Deep Learning: Biclass & Multiclass Perceptron. Laboratory No° 1

Ing. Andrés Manuel Prieto Álvarez <andres.prieto@utp.edu.co>
Maestría en ingeniería de sistemas y computación
Universidad Tecnológica de Pereira

Abstract - The following laboratory aims to present a set of techniques, methods and some strategies used to solve a set or 3 simple machine learning problems. To achieve this purpose this report will present some metrics and graphics, in order to show the process followed. Each dataset has been used to present a different characteristic from its respective perceptron; also this report shows images from matplotlib to evidence some concepts and data, to create those images python and numpy where the only images used. Check references to see the code base used.

Keywords - Perceptron, Biclass, Multiclass, One-vs-all

I. MATERIALS AND METHODS

A. Datasets

All the datasets used here have been taken from "ChenaoB/Database" github repository, then used in colab notebooks for simple perceptron "DataLeastSquareClasificationtTwoClass" (Fig 1) has been used for testing and training purposes; for simple binary perceptron implementation, "DataLeastSquareClasificationtTwoClassOutliers" (Fig 2) whose difference is that this dataset contains outliers as its name says; for multiclass perceptron "DataClasificationtMultiClass" (Fig 3) whose tags are 1, -1 and -2 by default, but those tags where changed to 0, 1 and 2, to make the analysis easier.

Each dataset was divided into training and testing sets; some representations below show each set using different markers, triangles are testing sets, circles are training.

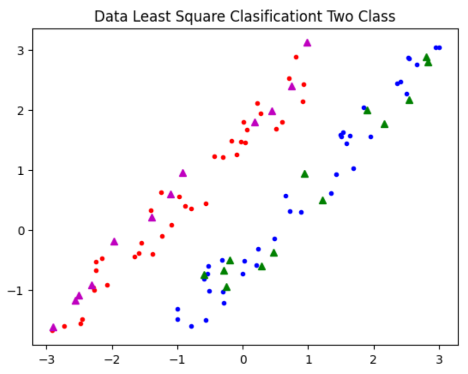

Fig 1. Data Least Square Classification Two Class

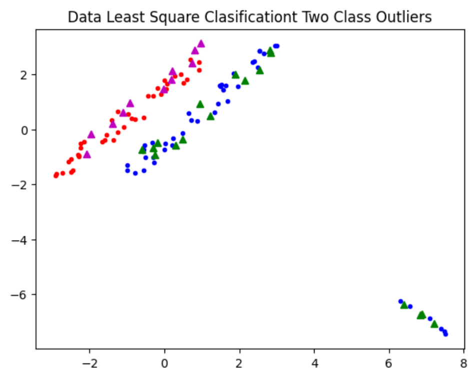

Fig 2. Data Least Square Classification Two Class Outliers

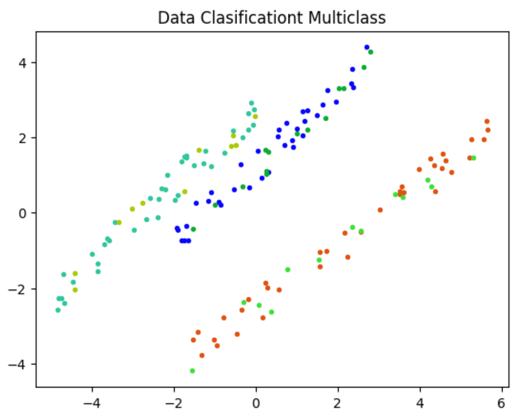

Fig 3. Data Classification Multiclass

B. Perceptron

A perceptron is a basic unit in artificial neural networks designed for binary classification. Inspired by the human brain's neuron, it takes multiple binary inputs, each multiplied by a corresponding weight, and produces an output through an activation function. Mathematically, the output is determined by the sum of the weighted inputs plus a bias term, passed through the activation function. The perceptron is trained by adjusting its weights based on the error in predictions. It is particularly effective for linearly separable tasks and forms the foundation for more complex neural network architectures.

Essentially the performance of this unit can be defined as the following pseudo code:

```
Initialize weights (w) to a random vector
Initialize learning rate (alpha) to a small value

For each training example (x, y)
  Calculate the net input (z) of the perceptron
  Calculate the output (y') of the perceptron
  Update the weights (w) of the perceptron
```

An implementation of this unit can be find in the reference[1].

