5/5
0.95	5	5	0	0	5/5	5/5

Step 3: Plot ROC Curves and Calculate AUC

Using the tables above, we can calculate the ROC curves and AUC for both classifiers. Here, we will be using python code.

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, auc

# True labels
y_true = np.array([0, 0, 1, 0, 1, 0, 0, 1, 1, 1])

# Probabilities for C1 and C2
probs_C1 = np.array([0.15, 0.20, 0.25, 0.37, 0.41, 0.55, 0.65, 0.80, 0.92, 0.99])
probs_C2 = np.array([0.33, 0.22, 0.10, 0.41, 0.68, 0.59, 0.72, 0.75, 0.64, 0.95])

# Calculate ROC curves
fpr_C1, tpr_C1, _ = roc_curve(y_true, probs_C1)
fpr_C2, tpr_C2, _ = roc_curve(y_true, probs_C2)

# Calculate AUC
auc_C1 = auc(fpr_C1, tpr_C1)
auc_C2 = auc(fpr_C2, tpr_C2)

# Plot ROC curves
plt.figure()
plt.plot(fpr_C1, tpr_C1, color='blue', lw=2, label=f'Classifier C1 (AUC = {auc_C1:.2f})')
plt.plot(fpr_C2, tpr_C2, color='red', lw=2, label=f'Classifier C2 (AUC = {auc_C2:.2f})')
plt.plot([0, 1], [0, 1], color='grey', lw=1, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc="lower right")
plt.show()

# Print AUC values
print(f"AUC for Classifier C1: {auc_C1:.2f}")
print(f"AUC for Classifier C2: {auc_C2:.2f}")
```

Loading...




```
AUC for Classifier C1: 0.80
AUC for Classifier C2: 0.72
```

(a) The ROC curves for both classifiers are already drawn on the same plot in the image. The blue line represents Classifier C1, and the red line represents Classifier C2.

(b) The AUC values are given at the bottom of the image:
- AUC for Classifier C1: 0.80
- AUC for Classifier C2: 0.72

Classifier C1 has the larger area under the curve with an AUC of 0.80 compared to Classifier C2's AUC of 0.72.

(c) Based on the ROC curves and AUC values, Classifier C1 can be preferred as the best classifier for this problem. Because of the following reasons:

1. Higher AUC: C1 has an AUC of 0.80, which is higher than C2's 0.72. A higher AUC indicates better overall performance across different classification thresholds.

2. Better ROC curve: C1's curve (blue) is consistently above C2's curve (red), indicating that it has a better trade-off between true positive rate and false positive rate at most thresholds.

3. Higher true positive rate: At most false positive rates, C1 achieves a higher true positive rate than C2, meaning it's better at identifying positive cases while maintaining a similar or lower false positive rate.

In summary, Classifier C1 demonstrates superior performance in distinguishing between classes across various decision thresholds, making it the preferable choice for this classification problem.