UNIVERSITÀ DI SIENA 1240

Department of Information Engineering and Mathematics

Separation Surfaces In Feedforward Neural Networks

Machine Learning Project

Jacopo Andreucci

Professor:
Marco Gori

Academic Year 2023-24

Abstract

This project focuses on the visualization and analysis of separation surfaces generated by neural networks with inputs in $\mathbb{R}^2$. The separation surface is defined as the set $S = \{x \in \mathbb{R}^2 : f(w, x) = 0\}$, where $f(w, x)$ represents the output of a neural network with given weights w. To explore the behavior of these surfaces, we implemented some programs capable of drawing them for both linear and non-linear classifiers. Two custom datasets were created to compare the performance and characteristics of these classifiers. The linear classifier produced a simple, straight-line separation surface, while the non-linear classifier generated more complex, curvilinear boundaries. The comparison highlighted the strengths and limitations of each approach, demonstrating that while linear classifiers are computationally simpler and easier to interpret, non-linear classifiers provide greater flexibility and accuracy in capturing complex patterns within the data. To further illustrate this, we applied a neural network to the famous Iris dataset, a real-world example known to be non-linearly separable. This application showcased the neural network's ability to generate non-linear separation surfaces, effectively capturing the underlying structure of the dataset. This project underscores the importance of choosing the appropriate model based on the nature of the dataset and the desired outcome.

Contents

Chapter 1

Introduction

1.1 Neural Networks and Their Significance

Neural networks have become a cornerstone of modern machine learning, enabling advancements in areas such as image recognition, natural language processing, and predictive analytics. These models are particularly valued for their ability to learn complex patterns and relationships within data. A fundamental aspect of neural network classifiers is their decision boundaries, also known as separation surfaces, which define the regions of input space where different classes are predicted.

1.2 Motivation and Objectives

Understanding and visualizing these separation surfaces is crucial for interpreting the behavior of neural networks. This project aims to investigate and visualize the separation surfaces of neural networks with inputs in $\mathbb{R}^2$ both for custom and real-world datasets. Specifically, the project will compare the characteristics of separation surfaces generated by linear and non-linear classifiers within custom datasets, to highlight the strengths and limitations of each approach in handling different types of data. In addition, the validity of neural networks is tested on real-world data to explore their capability in a more complex scenario.

1.3 Linear vs. Non-Linear Classifiers

Linear classifiers create decision boundaries that are simple, straight lines, making them easy to interpret. However, their simplicity can be a limitation when dealing with complex datasets. Non-linear classifiers, on the other hand, can form more intricate decision boundaries, allowing them to capture complicated relationships within the data. This project explores these differences by applying both types of classifiers to custom datasets and analyzing the resulting separation surfaces.

1.4 Custom Datasets

The choice of datasets is critical in evaluating the performance of classifiers. For this project, two custom datasets were created to provide a meaningful comparison between linear and non-linear classifiers. The characteristics of these datasets are tailored to demonstrate the potential strengths and weaknesses of each classifier type.

1.5 Real World Example with the Iris Dataset

To further illustrate the concepts of non-linear separation surfaces, we apply our analysis to the well-known Iris dataset, a real-world example in the field of machine learning. The Iris dataset is inherently non-linearly separable, making it an ideal candidate for demonstrating the advantages of non-linear classifiers. For the purpose of this project, we focus on only two features of the dataset, allowing us to visualize the separation surfaces in $\mathbb{R}^2$, although some precision is lost in the classification. Using a neural network, we generate complex, non-linear decision boundaries that effectively separate the different classes, highlighting the network's ability to capture the underlying structure of the data.

1.6 Structure of the Report

The remainder of this report is structured as follows: Chapter 2 covers the theoretical background of neural networks and separation surfaces. Chapter 3 details the methodology used to implement and visualize the classifiers, while Chapter 4 presents the results and analysis. Finally Chapter 5 discusses the implications of the findings and concludes the report suggesting directions for future work.

Chapter 2

Literature Review

2.1 Artificial Neural Networks (ANNs)

Artificial Neural Networks (ANNs) are computational models inspired by the human brain, designed to recognize patterns and make decisions based on input data. Introduced in the mid-20th century, ANNs have evolved significantly and are now a cornerstone of machine learning and artificial intelligence. The fundamental unit of an ANN is the neuron, which processes inputs, applies a transformation (via an activation function), and produces an output. ANNs typically consist of multiple layers: an input layer, one or more hidden layers, and an output layer. Learning in ANNs is primarily achieved through backpropagation, where errors are propagated backward through the network to adjust the weights, minimizing the loss function.

2.2 Feedforward Neural Networks (FNNs)

Feedforward Neural Networks (FNNs) represent a specific class of ANNs where connections between neurons do not form cycles. Information in FNNs flows in one direction: from the input layer, through the hidden layers, and finally to the output layer. The absence of cycles ensures that the model is acyclic, making it simpler to analyze and train. During training, FNNs adjust their weights through backpropagation, utilizing a loss function to measure the discrepancy between the predicted and actual outputs. FNNs are widely used in tasks such as image classification, regression, and pattern recognition.

