Linear Regression questions

August 12, 2025

Exercise 1 — Univariate Linear Regression

Problem Statement

You are given a dataset containing a single feature and a target value. Your task is to:

1. Perform basic **Exploratory Data Analysis (EDA)**.
2. Normalize the feature.
3. Train a **univariate linear regression model** using **gradient descent** (from scratch, no built-in linear regression functions).
4. Output the results in the **exact format** given below.
5. Predict the target values for feature values `150` and `200` (normalized using the dataset statistics).

Your code must **read the dataset from input** and print the results exactly as specified, with no extra text or formatting.

Essential Steps to Implement

1. **Read the dataset:**
    - First line: integer N, the number of rows.
    - Next N lines: two space-separated floats — feature and target.
2. **EDA:**
    - Print the **first 5 rows** of the dataset (or all rows if $N < 5$).
    - Print the **shape** of the dataset (N, d).
    - For each column, print: `mean std min max` (all rounded to 2 decimal places).
3. **Feature Normalization:**

- Normalize only the feature column to **zero mean** and **unit variance**.
- Keep the target values unchanged.

4. **Model Training:**

- Hypothesis:
$$\hat{y} = \theta_0 + \theta_1 x$$
- Initialize:
$$\theta_0 = 0, \quad \theta_1 = 0$$
- Learning rate:
$$\alpha = 0.01$$
- Epochs:
$$1000$$
- Mean Squared Error:
$$MSE = \frac{1}{2N} \sum_{i=1}^{N} \left(\hat{y}^{(i)} - y^{(i)} \right)^2$$
- Gradient Descent Update Rules:
$$\theta_0 \leftarrow \theta_0 - \alpha \cdot \frac{1}{N} \sum_{i=1}^{N} \left(\hat{y}^{(i)} - y^{(i)} \right)$$
$$\theta_1 \leftarrow \theta_1 - \alpha \cdot \frac{1}{N} \sum_{i=1}^{N} \left(\hat{y}^{(i)} - y^{(i)} \right) \cdot x^{(i)}$$
- Do **not** print per-epoch training logs. Only print final results.

5. **Final Output:**

- Print:

```
Final theta0=TH0 | theta1=TH1 | Final MSE=MSE
```

  where `TH0` and `TH1` are printed to 3 decimal places and `MSE` to 2 decimal places.
- Predict for feature values 150 and 200, printing each prediction on a new line, rounded to 2 decimal places.

Input Format

```
N
feature_1 target_1
feature_2 target_2
...
feature_N target_N
```

Output Format

1. First 5 rows of dataset (or all if N ¡ 5)
2. Shape (N,d)
3. Summary statistics for each column (mean, std, min, max) — two lines total
4. Final model parameters and MSE
5. Predictions for 150 and 200

Example Input

```
20
35 179
42 200
50 221
60 263
67 280
75 314
80 327
90 360
95 377
100 391
110 425
120 462
130 493
140 521
150 552
160 582
175 631
190 675
210 740
230 804
```

Example Output

```
35.0 179.0
```

```
42.0 200.0
50.0 221.0
60.0 263.0
67.0 280.0
(20,2)
115.45 55.22 35.00 230.00
439.85 177.34 179.00 804.00
Final theta0=439.831 | theta1=177.293 | Final MSE=7.58
550.77
711.31
```

Important Notes

- No extra text or headings — Moodle grading will fail if formatting is not exact.
- All rounding must be exactly as specified.
- You must implement gradient descent manually — do not use libraries like `sklearn` for training.

Exercise 2 — Multivariate Linear Regression

Problem Statement

You are given a dataset containing three features:

1. House size in m^2
2. Number of bedrooms
3. Age of the house in years

and a target value: price (in lakhs).

Your task is to:

1. Perform **basic EDA**.
2. Standardize the features.
3. Train a **multivariate linear regression model** using **vectorized gradient descent** (from scratch).
4. Verify your implementation using the **normal equation** and report the difference in Mean Squared Error (MSE).
5. Output results in the exact format described below.
6. Predict the prices for two new houses.

Essential Steps to Implement

1) Data Exploration (EDA)

- Print the **first 5 rows** of the dataset (or all rows if $N < 5$), space-separated.
- Print the **shape** of the dataset (N, d).
- For each column, print: `mean std min max` (rounded to 2 decimal places).

2) Standardization

- Standardize **only the features** (all three) to zero mean and unit variance.
- Keep the target column unchanged.

3) Model Definition

- Stack features into $X \in \mathbb{R}^{N \times d}$ (excluding the intercept).
- Augment X with a column of ones to form X'.
- Hypothesis:
$$\hat{y} = X'\theta'$$
- Mean Squared Error:
$$MSE = \frac{1}{2N}\|\hat{y} - y\|_2^2$$

4) Gradient Descent

- Initialize all parameters in θ' to 0.
- Learning rate:
$$\alpha = 0.01$$
- Number of epochs:
$$300$$
- Vectorized update rule:
$$\theta' \leftarrow \theta' - \alpha \cdot \frac{1}{N}(X')^\top(X'\theta' - y)$$
- Do **not** print per-epoch logs.

5) Verification via Normal Equation

- Compute:

$$\theta^* = \left((X')^\top X'\right)^{-1} (X')^\top y$$

- Compute MSE for θ' (GD) and θ^* (normal equation).
- Print the absolute difference between these MSE values, rounded to 5 decimal places.

6) Final Output

1. First 5 rows of dataset.
2. Shape (N, d).
3. Summary statistics for each column (mean, std, min, max) — 4 lines total.
4. `Final theta=[TH0, TH1, TH2, TH3]` with all parameters rounded to 3 decimal places.
5. `Final MSE=MSE` with MSE rounded to 2 decimal places.
6. `MSE Difference=DIFF` with DIFF rounded to 5 decimal places.
7. Predictions for:
   - House A: (150, 3, 5)
   - House B: (200, 4, 2)

   Each prediction on a new line, rounded to 2 decimal places.

Input Format

```
N
size_m2 bedrooms age_years price_lakhs
...
```

Example Input

```
20
35 1 20 179
42 2 15 200
50 2 18 221
60 3 10 263
67 3 8 280
75 3 12 314
80 4 5 327
90 4 5 360
95 4 6 377
```

```
100 5 5 391
110 5 3 425
120 5 2 462
130 6 2 493
140 6 1 521
150 6 1 552
160 7 1 582
175 7 2 631
190 8 2 675
210 8 1 740
230 9 1 804
```

Example Output

```
35.0 1.0 20.0 179.0
42.0 2.0 15.0 200.0
50.0 2.0 18.0 221.0
60.0 3.0 10.0 263.0
67.0 3.0 8.0 280.0
(20,4)
115.45 55.22 35.00 230.00
4.90 2.17 1.00 9.00
6.00 5.82 1.00 20.00
439.85 177.34 179.00 804.00
Final theta=[418.279, 82.598, 69.709, -25.244]
Final MSE=507.47
MSE Difference=503.76598
413.14
533.13
```

Important Notes

- No extra text or headings — Moodle grading will fail if formatting is not exact.
- All rounding must be exactly as specified.
- Implement gradient descent manually — no use of `sklearn` or other regression libraries.