Part 2: Coding Task

EE214 - Assignment 1
(2025 Spring)

March 25, 2025

1 Overview

In Part 2 of the assignment, you are required to complete the task below, which is self-contained and focuses on exploring model performance with various basis functions. You are expected to use AI tools (e.g., ChatGPT, GitHub Copilot) only as an aid. You must document all AI assistance in your final report.

2 Comparative Study of Basis Functions on the Concrete Compressive Strength Dataset

2.1 Objective

Compare the performance of two different basis functions—polynomial and Gaussian—on a real-world regression task. Analyze how each approach affects the bias-variance tradeoff and overall prediction accuracy.

2.2 Data Source

Download the Concrete Compressive Strength dataset from the UCI Machine Learning Repository:

`https://archive.ics.uci.edu/ml/datasets/Concrete+Compressive+Strength`

Save the CSV file (e.g., `Concrete_Data.xls`) in your working directory.

2.3 Steps and Requirements

1. **Data Loading and Preprocessing:**

   - Load the dataset using pandas.
   - Handle missing values if present.
   - Standardize the features.
   - Split the data into training (80%) and test (20%) sets.

2. **Modeling Approaches:**

   (a) **Polynomial Regression Model:**
   For various degrees (e.g., 1, 3, 5, 7), transform the features using polynomial expansion and fit a linear regression model.

   (b) **Gaussian Basis Regression Model:**
   For various numbers of basis functions (e.g., 5, 7, 10, 15), create a design matrix using Gaussian basis functions. Use centers $\mu_j = j$ and $\sigma = 1.0$. Fit a linear regression model on the transformed data.

3. **Evaluation:**
   For both approaches, compute the training and test MSE. Investigate how increasing model complexity (in polynomial regression, increasing degree; in Gaussian basis, varying the number of basis functions) affects performance.

4. **Visualization:**
   Plot error curves (MSE vs. model complexity) for both approaches on the same graph for comparison.

5. **Analysis Questions:**
   In your written report, answer:

   (a) Which basis function (polynomial or Gaussian), and with which degree, provided better test performance and why?

   (b) How did the training and test errors change as model complexity increased for each method? Based on this, identify examples of underfitting and overfitting.

   (c) Discuss the bias-variance tradeoff observed in both approaches.

   (d) What are the potential advantages and disadvantages of each basis function in real-world regression problems?

   (e) Apply **a different basis function** from those used so far (polynomial or Gaussian), and analyze it in terms of the bias-variance tradeoff and the pros and cons in real-world regression problems. You may also try regularization, i.e., add a penalty term in the loss function.

2.4 Code Skeleton Example

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

% Load the dataset
data = pd.read_excel('Concrete_Data.xls')
print(data.head())

```

```
% Preprocess the data
X = data.drop('Concrete compressive strength(MPa, megapascals) ', axis=1).values
y = data['Concrete compressive strength(MPa, megapascals) '].values

# ----- Standardize features -----
# scaler = ...
# X_scaled = ...

# ----- Split the data -----
# X_train, X_test, y_train, y_test = ...

% (a) Polynomial Regression
from sklearn.preprocessing import PolynomialFeatures
degrees = [1, 3, 5, 7]
poly_train_errors = []
poly_test_errors = []

for degree in degrees:
    # ----- Blank: Transform features into polynomial features -----
    # Create a PolynomialFeatures object with the given degree (include bias term)
    # Transform both the training and test data

    # TODO: Fill in the code below
    # poly = ...
    # X_train_poly = ...
    # X_test_poly = ...
    # ----------------------------------------------------------------

    # ----- Blank: Fit linear regression model and evaluate performance -----
    # TODO:
    # - Initialize and train LinearRegression model
    # - Predict on both training and test sets
    # - Compute train and test MSE
    # - Append errors to respective lists

    # model_poly = ...
    # model_poly.fit(...)
    # y_train_pred = ...
    # y_test_pred = ...
    # train_mse = ...
    # test_mse = ...
    # poly_train_errors.append(...)
    # poly_test_errors.append(...)
    # ------------------------------------------------------------------------

% (b) Gaussian Basis Regression
def create_gaussian_design_matrix(x, degree, sigma=1.0):
    """
    Create a design matrix using Gaussian basis functions.
    Args:
      x: numpy array of shape (n_samples,)
      degree: number of Gaussian basis functions
      sigma: width of the Gaussian functions
    Returns:
      X: design matrix with shape (n_samples, degree + 1) including bias term
    """
```

```
    n_samples = x.shape[0]
    X = np.ones((n_samples, degree + 1))
    # ----- Blank: Fill in Gaussian basis transformation -----
    # TODO: Fill in the loop to compute Gaussian basis functions
    # for j in range(1, degree + 1):
    #     mu = ...
    #     X[:, j] = ...
    # ------------------------------------------------------------
    return X

% For simplicity, use one feature (the first column) from the training set
x_train_feature = X_train[:, 0]
x_test_feature = X_test[:, 0]

gaussian_degrees = [5, 7, 10, 15]
gauss_train_errors = []
gauss_test_errors = []

for degree in gaussian_degrees:
    X_train_gauss = create_gaussian_design_matrix(x_train_feature, degree, sigma
    =1.0)
    X_test_gauss = create_gaussian_design_matrix(x_test_feature, degree, sigma
    =1.0)

    # --- Blank: Fit Gaussian basis regression model and evaluate performance ---
    # TODO:
    # - Initialize and train LinearRegression model
    # - Predict on both training and test sets
    # - Compute train and test MSE
    # - Append errors to respective lists

    # model_gauss = ...
    # model_gauss.fit(...)
    # y_train_pred = ...
    # y_test_pred = ...
    # train_mse = ...
    # test_mse = ...
    # gauss_train_errors.append(...)
    # gauss_test_errors.append(...)
    # ------------------------------------------------------------------------


% Plot comparison of test errors
plt.figure(figsize=(10,6))
plt.plot(degrees, poly_test_errors, 'ro-', label='Polynomial Regression Test MSE')
plt.plot(gaussian_degrees, gauss_test_errors, 'bo-', label='Gaussian Basis Test
    MSE')
plt.xlabel('Model Complexity (Degree or Number of Basis Functions)')
plt.ylabel('Test Mean Squared Error')
plt.title('Comparison of Basis Functions on Concrete Data')
plt.legend()
plt.show()
```

Listing 1: Python Code for Part 2

3 Submission Guidelines for Part 2

- Submit a complete Jupyter Notebook containing your code, plots, and experimental results. Ensure the notebook runs end-to-end with a fixed random seed (where applicable) for reproducibility.
- Provide a written report that includes answers to the analysis questions, along with a thoughtful reflection on your work for Part 2.
- Clearly document every instance of AI assistance used (if any). Include which tool was used, what you asked, and how you integrated the response.

Good luck and happy coding!