2.3 Logistic Regression

Logistic regression is a fundamental statistical method used for binary classification. It models the probability that a given input belongs to a particular class using the logistic function, which produces a value between 0 and 1. The logistic function is defined as:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

where z is the linear combination of input features and their corresponding weights. Logistic regression is closely related to neural networks, as it can be viewed as a simple neural network with no hidden layers. The model is trained by optimizing the cross-entropy loss, which measures the difference between the predicted probabilities and the actual labels.

2.4 Separation Surfaces

Separation surfaces, also known as decision boundaries, are crucial concepts in the study of neural networks and machine learning models. Mathematically, a separation surface is defined as the set of points in the input space for which the classifier's output is constant, typically at a threshold that distinguishes between different classes. In the simplest case, a linear classifier, such as a logistic regression model, produces a separation surface that is a hyperplane. In $\mathbb{R}^2$, this corresponds to a straight line, while in higher dimensions, it becomes a plane or hyperplane. The linearity of this surface implies that the classifier can only separate classes that are linearly separable, meaning the classes can be divided by a straight line or plane. However, many real-world datasets are not linearly separable. To handle such complexity, neural networks with hidden layers, known as deep neural networks, can be employed. These networks are capable of learning non-linear separation surfaces, which are often curvilinear and can adapt to the intricate structure of the data. The non-linear surfaces arise from the complex interactions between multiple layers of neurons, where each layer applies a non-linear transformation to the data. As a result, the final decision boundary can warp and twist through the input space, effectively capturing the underlying patterns that simpler models might miss. Visualizing separation surfaces is particularly valuable in understanding how a model partitions the input space and makes predictions. In $\mathbb{R}^2$, these surfaces can be represented as curves, and in $\mathbb{R}^3$ as surfaces. For higher-dimensional spaces, although direct visualization is not possible, techniques such as dimensionality reduction can be employed to project the data into lower dimensions where the separation surfaces can be examined. The study of separation surfaces provides insight into the decision-making process of neural networks and helps in understanding their generalization capabilities. By analyzing these surfaces, researchers can assess how well a model is likely to perform on unseen data, identify potential overfitting, and explore the model's robustness to variations in the input space. This analysis is essential for advancing the design and application of neural networks in various complex tasks.

Chapter 3

Methodology

This chapter outlines the methodology used in this project, including the generation and description of two custom datasets and the illustration of the real-world dataset. There will be reported also the implementation of a linear classifier using logistic regression, and a non-linear classifier using a feedforward neural network (MLP). The goal is to analyze the performance of these classifiers on both linearly and non-linearly separable data.

3.1 Datasets

3.1.1 Dataset 1: Linearly Separable Points

The first dataset consists of points in $\mathbb{R}^2$ that belong to two distinct classes. These points are generated from two Gaussian distributions with different means, ensuring that the classes are linearly separable. This type of dataset is ideal for evaluating the performance of linear classifiers, as the decision boundary between the two classes is a straight line.

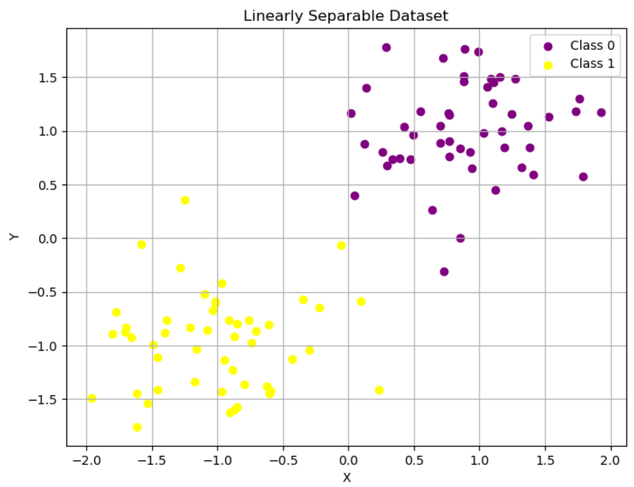

Figure 3.1: Linearly separable dataset in $\mathbb{R}^2$. The yellow and purple points represent the two different classes.

3.1.2 Dataset 2: Concentric Circles

The second dataset consists of points in $\mathbb{R}^2$ distributed in concentric circles. Each circle corresponds to a different class, making the dataset non-linearly separable. The points are generated uniformly along circles with different radii. This dataset is designed to test the limitations of linear classifiers and the capabilities of non-linear models.

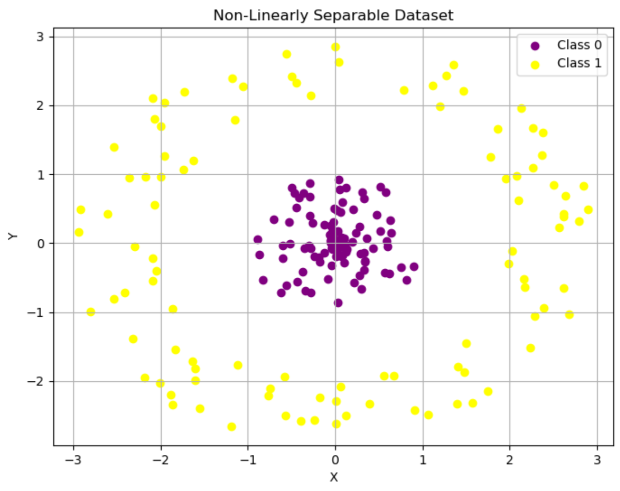

Figure 3.2: Concentric circles dataset in $\mathbb{R}^2$. The inner and outer circles represent two different classes.