C. Multiclass Perceptron

The multiclass perceptron extends the foundational concept of the perceptron to handle tasks involving multiple classes. In this framework, each class has its set of weights, and the decision-making process involves selecting the class with the highest score. Mathematically, the multiclass perceptron calculates a score for each class based on the weighted sum of input features and then assigns the input to the class with the highest score. During training, the weights are adjusted to improve the model's accuracy across all classes. This extension of the perceptron allows for the classification of instances into multiple categories, making it a versatile tool for solving complex problems in machine learning.

The pseudo-code corresponding to this kind of perceptron can be defined as:

```
Initialize weights (W) randomly
Initialize learning rate (alpha) to a small value

For each training example (x, y)
  Calculate the net input (z) for each class
    For each class c = 1, 2, ..., C
      Class calculus

  Calculate the predicted class (y')

  Update the weights (W) for each class:
    For each class c = 1, 2, ..., C
      Update calculus
```

An implementation of this kind of perceptron can be find in the same reference[1] of the previous unit.

II. RESULTS

A. Simple Perceptron

At figure 4 a summary of the performance of the binary class perceptron can be found. The model used to get those results was the implementation of the previous pseudo-code described at section I.B.

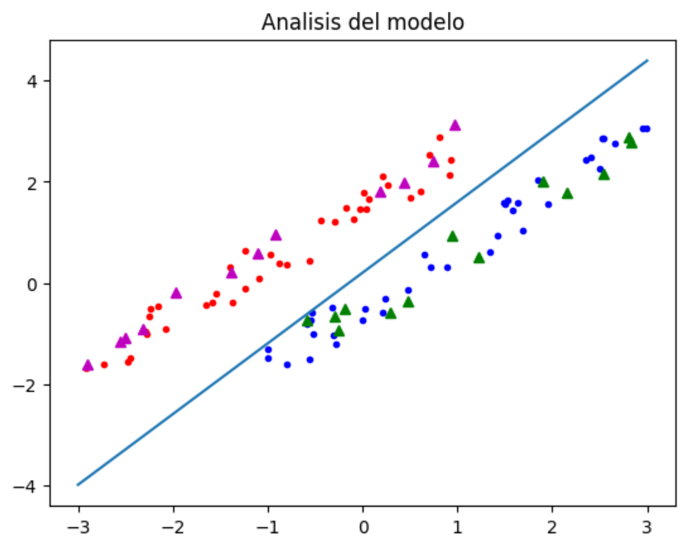

Fig 4. Model summary of simple perceptron

This performance for the simplest biclass perceptron was obtained using a 75 x 3 matrix (75% of the dataset) for training and a 25 x 3 matrix (25% of the dataset) for testing the training process, as seen in previous figure 4, the results succeed using 1000 iterations and 0.3 as learning rate.

In the following table, there are some useful metrics showing the performance of biclass perceptron.

Metric	Value
Precision	1.0
Sensibility	1.0
Specificity	1.0
Accuracy	1.0
F1 score	1.0

Table 1. Metrics simple perceptron

The results obtained for the dataset DataLeastSquareClasificationtTwoClass using a simple biclass perceptron demonstrate exceptional performance across various evaluation metrics. The precision of 1.0 indicates that all instances predicted as positive are indeed true positives, reflecting a lack of false positives. Sensibility and specificity both at 1.0 signify the model's ability to correctly identify all positive instances and exclude all negative instances, respectively. The accuracy of 1.0 indicates that the model makes no misclassifications, achieving perfect overall correctness. Additionally, the F1 score of 1.0 suggests a harmonious balance between precision and sensibility. These results collectively underscore the effectiveness of the simple biclass perceptron in accurately classifying instances in the given dataset, showcasing its robust performance and high reliability.

The confusion matrix is a table that summarizes the model's predictions, distinguishing between true positive (TP), true negative (TN), false positive (FP), and false negative (FN) instances. Each entry in the confusion matrix contributes to the precision, sensibility, specificity, and accuracy metrics. The following figure showcases this.

