AI 600 - Deep Learning

Assignment 0 - Introduction

31 January 2026

Guidelines

This assignment will not be submitted or graded. However, we strongly encourage you to try and complete it as it will help you set up (or revisit and revise) the foundations of conventional machine learning models. These foundations will be highly beneficial throughout the AI-600 Deep Learning course.

Please note the following requirements carefully:

- **Policy on Usage of Generative AI Tools:** Students are most welcome to use generative AI tools as partners in their learning journey. However, these tools should not be blindly trusted, and it is important to rely on your own understanding before finalizing any solution or code.

Task 1: Linear, Polynomial, and Kernel-Based Regression

Dataset Description

The dataset ***data-a1.csv*** we are working with is a collection of used car listings from **PakWheels**, a popular online marketplace for buying and selling cars in Pakistan. The dataset contains various attributes of the cars listed for sale, including:

- **make**: The manufacturer of the car (e.g., Toyota, Honda).
- **model**: The specific model of the car (e.g., Corolla, Civic).
- **year**: The year the car was manufactured.
- **engine**: The engine capacity of the car in cubic centimeters (cc).
- **mileage**: The total distance the car has traveled, typically measured in kilometers.
- **price**: The listed price of the car.
- **fuel**: The type of fuel the car uses (e.g., Petrol, Diesel, Hybrid).
- **transmission**: The type of transmission (e.g., Manual, Automatic).
- **assembly**: Whether the car is locally assembled or imported.
- **color**: The color of the car.
- **body**: The body type of the car (e.g., Sedan, SUV).
- **city**: The city where the car is listed for sale.
- **registered**: Whether the car is registered or not.

Exercise Objective

In this task, you will predict the **price** of a car using the remaining attributes as input features. The goal is not only to build predictive models, but also to understand how different modeling choices (linear vs. non-linear) affect learning behavior, generalization, and interpretability.

Data Cleaning and Preprocessing

- Handle missing values by imputing appropriate defaults or dropping rows/columns with excessive missingness.
- Encode categorical variables (e.g., fuel, transmission, or any other data point that is suitable for it) into numerical representations suitable for regression models.
- Standardize numerical features and analyze why standardization becomes increasingly important for certain models.

Feature Engineering

- Create derived features (e.g., concatenate `make` and `model` into a single `make_model` feature, or propose a better feature construction strategy suited to this dataset).
- Apply one-hot encoding to categorical variables and analyze how this impacts model complexity and interpretability.

Model Training and Evaluation

- Split the dataset into training, validation, and testing sets.
- Implement a custom linear regression model from scratch, with or without L2 regularization (Ridge) - Your choice, whichever you feel provides better generalizability and why so?
- Train the model using gradient descent and analyze convergence behavior.

Polynomial and Kernel-Based Extensions

Linear regression assumes a linear relationship between features and the target variable. In this part, you will extend the same task to non-linear models and analyze the resulting behavior.

- Implement polynomial regression by expanding the input features to higher-degree polynomial terms. Experiment with different degrees (e.g., degree 2, 3, and higher where feasible) and observe how increasing the polynomial degree affects training error, validation error, and test error. Find out whether improvements in training performance translate to better generalization. Also, analyze any phenomena you observe, such as overfitting, exploding weights, sensitivity to noise, or numerical instability. Why do these effects become more pronounced as the degree increases?
- Implement a kernelized regression model (RBF/Gaussian; your choice) using a polynomial kernel. Compare its behavior with explicit polynomial feature expansion, and explain any similarities or differences you observe.
- For all non-linear models, focus on **what changes you observe** and **why these changes occur**, rather than only evaluating performance numbers.

Analysis and Interpretation

- Evaluate all models using Mean Squared Error (MSE) on training, validation, and test sets.
- Compare linear, polynomial, and kernel-based models in terms of generalization performance and robustness.

- Print and compare actual vs. predicted prices for a subset of examples across different models.
- Inspect learned weights and bias terms. Identify any unusual or unstable behavior when moving from linear to higher-degree or kernel-based models, and the potential causes.

By the end of this task, you will have hands-on experience with linear and non-linear regression models, understand the practical consequences of increasing model capacity, and develop intuition about when and why more complex models succeed or fail. This task provides substantial guidance; subsequent tasks will require greater independence and deeper experimentation to achieve strong performance.