3.1.3 Dataset 3: Iris Dataset with 2 Features and 3 Classes

The Iris dataset is a well-known example in machine learning, consisting of 150 samples from three different species of iris flowers: Iris Setosa, Iris Versicolor, and Iris Virginica. Each sample has four features: sepal length, sepal width, petal length, and petal width. For this project, we focus on all possible pairwise combinations of these features, resulting in six 2D plots. This reduction to two features allows us to visualize the separation surfaces in $\mathbb{R}^2$. Below are the six scatter plots, each corresponding to a different combination of two features from the Iris dataset:

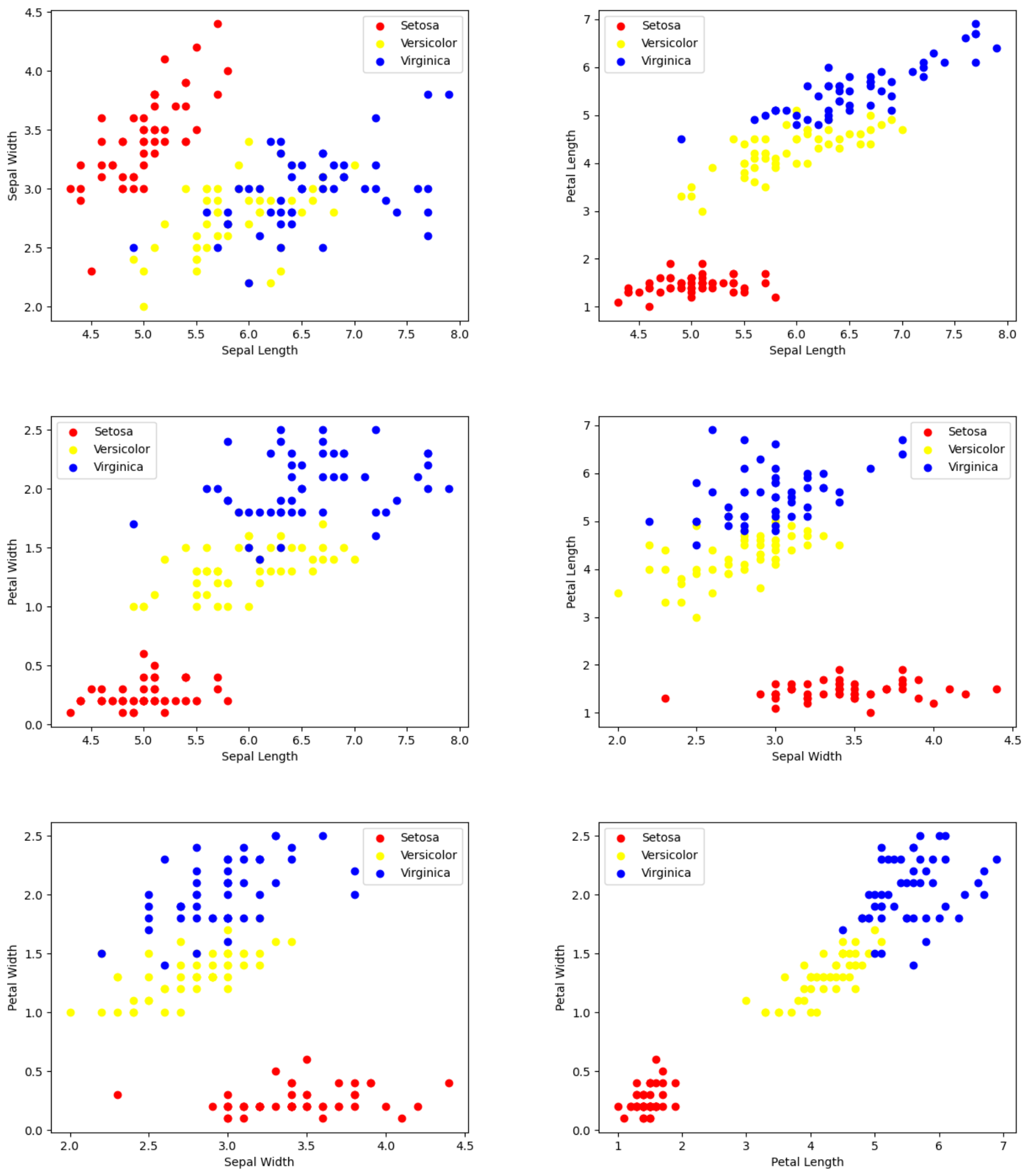

Figure 3.3: Pairwise feature combinations of the Iris dataset showing the distribution of three species (Setosa, Versicolor, Virginica). Each plot demonstrates the overlap between the classes, indicating non-linear separability.