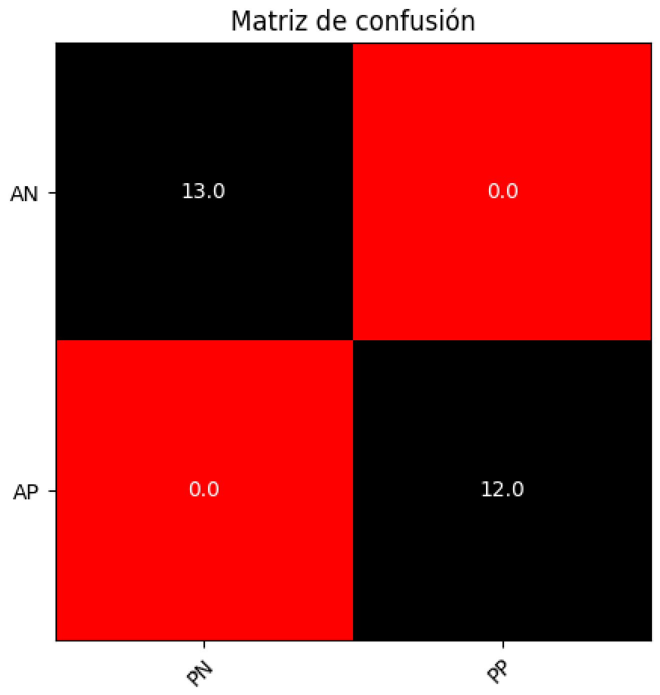

Fig 5. Confusion matrix of simple perceptron

Having a precision of 1.0 in the context of the confusion matrix implies that all instances predicted as positive are indeed true positives, leading to zero false positives. Similarly, a sensibility of 1.0 indicates that the model successfully identified all actual positive instances without any false negatives. The specificity of 1.0 signifies the absence of false positives among the negative predictions. The accuracy of 1.0 aligns with the sum of true positives and true negatives divided by the total number of instances.

These values collectively illustrate a confusion matrix characterized by perfect classification performance as seen in figure 4, further emphasizing the robustness of the simple biclass perceptron on the given dataset.

B. Biclass Perceptron with outliers

In this section, the performance of the biclass perceptron will be analyzed using the dataset "DataLeastSquareClasificationtTwoClassOutliers," which incorporates outliers, adding a layer of complexity to the analysis. Outliers are data points that deviate significantly from the overall pattern of the dataset and can challenge the robustness of machine learning models. A summary of the performance can be found in figure 6.

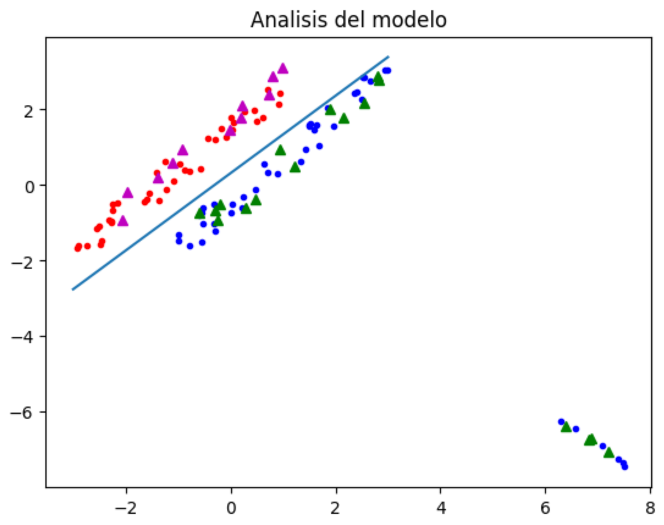

Fig 6. Model summary of simple perceptron with outliner

For this implementation, the performance of the biclass perceptron was obtained using a 82 x 3 matrix (75% of the dataset) for training and a 28 x 3 matrix (25% of the dataset) for testing the training process, biclass perceptron succeed using 1000 iterations and 0.3 as learning rate, same as previous dataset.

In the following table, there are some useful metrics showing the performance of biclass perceptron under the outliers case.

Metric	Value
Precisión	1.0
Sensibility	1.0
Specificity	1.0
Accuracy	1.0
F1 score	1.0

Table 2. Metrics simple perceptron

The evaluation metrics for the biclass perceptron on the dataset "DataLeastSquareClasificationtTwoClassOutliers" reveals consistent and robust performance, mirroring the outcomes observed in the previous dataset without outliers. Despite the introduction of outliers, the model maintains precision, sensibility, specificity, accuracy, and F1 score all at 1.0. This uniformity in performance metrics suggests that the biclass perceptron continues to accurately classify instances even in the presence of outliers.