Task 2: Naive Bayes Classifier

Dataset Description

Each row in the dataset corresponds to the clinical attributes of a patient. The target variable is a binary indicator representing the presence or absence of heart disease. The dataset consists of the following features:

- **age** (`age`): Age of the patient in years.
- **sex** (`sex`): Sex of the patient.
- **chest pain type** (`cp`): Type of chest pain experienced by the patient (4 categories).
- **resting blood pressure** (`trestbps`): Resting blood pressure measured in mm Hg upon hospital admission.
- **serum cholesterol** (`chol`): Serum cholesterol level measured in mg/dL.
- **fasting blood sugar** (`fbs`): Indicator of whether fasting blood sugar is greater than 120 mg/dL.
- **resting electrocardiographic results** (`restecg`): Resting ECG results (3 categories).
- **maximum heart rate achieved** (`thalach`): Maximum heart rate achieved during exercise.
- **exercise induced angina** (`exang`): Whether exercise induces angina (Yes/No).
- **oldpeak** (`oldpeak`): ST depression induced by exercise relative to rest.
- **slope** (`slope`): The slope of the peak exercise ST segment (3 categories).
- **number of major vessels** (`ca`): Number of major vessels (0–3) colored by fluoroscopy.
- **thal** (`thal`): Thalassemia status (3 categories).

In this task, you will implement a **Naive Bayes classifier** from scratch and apply it to the heart disease dataset. The objective is to understand the probabilistic foundations of Naive Bayes, analyze its assumptions, and evaluate its performance on medical data.

It is important to recognize that this dataset contains a mix of **categorical** and **continuous-valued** attributes (e.g., age, cholesterol, resting blood pressure). The simple discrete Naive Bayes formulation assumes features take values from a finite set. Therefore, prior to applying Naive Bayes, you may need to select a suitable approach, for example:

- Discretizing continuous variables into bins,
- Applying a Gaussian likelihood model for continuous features,
- Or using a hybrid approach that combines categorical and continuous assumptions.

Probabilistic Model

Given a feature vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, the posterior probability of class C_k is defined using Bayes' theorem as:

$$P(C_k \mid \mathbf{x}) = \frac{P(C_k) \prod_{i=1}^{n} P(x_i \mid C_k)}{P(\mathbf{x})}$$

Where:

- $P(C_k)$ is the prior probability of class C_k,
- $P(x_i \mid C_k)$ is the likelihood of observing feature x_i given class C_k,
- $\mathbf{x}$ represents the feature vector.

Since $P(\mathbf{x})$ is constant across classes, it can be ignored during classification.
To avoid zero probabilities, apply **Laplace smoothing** to the likelihood estimates:

$$P(x_i \mid C_k) = \frac{\text{count}(x_i, C_k) + \alpha}{\text{count}(C_k) + \alpha \cdot |X_i|}$$

Where:

- α is the smoothing parameter,
- $|X_i|$ is the number of possible values of feature x_i.

Fit the Naive Bayes Classifier

- Implement a Naive Bayes classifier from scratch using the probabilistic formulation above.
- Estimate class priors and feature likelihoods from the training data.
- Apply Laplace smoothing and analyze how different values of α affect the learned probabilities.

Model Evaluation

- Create a function to evaluate your classifier on the test split.
- Use the following metrics:
  - Accuracy
  - Precision
  - Recall
  - F1-score
- Analyze class-wise performance and see if any imbalance-related behavior you observe.

Error Analysis and Interpretation

- Identify examples where the model makes incorrect predictions and analyze which features contributed most to the error.
- See how the Naive Bayes independence assumption affects performance on this dataset.
- Explain situations where Naive Bayes performs surprisingly well despite its strong assumptions.

Benchmarking the Implementation

Library Implementation

- Train a Naive Bayes classifier using the appropriate implementation from `scikit-learn` on the same training data.
- Use the same preprocessing, encoding, and data splits as in your custom implementation.

Refer to the official `scikit-learn` documentation for correct usage.

Evaluating Library Implementation

- Predict the target labels on the test set using the library model.
- Compute accuracy, precision, recall, and F1-score.

Comparative Analysis

- Compare the performance of your custom Naive Bayes classifier with the library-based implementation.
- If results differ, analyze potential reasons such as numerical stability, smoothing implementation, or preprocessing differences.
- See the advantages and disadvantages of implementing machine learning algorithms from scratch versus using well-tested library implementations.