As seen in the plots above (Figure 3.3), the Iris dataset exhibits overlapping regions between different species, particularly between Iris versicolor and Iris virginica. This overlap makes the dataset non-linearly separable in some feature combinations. For instance, the 'petal length vs. petal width' plot shows a more distinct separation, especially for the Iris setosa class, while the other species are less clearly separated by a linear boundary. These visualizations highlight the complexity of the data and the necessity of using non-linear classifiers, such as neural networks, to effectively capture the underlying patterns. Linear classifiers would struggle to create effective decision boundaries for this dataset due to the intertwined nature of the classes in many of these feature spaces. By focusing on two features at a time, we can clearly see how non-linear separation surfaces are needed to distinguish the classes in $\mathbb{R}^2$.

3.2 Logistic Regression Classifier

Logistic Regression is a linear model commonly applied to binary classification tasks. The model predicts the probability that a given input belongs to a specific class, allowing for classification into one of two categories.

3.2.1 Model Architecture

The Logistic Regression model is implemented as a custom class in PyTorch, inheriting from the `nn.Module` class. The architecture consists of a single linear layer that directly maps the input features to a single output neuron. The output neuron represents the predicted probability of the positive class. The model structure is as follows:

- **Linear Layer:** The model includes a single linear transformation defined by the `nn.Linear` layer, which performs the operation:

  $$z = W \cdot X + b$$

  where W is the weight matrix, X is the input feature vector, and b is the bias term. The input size of this layer corresponds to the number of features in the dataset, and the output is a single scalar value.

- **Activation Function:** The output of the linear layer is passed through a sigmoid activation function:

  $$\hat{y} = \sigma(z) = \frac{1}{1 + e^{-z}}$$

  The sigmoid function squashes the output to the interval $[0, 1]$, which can be interpreted as the probability of the input belonging to the positive class.

3.2.2 Training Methodology

The model is trained using the `fit` method, which implements a custom training loop. The methodology consists of the following key components:

- **Loss Function:** The binary cross-entropy loss function (`nn.BCELoss`) is used to measure the discrepancy between the predicted probabilities and the actual target labels. The loss function is defined as:

  $$\mathcal{L}(\hat{y}, y) = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

where $\hat{y}_i$ is the predicted probability, y_i is the true label, and N is the number of samples.

- **Optimizer:** Stochastic Gradient Descent (SGD) is employed to minimize the loss function. The learning rate is set to 0.01 by default, and the model weights are updated iteratively using backpropagation.
- **Training Loop:** The model is trained for a specified number of epochs (default 1000). During each epoch, the following steps are performed:
  1. Perform a forward pass to compute the predicted outputs.
  2. Calculate the loss using the binary cross-entropy criterion.
  3. Perform the gradient descent algorithm to compute gradients.
  4. Update the model parameters using the optimizer.

3.2.3 Evaluation Methodology

The model's performance is evaluated using the `evaluate_model` function. The evaluation process involves:

- **Prediction:** The `predict` method is utilized to generate predictions for new data points. The input features are passed through the network, and the output probabilities are thresholded at 0.5 to produce binary class labels:

$$\text{prediction} = \begin{cases} 1 & \text{if } \hat{y} \geq 0.5 \\ 0 & \text{if } \hat{y} < 0.5 \end{cases}$$

- **Accuracy Calculation:** The accuracy is computed as the proportion of correctly classified samples:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Samples}}$$

3.3 Multi-Layer Perceptron Classifier

This section details the Multi-Layer Perceptron (MLP) model employed for both linear and non-linear dataset classification. The MLP is a type of feedforward neural network that is particularly effective for tasks requiring non-linear decision boundaries.

3.3.1 Model Architecture

The MLP model is implemented as a custom class using PyTorch, and it consists of three key components:

- **Input Layer:** The input layer receives the feature vector X, where the dimensionality corresponds to the number of input features in the dataset.

- **Hidden Layer:** The hidden layer is defined by a linear transformation, followed by a ReLU activation function:

  $$h = \mathrm{ReLU}(W_1 \cdot X + b_1)$$

  where W_1 and b_1 represent the weights and biases for the hidden layer, respectively. The ReLU activation introduces non-linearity, enabling the model to capture complex patterns in the data.

- **Output Layer:** The output layer applies a linear transformation to the hidden layer output, followed by a Sigmoid activation function:

  $$\hat{y} = \sigma(W_2 \cdot h + b_2)$$

  where W_2 and b_2 are the weights and biases for the output layer. The Sigmoid function outputs a probability in the range $[0, 1]$, suitable for binary classification.

3.3.2 Training Methodology

The model is trained using a custom `fit` method, which integrates the following components:

- **Loss Function:** The Binary Cross-Entropy Loss function (`nn.BCELoss`) is used to evaluate the discrepancy between the predicted probabilities and the true labels:

  $$\mathcal{L}(\hat{y}, y) = -\frac{1}{N}\sum_{i=1}^{N} \left[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)\right]$$

  where $\hat{y}_i$ represents the predicted probability for sample i, and y_i is the actual label.

- **Optimizer:** The Adam optimizer (`optim.Adam`) is employed to update the model parameters. Adam adapts the learning rate during training, improving convergence speed and performance.