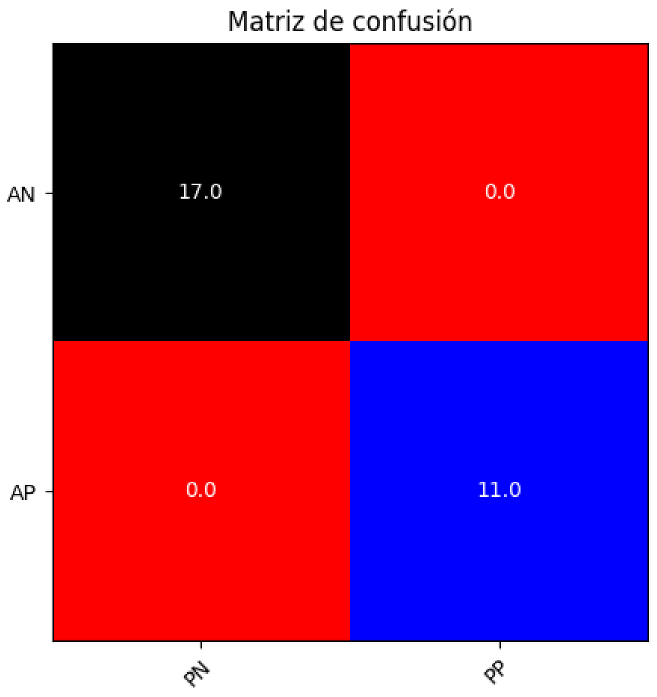

Fig 7. Model summary of simple perceptron with outliner

The confusion matrix further corroborates these results, highlighting zero false positives and false negatives. This resilience to outliers underscores the model's adaptability and reliability, reinforcing its effectiveness in scenarios where the dataset may contain unexpected or anomalous data points.

C. Multiclass Perceptron

The multiclass perceptron introduces a shift in focus from binary to multiple classes, representing a natural progression in complexity. Unlike the previous analyses on biclass perceptrons, this section explores the model's capacity to categorize instances into multiple classes in the dataset. The multiclass perceptron addresses the nuanced challenges posed by classifying data points into more than two categories. This extension requires an intricate interplay of weights for each class and introduces a decision-making process that considers the highest scoring class.

"DataClasificationtMultiClass" was harder to summarize, so in it was splitted into two different figures below, figure 8 shows training stage, while figure 9 shows testing stage.

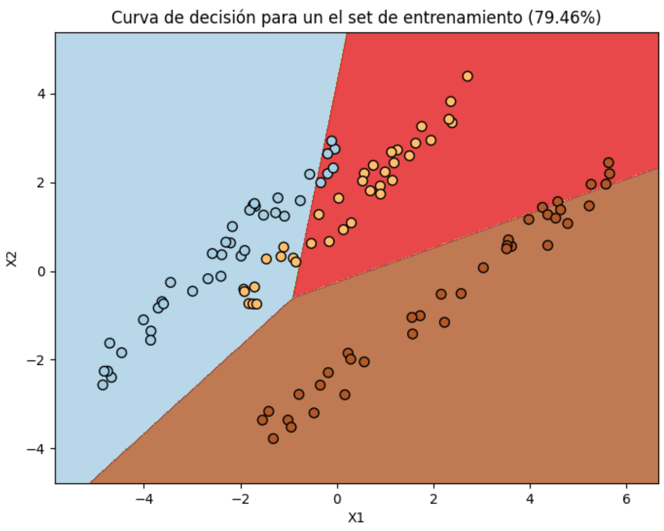

Fig 8. Train set of multiclass perceptron

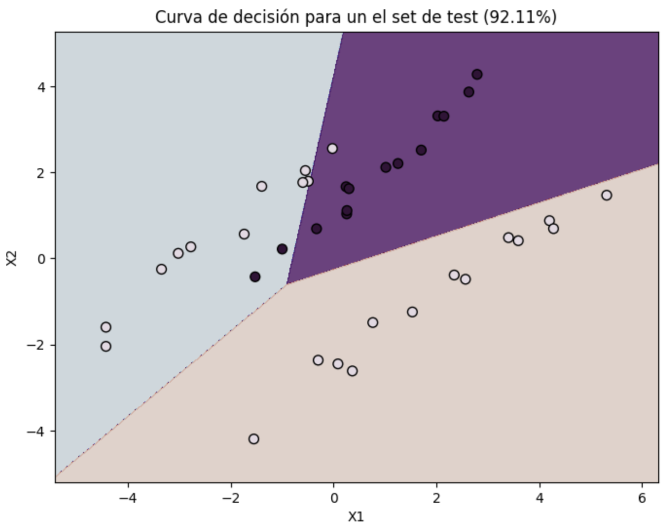

Fig 9. Test set of multiclass perceptron

For multiclass perceptron the implementation was created using the One-vs-all [3] approach to transform existing code base into a multiclass perceptron, this process caused some issues in complexity, that might be the source cause of the accuracy loss because, figure 9 shows that testing the multiclass prediction the accuracy obtained was 92.11% instead of 100% as expected, another possible cause of this could be randomness in Initialization of parameters, since this common issue affected this practice significantly more than previous ones. Another possible failure could have been the learning rate or a convergence issue; the model may not converge to an optimal solution even through several test where done using different parameters, but the best accuracy obtained while testing was 92.11%, otherwise overfitting or underfitting could still be the source cause of accuracy loss, but to determine if this is the source of error would implicated training and testing white even more possible sets of iterations and learning rates.

