New Uzbekistan University

Intro to Machine Learning

Week 2: Lab 2 — Linear Regression & Gradient Descent

Dataset: USA Housing - Each row represents a house in the USA.

Dataset (CSV): https://www.kaggle.com/datasets/kanths028/usa-housing/data

Features (X):

- Avg. Area Income — Average income of residents in the area
- Avg. Area House Age — Average age of houses
- Avg. Area Number of Rooms — Average number of rooms
- Avg. Area Number of Bedrooms — Average number of bedrooms
- Area Population — Population of the area

Target (Y):

- Price — House price (continuous value)

This is a regression problem because we are predicting a number.

Part 1. Simple Linear Regression in WEKA

In this part, you will build a Simple Linear Regression model using only one feature to predict house prices. Predict Price using Avg. Area Income only.

Step 1. Load Dataset

1. Open Weka → Explorer
2. Go to Preprocess
3. Click Open file
4. Load *USA_Housing.csv*

Step 2. If WEKA Cannot Load the CSV Remove Non-Numeric Column

If WEKA shows an error (due to the Address column), clean the dataset using Python:

1. Open Google Colab (or Jupyter Notebook)
2. Run the code to remove the Address column and save a WEKA-friendly CSV. (refer to Lab 1)
3. Go back to WEKA and load the new file.

Step 3. Keep Only One Feature

Since we are doing Simple Linear Regression, we keep only:

- Avg. Area Income (feature X)
- Price (target Y)

Remove all other attributes:

1. Select each column except:
    - Avg. Area Income
    - Price
2. Click Remove

Now WEKA has only one input variable.

Step 4. Set Target Variable

At the bottom right set Class = Price

Step 5. Train Simple Linear Regression Model

- Go to the Classify tab
- Click Choose → functions → LinearRegression
- Under Test options, choose:
    - Cross-validation
    - 10 folds
- Click Start

Step 6. Analyze the Output

You will see:

- Correlation Coefficient → how well the model follows the trend
- Mean Absolute Error (MAE) → average prediction error
- Root Mean Squared Error (RMSE) → penalizes large errors more

Because we use only one feature, WEKA is fitting a straight line:

$$Price = w_0 + w_1 * (Avg.\ Area\ Income)$$

This is the “best line” discussed in the lecture, the one that minimizes the Residual Sum of Squares (RSS)

Part 2 — Linear Regression in Python

In this part, we will first train a simple linear regression model using a ready-made library (scikit-learn), and then implement the same model ourselves using Gradient Descent. **scikit-learn** is a popular Python library for machine learning that provides ready-to-use algorithms without implementing the math ourselves.

Task 1. Load the Dataset

1. Load the dataset using pandas.

2. Display the first 5 rows.

Task 2. Keep Only One Feature + Target

We will use only one feature for simple linear regression: *Avg. Area Income*

Task 3. Train/Test Split

Split the data into training and testing sets:

- Train set: used to fit (learn) the model - 80%
- Test set: used to evaluate performance on unseen data - 20%

Task 4. Train Linear Regression Model (Library Method)

Using *from sklearn.linear_model import LinearRegression*
This model automatically computes the best line.

Task 5. View the Learned Weights

Task 6. Make Predictions

Task 7. Plot Actual Data and Regression Line

Task 8. Model Evaluation

Evaluate regression performance using:

- MAE: average absolute error
- RMSE: penalizes larger errors
- R^2: how much variance is explained by the model

Part 3 — Implement Linear Regression Using Gradient Descent

Instead of using a library, we now calculate the weights **manually** using the Gradient Descent algorithm from the lecture.

Task 9. Define Prediction Function
Task 10. Define Gradient Descent Algorithm

Gradient Descent updates model weights step by step to minimize the prediction error.

Task 11. Train Model Using Gradient Descent
Task 12. Plot Gradient Descent Regression Line
Task 13. Plot Cost Decreasing Over Time
Task 14. Compare the Two Models

Discussion Questions

1. Are the weights from Gradient Descent close to the library model weights?
2. What happens if the learning rate is too large?
3. Why does the cost decrease over epochs?

Suggestions:

1. Try Different Learning Rates
2. Try Different Features
3. Try Different Train/Test Split