- **Training Loop:** The model is trained for a specified number of epochs (default 1000). Each epoch consists of:

  1. Performing a forward pass to compute the predictions.
  2. Calculating the loss using the Binary Cross-Entropy Loss function.
  3. Performing backpropagation to compute gradients.
  4. Updating the model weights using the Adam optimizer.

3.3.3 Evaluation Methodology

The model's performance is evaluated using the `evaluate_model` function that estimates the value of the predictions and of the accuracy exactly as the logistic regression model.

3.4 Multi-Layer Perceptron Classifier for Multi-Class Classification

This section describes the implementation of a Multi-Layer Perceptron (MLP) classifier designed for multi-class classification tasks. The model is specifically applied to the Iris dataset, where it classifies the data into three different classes using only two features.

3.4.1 Model Architecture

The MLP model is implemented using PyTorch and consists of the following layers:

- **Input Layer:** The input layer receives a feature vector X of size 2, corresponding to the two selected features from the Iris dataset.
- **Hidden Layer:** The hidden layer is a fully connected linear layer with a specified number of neurons (hidden size). The output of this layer is passed through a ReLU activation function:
  $$h = \text{ReLU}(W_1 \cdot X + b_1)$$
  where W_1 and b_1 represent the weights and biases of the hidden layer.
- **Output Layer:** The output layer is also a fully connected linear layer with three neurons, corresponding to the three classes in the dataset. A Softmax activation function is applied to the output:
  $$\hat{y} = \text{Softmax}(W_2 \cdot h + b_2)$$
  The Softmax function normalizes the output into a probability distribution across the three classes.

3.4.2 Training Methodology

The model is trained using a custom training loop implemented in the `train_model` function. The following components are integral to the training process:

- **Loss Function:** The Cross-Entropy Loss (`nn.CrossEntropyLoss`) is used to measure the difference between the predicted probability distribution and the true class labels. The loss function is defined as:
  $$\mathcal{L}(\hat{y}, y) = -\sum_{i=1}^{C} y_i \log(\hat{y}_i)$$
  where C is the number of classes, $\hat{y}_i$ is the predicted probability for class i, and y_i is the true label.
- **Optimizer:** The Adam optimizer (`optim.Adam`) is employed to update the model parameters. Adam optimizes the model by adjusting the learning rate based on the first and second moments of the gradient.
- **Training Loop:** The model is trained over a specified number of epochs. Each epoch consists of:

1. A forward pass to compute the model output.
2. Calculation of the loss using the Cross-Entropy Loss function.
3. Backpropagation to compute the gradients.
4. Weight update using the Adam optimizer.

3.4.3 Evaluation Methodology

The model's performance is evaluated using the `evaluate_model` function. The evaluation process involves:

- **Prediction:** The model predicts the class of each sample in the test set by selecting the class with the highest probability from the Softmax output.
- **Accuracy Calculation:** The accuracy is computed as the proportion of correctly classified samples:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Samples}}$$

Chapter 4

Results

For each experiment, 80% of the dataset was used for training the models (Training set), while the remaining 20% was reserved for testing (Test set). The performance of each classifier in terms of accuracy and loss is evaluated only on the test set, and their decision boundaries are visualized to demonstrate their behavior on both linearly and non-linearly separable data.

4.1 Results on Linearly Separable Dataset

4.1.1 Logistic Regression Performance

The logistic regression model was first applied to the linearly separable dataset. As expected, the model produced a linear decision boundary that effectively separated the two classes, as shown in Figure 4.1.

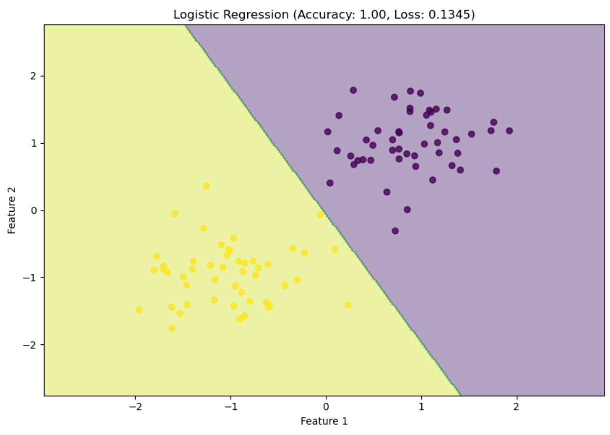

Figure 4.1: Decision boundary of logistic regression on the linearly separable dataset.

The model achieved an accuracy of 100% on the test set. This indicates that the model is perfectly classifying all the instances in the dataset, which is an excellent outcome. Given that the classes in the dataset are linearly separable, this result is expected and confirms that the model has learned the decision boundary effectively.
The model's loss is 0.1345 so while the accuracy is perfect, the log-loss value reflects the confidence level of the model's predictions. Although the model correctly classifies every

instance, it does not assign a probability of exactly 1.0 to the correct class in all cases. The log-loss measures the extent to which the predicted probabilities deviate from 1.0 (for the correct class) and 0.0 (for the incorrect class).

4.1.2 MLP Performance

The MLP was also applied to the linearly separable dataset. The resulting decision boundary is shown in Figure 4.2. The MLP, with its non-linear capabilities, also effectively separated the classes, although the boundary appears slightly more complex due to the model's flexibility.

