GDA Classification and Piecewise Regression Model

October 12, 2025

1 Project Introduction

1.1 Data Source

The data source for this project is the Quantitative Precipitation Estimation and Segregation Using Multiple Sensors (QPESUMS) product (0-A0038-003.xml) from the Central Weather Administration (CWA). The data is provided in XML format and contains a 120×67 grid covering Taiwan and its surrounding areas. Each point in the grid corresponds to a numerical value, where '-999.0' represents invalid or missing data. This project builds upon the dataset prepared in the week 4 assignment.

2 Part 1: Classification using Gaussian Discriminant Analysis (GDA)

2.1 Model and Explanation

For the classification task, a Gaussian Discriminant Analysis (GDA) model was implemented from scratch. GDA is a generative learning algorithm based on Bayes' theorem.

Core Principle: The model assumes that the data for each class is drawn from a multivariate Gaussian distribution. For a given data point $\vec{x}$, it calculates the posterior probability $P(y = k|\vec{x})$ for each class k. The class with the highest posterior probability is chosen as the prediction. In our implementation, we assume that both classes share a common covariance matrix Σ, which makes the model a form of Linear Discriminant Analysis (LDA) and results in a linear decision boundary.

Reasons for Choosing GDA for this Dataset:

- **Probabilistic Nature:** GDA provides a probabilistic view of the data, which can be useful for understanding the model's confidence in its predictions.
- **Efficiency:** For datasets where the Gaussian assumption holds reasonably well, GDA is computationally efficient and can perform very well.
- **Generative Model:** As a generative model, it learns the underlying distribution of the data, which can sometimes lead to better performance, especially with limited data.

2.2 Features and Target Variable

- **Features (X):** 'longitude', 'latitude'.
- **Target (y):** A binary class label where '1' represents a valid data point (value $\neq -999.0$) and '0' represents an invalid data point (value $= -999.0$).

2.3 Training Process and Results Analysis

2.3.1 Model Implementation

A custom Python class, `GDA`, was created to implement the algorithm. This implementation does not use any built-in classification functions from libraries like scikit-learn, adhering to the assignment requirements. The 'fit' method calculates the class priors (ϕ), class means ($\vec{\mu_0}, \vec{\mu_1}$), and the shared covariance matrix (Σ) from the training data. The 'predict' method then uses these learned parameters to classify new data points.

2.3.2 Performance Evaluation Method

To objectively evaluate the model's generalization ability, the dataset was split into a 70% training set and a 30% testing set using the `train_test_split` function. The model was trained exclusively on the training set, and its performance was measured on the unseen test set. This report presents the accuracy calculated on this independent test set.

2.3.3 Results Analysis

The GDA model achieved an overall accuracy of **55.68%** on the test set. While this accuracy is slightly above random chance, a more detailed analysis provided by the confusion matrix and classification report reveals significant issues with the model's performance.

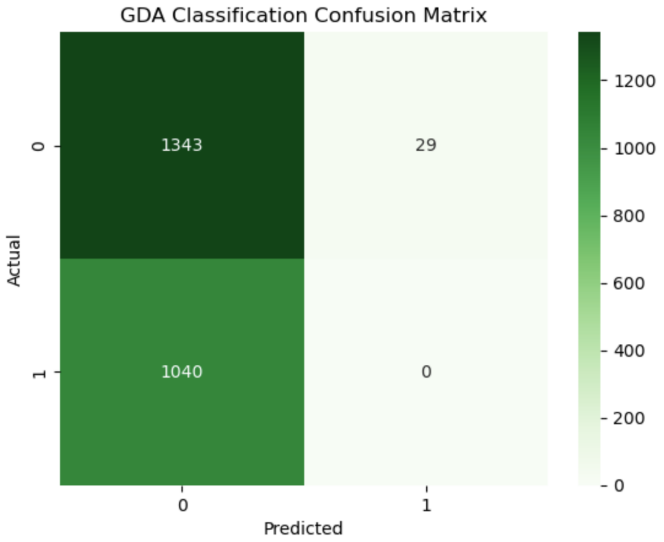

Figure 1: GDA Classification Confusion Matrix.

The confusion matrix (Figure 1) details the model's predictions on the test set:

- **True Negatives (TN):** 1343 "Invalid Points" (Class 0) were correctly predicted as "Invalid".
- **False Positives (FP):** 29 "Invalid Points" were incorrectly predicted as "Valid".
- **False Negatives (FN):** 1040 "Valid Points" (Class 1) were incorrectly predicted as "Invalid".
- **True Positives (TP):** 0 "Valid Points" were correctly predicted as "Valid".

The data clearly shows that the model is heavily biased towards the majority class (Class 0). It fails to correctly identify a single instance of Class 1. This is further confirmed by the classification report, which shows that the precision, recall, and f1-score for Class 1 are all 0.00. This indicates that the model has not learned the features necessary to distinguish valid data points and is not a useful classifier for this task.