The multiclass perceptron used 5000 iterations with a learning rate of 0.3 using a 112 x 3 matrix (75% of the dataset) for training and a 38 x 3 matrix (25% of the dataset) for testing the training process, to achieve a good accuracy of 92.11% in the best test.

To analyze deeper the multiclass perceptron performance table 3 can showcase some important data.

Metrics	Class 1	Class 2	Class 3
Accuracy	92.11%	92.11%	92.11%
Precision	83.33%	92.31%	100%
Recall	90.1%	85.72%	100%
F1 score	86.96%	88.89%	100%

Table 3. Metrics simple perceptron

Disclaimer: The average presented in the table below is not statistically significant, it is a completion rate percentage

The performance metrics for the multiclass perceptron on the dataset provide valuable insights into its classification capabilities across multiple classes. The accuracy, precision, recall and F1 score for each class showcase a good understanding of the model's strengths. The perceptron achieves an accuracy of 92.11% for each class, demonstrating its ability to correctly classify instances belonging to each class. The precision superior to 80% indicates the reliability of positive predictions, even if there's a margin of improvement for each class; while recall showcases the model's effectiveness in capturing a substantial portion of true positive instances.

The confusion matrix holds pivotal significance in the evaluation of a classification problem with three classes like this. In such scenarios, where the model is tasked with distinguishing among multiple classes, the confusion matrix provides a detailed breakdown of the model's predictions.

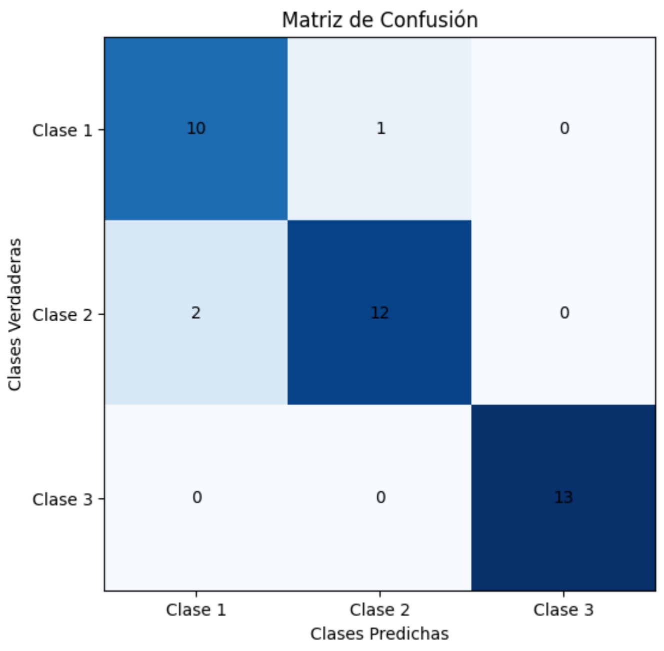

Fig 10. Confusion matrix

The confusion matrix for this implementation represents a quick view of the model's behavior, its failures and accertions the previous figure 10 showcases the misclassifications for classes 2 and 1 which are evidence of an error.

III. CONCLUSIONS

Certainly, perceptrons are a good way to solve classification problems, some implementations could face some typical issues from machine learning with stronger effects than others, a good way to achieve an automatic synchronization of parameters could be any heuristic for training processes such as genetic algorithms.

Perceptrons are a quick, not so hard way to implement classification algorithms with a good relation between the time a developer can spend coding it and the results a perceptron can achieve.

The multiclass perceptron, applied to a dataset with three classes, demonstrated commendable accuracy, precision, recall, and F1 scores across all classes. While not achieving 100% accuracy, the model's performance was notably high, showcasing its ability to handle a more complex classification task involving multiple classes.

IV. REFERENCES

[1] AndresMpa "Deep Learning". GitHub. October 13, 2023. Rerence: https://github.com/AndresMpa/deep-learning/tree/main/labs/task/laboratory_1

[2] ChenaoB "ChenaoB/Database". GitHub. November 7, 2023. Reference: https://github.com/ChenaoB/Database

[3] Amey Band "Multi-class Classification One-vs-all & One-vs-one fdsfdasfdafd". Towards Data Science. May 6, 2020