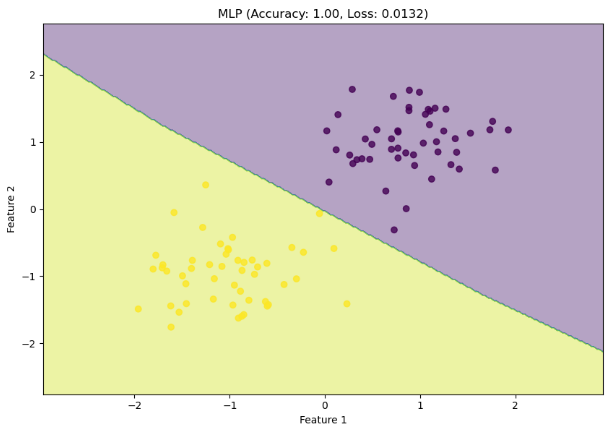

Figure 4.2: Decision boundary of MLP on the linearly separable dataset.

The MLP model achieved an accuracy of 100% on the test set, just like the logistic regression model. This means that the model is also perfectly classifying all the instances in the dataset and this result reaffirms that the MLP has successfully learned the decision boundary.
The model's loss is significantly lower than the loss obtained with the logistic regression model (0.1345). This lower loss indicates that the MLP model is not only classifying the instances correctly but is also doing so with greater confidence. The predicted probabilities for the correct classes are closer to 1.0, resulting in a smaller deviation and thus a lower loss.

4.2 Results on Concentric Circles Dataset

4.2.1 Logistic Regression Performance

Applying logistic regression to the concentric circles dataset resulted in poor performance, as shown in Figure 4.3. The linear decision boundary was unable to separate the two classes effectively, resulting in a significant number of misclassifications.

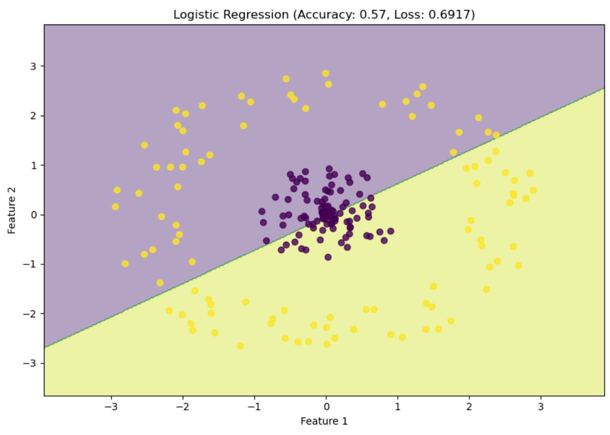

Figure 4.3: Decision boundary of logistic regression on the concentric circles dataset.

The linear regression model achieved an accuracy of 0.57, which is significantly lower than what would be considered effective for most classification tasks. An accuracy of 57% indicates that the model is only slightly better than random guessing (which would yield an accuracy of around 50% for a balanced binary classification problem). This poor performance is a clear indication that the linear model is not suitable for capturing the underlying patterns in the data.
Moreover a higher loss value like this suggests that the model's predictions are not only often incorrect but also lack confidence. The predicted probabilities are likely close to 0.5, indicating uncertainty in classification decisions.

4.2.2 MLP Performance

The MLP, when applied to the concentric circles dataset, successfully learned the non-linear decision boundary required to separate the two classes, as shown in Figure 4.4.

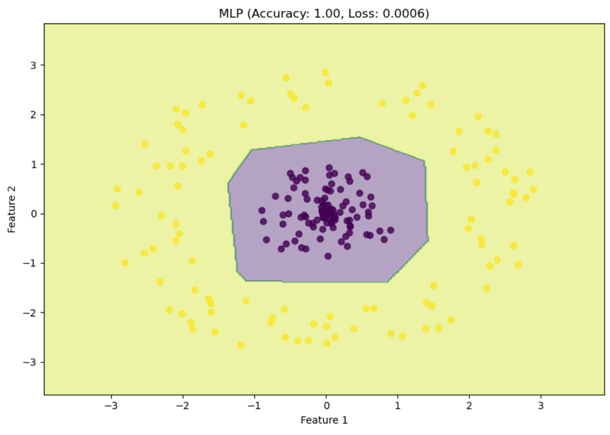

Figure 4.4: Decision boundary of MLP on the concentric circles dataset.

The MLP model achieved an accuracy of 100% on the non-linear dataset. This indicates that the MLP model is perfectly classifying all instances, which demonstrates its ability to capture and model the non-linear relationships present in the data. The perfect

accuracy highlights the MLP's flexibility and capacity to learn complex patterns that a linear model cannot.
The model's loss is exceptionally low, at 0.0006. This near zero loss suggests that not only is the MLP model classifying every instance correctly, but it is also doing so with extremely high confidence. The predicted probabilities for the correct classes are very close to 1.0, indicating a model that is well-calibrated and certain in its predictions.

4.3 Results on Iris Dataset

The MLP was then applied to the Iris dataset using the architecture designed for multi-class classification, as described in Chapter 3. The resulting decision boundaries are shown in Figure 4.5.