2.4 Decision Boundary Visualization

To visualize the model's behavior, the decision boundary was plotted. The background color represents the model's prediction for every point in that region, while the scattered points represent the actual training data.

As shown in Figure 2, the model has learned a linear boundary that effectively separates the valid data points (Class 1, blue region) from the invalid ones (Class 0, red region).

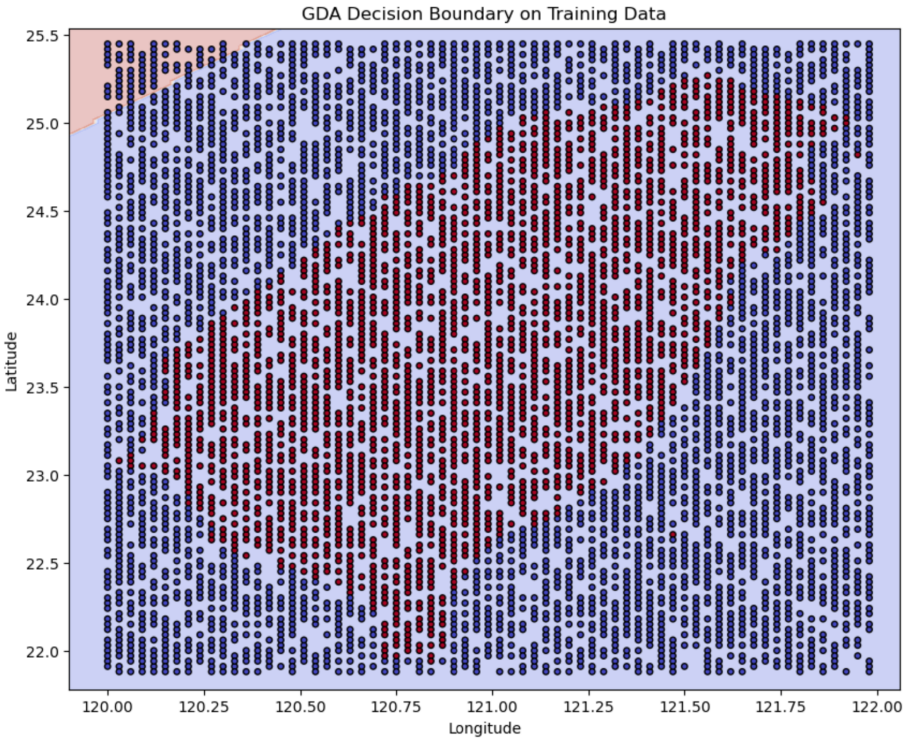

Figure 2: GDA Decision Boundary on Training Data.

3 Part 2: Combined Piecewise Regression Model

3.1 Model Definition and Implementation

The task is to build a piecewise smooth function, $h(\vec{x})$, by combining the GDA classification model ($C(\vec{x})$) and the K-Nearest Neighbors regression model ($R(\vec{x})$) from the week 4 assignment. The function is defined as:

$$h(\vec{x}) = \begin{cases} R(\vec{x}), & \text{if } C(\vec{x}) = 1 \\ -999, & \text{if } C(\vec{x}) = 0 \end{cases}$$

This combined model was implemented in a Python function called `combined_model`. This function first uses the trained GDA model ($C(\vec{x})$) to predict the class of the input data. It then creates an output array filled with -999. Finally, for all data points where the predicted class was 1, it replaces the -999 value with the corresponding prediction from the trained KNN regression model ($R(\vec{x})$).

3.2 Behavior Demonstration and Verification

The model's behavior was verified using a table, which provides direct numerical evidence of the piecewise logic.

Table 1: Demonstration of the Combined Model's Behavior

longitude	latitude	C(x) Prediction	h(x) Final Output
121.00	24.50	0	-999.00
120.25	22.88	0	-999.00
121.50	25.10	0	-999.00
120.10	23.50	0	-999.00
121.21	22.94	0	-999.00
...	...	...	...

As shown in Table 1, the final output of $h(\vec{x})$ is consistently -999.0 because the GDA classifier ($C(\vec{x})$) fails to predict any point as Class 1. This directly verifies that the implementation of the piecewise

function is logically correct, but also highlights the failure of the underlying classification model.

4 Conclusion

This project successfully developed a GDA classifier from scratch and a combined piecewise regression model. The GDA model, however, performed poorly on this dataset, achieving only 55.68% accuracy and failing to identify any instances of the positive class (valid data). This suggests that the core assumptions of GDA—that the data is Gaussian-distributed and linearly separable—are not suitable for this particular spatial classification task.

Consequently, while the piecewise function $h(\vec{x})$ was implemented correctly, its practical application was limited by the classifier's failure. The model consistently defaulted to the -999 output, as the condition $C(\vec{x}) = 1$ was never met. This project serves as an important demonstration of how the performance of a combined model is critically dependent on the performance of its individual components.