Assignment 05

Applied Machine Learning with Python

Fourth Year BS (Honors) 2023-2024

Course Title: Math Lab IV, Course Code: AMTH 450

Department of Applied Mathematics, University of Dhaka

Name: **Roll No.:** **Group:**

1. Linear Regression with One Variable

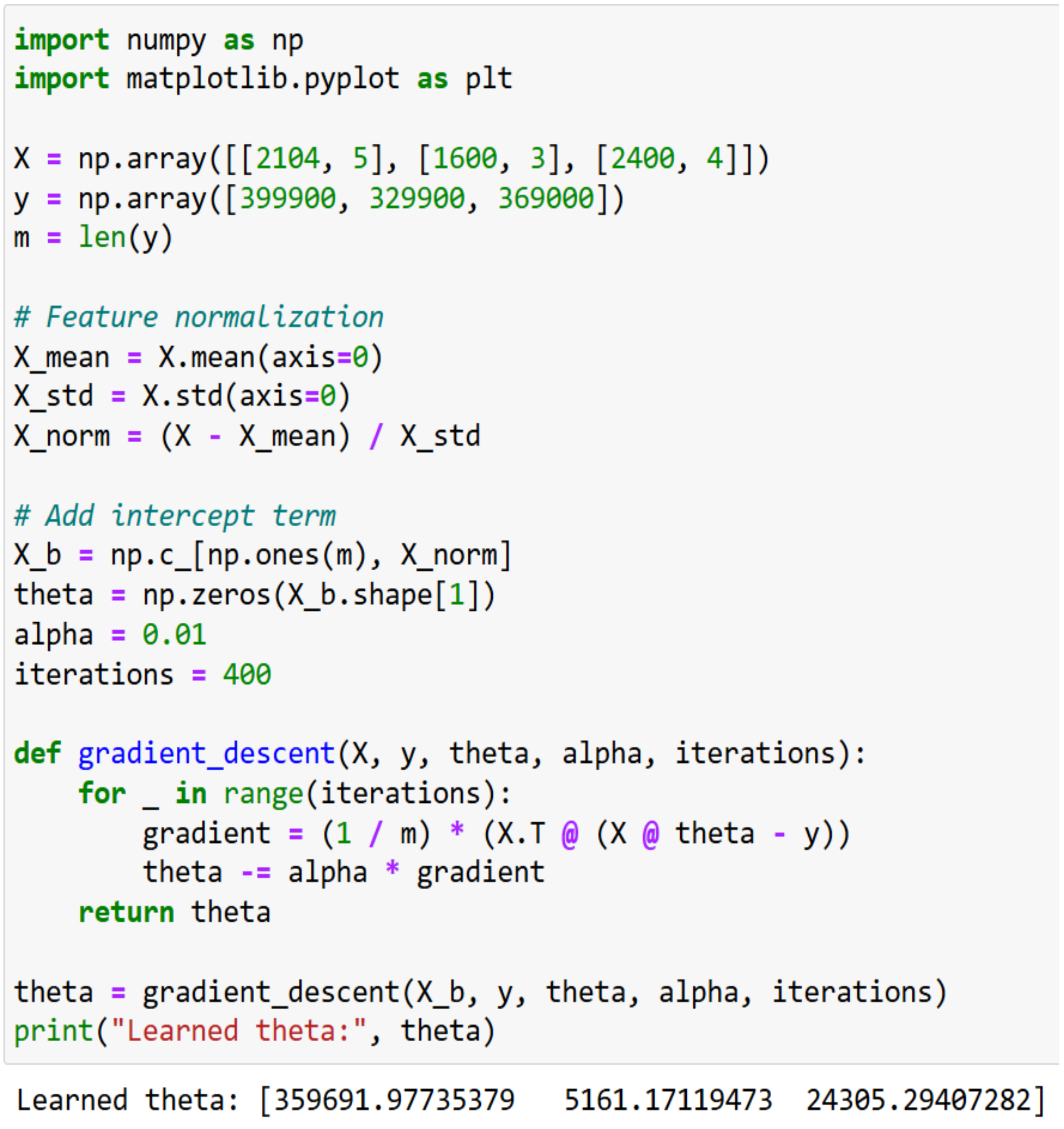
A sample code with the output is given as follows. Please modify the code to read the data file (ex2data1.txt) and produce the output learned theta.

```python
import numpy as np
import matplotlib.pyplot as plt

X = np.array([6.1101, 5.5277, 8.5186]) # Population (x)
y = np.array([17.592, 9.1302, 13.662]) # Profit (y)
m = len(y)

# Add intercept term
X_b = np.c_[np.ones(m), X]
theta = np.zeros(2)
alpha = 0.01
iterations = 1500

def compute_cost(X, y, theta):
    errors = X @ theta - y
    return (1 / (2 * m)) * np.dot(errors, errors)

def gradient_descent(X, y, theta, alpha, iterations):
    for _ in range(iterations):
        gradient = (1 / m) * (X.T @ (X @ theta - y))
        theta -= alpha * gradient
    return theta

theta = gradient_descent(X_b, y, theta, alpha, iterations)
print("Learned theta:", theta)
```

```
Learned theta: [4.02750103 1.37581788]
```

