Assignment 7: Underfitting, Overfitting, and Model Selection

Data Science and Machine Learning Module

Dataset : Link

1 Conceptual Background

1.1 What is Underfitting?

Underfitting occurs when a model is too simple to capture the underlying pattern of the data.

- **Example:** Trying to fit a straight line ($y = mx + c$) to data that clearly follows a parabolic curve.
- **Signage:** High error on the Training set AND high error on the Validation set. This is often called **"High Bias."**

1.2 What is Overfitting?

Overfitting occurs when a model is too complex and begins to "memorize" the random noise in the training data rather than the actual signal.

- **Example:** Using a 15th-degree polynomial to connect 10 noisy data points. The line might touch every point, but it wiggles wildly in between them.
- **Signage:** Very low error on the Training set but very high error on the Validation set. This is often called **"High Variance."**

1.3 The "Sweet Spot"

The goal of machine learning is **Generalization**—performing well on data the model has never seen before. By plotting the error of the training and validation sets against the model's complexity (degree), we look for the point where the Validation Error is at its minimum. This is the "Sweet Spot" where the model is complex enough to learn the pattern but simple enough to ignore the noise.

2 Part 1: Polynomial Regression Analysis

Goal: Determine the optimal polynomial degree for the provided noisy dataset.

2.1 The Logic (Pseudo-code Steps)

1. **Load Data:** Import `regression_train.csv` and `regression_val.csv`.
2. **Define MSE:** Create a function to calculate Mean Squared Error:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

3. **The Experiment Loop:**
   - Create two empty lists: `train_error_list` and `val_error_list`.
   - **For degree d from 1 to 8:**
     (a) Generate polynomial features for x (i.e., $x, x^2, \ldots, x^d$).
     (b) Find the optimal weights β that minimize the MSE on the training data.
     (c) Calculate the Training MSE using the trained weights on the training features.
     (d) Calculate the Validation MSE using the same weights on the validation features.
     (e) Append both errors to their respective lists.
     (f) **Plot the fit:** Plot the training points and overlay the predicted polynomial curve for this specific degree.
4. **The Validation Curve:** Plot `train_error_list` and `val_error_list` against the degrees $[1, 2, \ldots, 8]$.
5. **Selection:** Find the index in `val_error_list` with the lowest value. Report this as the **"Optimal Degree."**

3 Part 2: Logistic Regression & The Accuracy Metric

Goal: Use non-linear boundaries to classify 2D data and evaluate performance using Accuracy.

3.1 Defining Accuracy vs. Binary Cross Entropy (BCE)

During training, we minimize **BCE** because it is a smooth, differentiable surface.

However, in a real-world engineering system, we don't care if the Sigmoid output was 0.51 or 0.99—we care if the final classification (1 or 0) was correct. Therefore, for Validation, we use **Accuracy**:

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

3.2 The Logic (Pseudo-code Steps)

1. **Load Data:** Import `classification_train.csv` and `classification_val.csv`.
2. **Expansion:** For each degree $d \in [1, 8]$, generate polynomial combinations of x_1 and x_2.
3. **Training:** Minimize the Log-Loss function on the training set to find the best weights.
4. **Validation:**
   - Predict the probability P for the validation set using the Sigmoid function.
   - Convert probabilities to labels: $\hat{y} = 1$ if $P \geq 0.5$, else 0.
   - Calculate the Accuracy of these labels against the ground truth.
5. **Selection:**
   - Plot Training Accuracy and Validation Accuracy vs. Degree.
   - Identify the degree where Validation Accuracy is highest. Note how high-degree models might achieve 100% Training Accuracy but perform worse on Validation data (Overfitting).