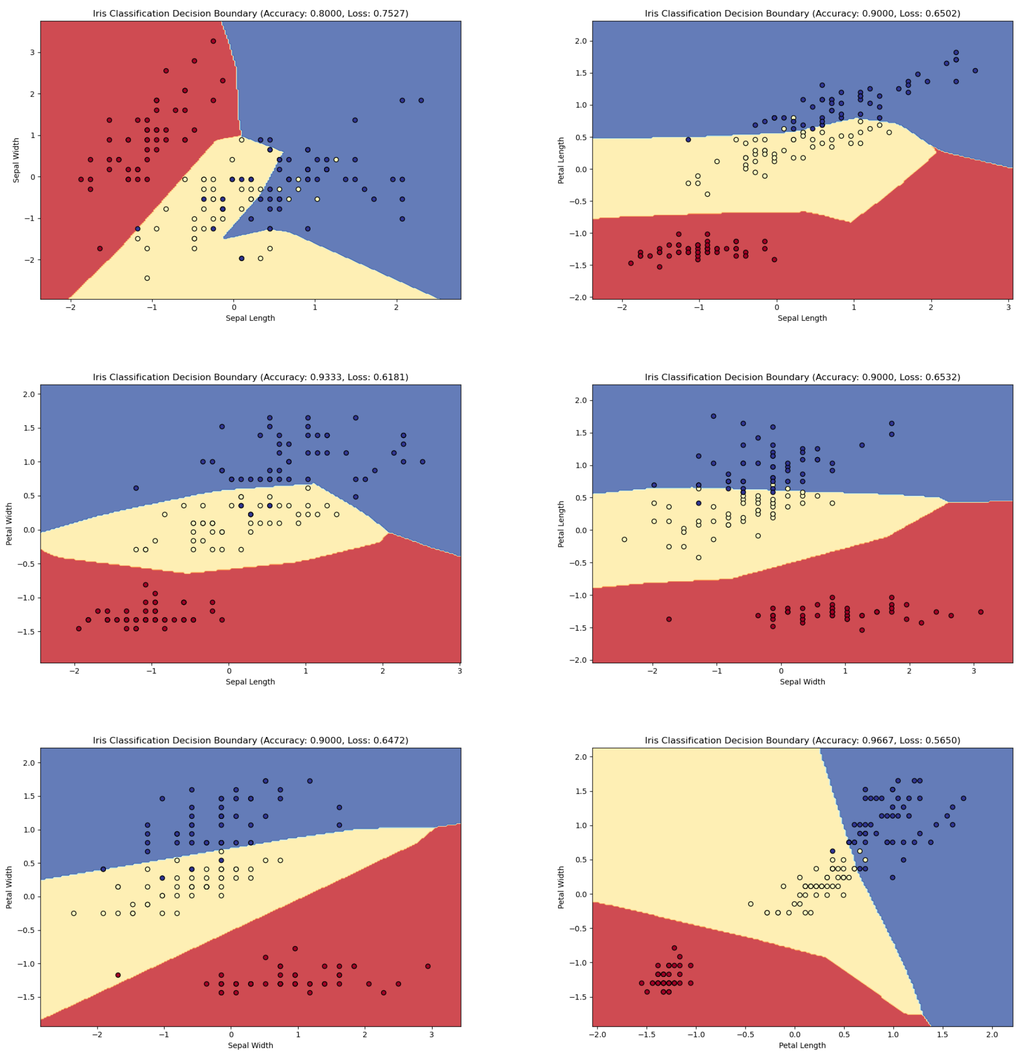

Figure 4.5: MLP decision boundaries for each pair of features in the Iris dataset.

Plotting the decision boundaries for the MLP model, even with only two features and despite the lower accuracy, can effectively demonstrate the model's ability to handle complex decision-making in a multi-class scenario. This is particularly interesting in the

context of a real-world dataset like Iris, where the classes are not perfectly separable with just two features.
Even if the accuracy isn't perfect, the plot will show the regions of the feature space that the model associates with each class.
When using only two features, the model's accuracy drops from what would typically be near 100% with all four features to 80% or 90%. This drop in accuracy occurs because the model now has less information to work with. The two selected features may not provide enough separation between the classes, causing the model to misclassify some instances.
The high loss values indicate that the model's predictions are not very confident and uncertain. This uncertainty arises from the reduced ability to capture the nuances in the data when only two features are used.

4.4 Comparison and Analysis

The results clearly demonstrate the strengths and limitations of both classifiers. Logistic regression, with its linear decision boundary, performed exceptionally well on the linearly separable dataset but failed to generalize to the non-linear concentric circles dataset. The MLP, on the other hand, handled all datasets effectively, showcasing its ability to model complex decision boundaries.
Due to this characteristic, it has been used in trying to create plausible decision surfaces for examples from the real world (Iris dataset) even if it has been treated in a simplified way, taking 2 features at a time keeping within the scope of the description of separation surfaces in $\mathbb{R}^2$.

Chapter 5

Discussion and Conclusions

This chapter synthesizes the results from applying logistic regression and a feedforward neural network (MLP) to three datasets: a linearly separable dataset, a dataset of concentric circles, and the Iris dataset. The discussion evaluates the performance of the models on the first two datasets, focuses on the neural network's performance on the Iris dataset, examines the implications of the findings, acknowledges limitations, and proposes directions for future research.