2. Multivariate Linear Regression

A sample code with the output is given as follows. Please modify the code to read the data file (ex2data1.txt) and produce the output learned theta.



```python
import numpy as np
import matplotlib.pyplot as plt

X = np.array([[2104, 5], [1600, 3], [2400, 4]])
y = np.array([399900, 329900, 369000])
m = len(y)

# Feature normalization
X_mean = X.mean(axis=0)
X_std = X.std(axis=0)
X_norm = (X - X_mean) / X_std

# Add intercept term
X_b = np.c_[np.ones(m), X_norm]
theta = np.zeros(X_b.shape[1])
alpha = 0.01
iterations = 400

def gradient_descent(X, y, theta, alpha, iterations):
    for _ in range(iterations):
        gradient = (1 / m) * (X.T @ (X @ theta - y))
        theta -= alpha * gradient
    return theta

theta = gradient_descent(X_b, y, theta, alpha, iterations)
print("Learned theta:", theta)
```

```
Learned theta: [359691.97735379   5161.17119473  24305.29407282]
```

3. Logistic Regression for Binary Classification

Implement logistic regression to predict whether a student is admitted based on exam scores.

Dataset: Generate random data sets containing scores of two exams and admission result (0 or 1).

Tasks:

- Visualize data using a scatter plot.
- Implement the sigmoid function.
- Implement the cost function for logistic regression.
- Implement gradient descent and find optimal θ.
- Plot the decision boundary.
- Evaluate accuracy on the training set.

4. Regularization in Logistic Regression

Apply regularization to logistic regression for non-linearly separable data.

Dataset: Generate microchip test results (2 test scores) with pass/fail labels.

Tasks:

- Map features into polynomial terms (e.g., up to 6th degree).
- Implement regularized cost function and gradient.
- Choose different values of regularization parameter λ (e.g., 0, 1, 100).
- Plot the decision boundaries.
- Discuss the effect of underfitting and overfitting due to different λ values.

5. Polynomial Regression and Learning Curves

Implement polynomial regression to capture non-linear trends in data.

Dataset: Generate or use a dataset where the relationship between input and output is non-linear (e.g., housing prices vs. size, $y = x\, sin(x)$, etc.).

Tasks:
● Map features to polynomial terms (e.g., degree 5 or higher).
● Implement regularized linear regression.
● Plot training and cross-validation error (learning curves) for different training set sizes.
● Vary regularization parameter λ and observe the effect on bias-variance.
● Discuss bias-variance trade-off using plots and numerical outputs.

Expected Output:
● Learned parameters (θ).
● Learning curves (training error vs. cross-validation error).
● Plot showing polynomial regression fit.
● Brief explanation on how λ and training set size affect model performance.

6. One-vs-All Logistic Regression for Multi-class Classification

Extend logistic regression to handle multi-class classification using the One-vs-All (OvA) approach.

Dataset: Use the ***digits_data*** dataset (from the digit classification exercise) or simulate your own data with 3+ classes (e.g., flower species, handwritten digits, etc.).

Tasks:
● Implement the one-vs-all classification strategy.
● Train one logistic regression classifier per class.
● Predict the class label for new inputs using all trained classifiers.
● Compute and report the accuracy on the training set.
● Visualize data samples (e.g., digits or class distributions).
● Optional: Use scikit-learn's LogisticRegression for comparison.

Expected Output:
● Training accuracy.
● Learned parameters for each class.
● Plot(s) if using a visual dataset.