5.1 Discussion of Results

5.1.1 Performance of Logistic Regression

Logistic regression performed exceptionally well on the linearly separable dataset, achieving perfect classification accuracy with a straightforward linear decision boundary. This result highlights its effectiveness for problems where classes are linearly separable. However, its performance deteriorated on the concentric circles dataset, where the linear model failed to capture the non-linear boundaries needed to separate the classes. This outcome underscores the limitations of logistic regression when dealing with complex, non-linear problems.

5.1.2 Performance of the MLP

The MLP demonstrated its capability to handle both linearly and non-linearly separable data. On the linearly separable dataset, it achieved perfect accuracy, although with a slightly more complex decision boundary than necessary. On the concentric circles dataset, the MLP successfully modeled the non-linear decision boundary, clearly separating the classes. These results confirm the MLP's versatility and robustness across a range of classification tasks.

5.1.3 Comparison of Classifiers on Synthetic Datasets

The comparison between logistic regression and the MLP on the linearly separable and concentric circles datasets highlights a trade-off between simplicity and flexibility. Logistic regression is simple, interpretable, and computationally efficient but is limited to linearly separable data. The MLP, while more powerful and adaptable to non-linear data, requires careful tuning to avoid overfitting and comes with a higher computational cost. This trade-off is crucial when selecting models for specific tasks, especially when interpretability, performance, and computational resources must be balanced.

5.1.4 Performance of the MLP on the Iris Dataset

The Iris dataset serves as a real-world example that showcases the strengths of the MLP. In this case, only the MLP was tested due to the inherent non-linear relationships in the dataset, especially between the Iris Versicolor and Iris Virginica classes. The MLP effectively managed the multi-class problem, producing non-linear decision boundaries that captured the relationships between features. The decision surfaces generated by the MLP highlighted the model's ability to handle real-world data with complex feature interactions, even if as we have seen in the previous chapter working on only 2 features had a big impact in terms of performance.

5.2 Implications of the Findings

5.2.1 Impact on Model Selection

The results suggest that model selection should align with the data's characteristics. For simple, linearly separable data, logistic regression remains an effective choice. However, for complex, non-linear datasets, models like the MLP are essential. The decision to use a more complex model should consider not only the data's complexity but also the need for interpretability and the computational resources available.

5.2.2 Theoretical Insights

The findings reinforce the theoretical understanding that linear classifiers are insufficient for non-linear decision boundaries. The MLP's ability to model non-linear relationships through hidden layers and activation functions is crucial for handling complex datasets. This underscores the importance of non-linear models in modern machine learning, particularly as the complexity of data and tasks increases.

5.3 Limitations of the Study

5.3.1 Dataset Limitations

The study primarily used synthetic datasets, which, while useful for controlled experiments, may not fully capture the complexity of real-world data. The results, therefore, may not generalize to more diverse or noisy datasets. Applying these models to real-world datasets, as done with the Iris dataset, is essential for validating the findings in practical contexts.

5.3.2 Model Limitations

Logistic regression's simplicity is both its strength and its limitation, as it cannot handle non-linear data. The MLP, although flexible, is prone to overfitting and requires extensive hyperparameter tuning. Additionally, the MLP's interpretability is a concern, often being seen as a “black box” model, which can complicate understanding and trust in its predictions.

5.4 Future Work

Based on the study’s findings, several directions for future research are proposed:

1. **Apply to Real-World Datasets:** Future work should apply these models to more diverse real-world datasets to validate their performance and uncover any additional challenges in practical scenarios.
2. **Explore Advanced Non-Linear Classifiers:** Investigate other non-linear classifiers, such as Support Vector Machines (SVMs) with non-linear kernels, or more sophisticated neural network architectures.
3. **Optimize MLP Architecture:** Systematic exploration of different MLP architectures, including varying the number of hidden layers, neurons, and activation functions, could optimize performance for specific datasets.
4. **Improve Hyperparameter Tuning:** Implement advanced hyperparameter tuning techniques like grid search, random search, or Bayesian optimization to enhance the generalizability of the MLP.
5. **Enhance Model Interpretability:** Develop methods to improve the interpretability of non-linear models, such as visualizing hidden layer activations or using model-agnostic interpretation techniques.
6. **Analyze Decision Boundaries:** Further analysis of the stability and sensitivity of decision boundaries, particularly in real-world applications, could provide deeper insights into model behavior.
7. **Extend to Higher Dimensions:** Although this study focused on $\mathbb{R}^2$, future research could explore separation surfaces in higher-dimensional spaces and their implications for complex data.

5.5 Conclusions

This study has provided valuable insights into the strengths and limitations of linear and non-linear classifiers. The visualization of separation surfaces across different datasets, including the Iris dataset, has highlighted the necessity of selecting the appropriate model based on the data’s underlying structure. While logistic regression offers simplicity and interpretability, it falls short in handling complex, non-linear relationships. The MLP, although more complex, demonstrates superior flexibility and performance in such scenarios. The findings emphasize the critical role of non-linear models in modern machine learning and suggest avenues for future work to further enhance our understanding and application of these techniques in real-world situations.