**Assignment 4: Maximum Likelihood Estimation with Features**

**Instructions:** This assignment engages you in the practical application of maximum likelihood estimation with features for linear regression modeling. Follow the steps carefully, and submit your work as a single PDF file that includes all required code and graphs. **Attach** supplementary code files (.py and .ipynb) as used in this assignment.

**Objective:** Learn how to fit a linear regression model using maximum likelihood estimation with features polynomial and understand the effects of model adjustments, such as adding a degree of polynomial.

# Step-by-Step Instructions

## Q1. Maximum Likelihood Estimation with Features

### Step 1: Understanding and Plotting the Data

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

# Data Initialization
X = np.array([-4.5, -3.5, -3, -1.8, -0.2, 0.3, 1.3, 2.6, 3.8, 4.8]).reshape(-1,1)
y = np.array([
    [-0.91650116],
    [-0.47546053],
    [-0.10972425],
    [0.29504095],
    [-0.01596218],
    [0.10014949],
    [0.48104303],
    [0.10979023],
    [-0.99742128],
    [-0.91221826]
]).reshape(-1,1)

X_test = np.array([-3.99, -1.38, -1.37, -0.94,
  0.69,  1.4,   1.57,  1.78,  1.81,  4.89]).reshape(-1,1)
y_test = np.array([
    [-0.80737607],
```

```
    [0.19813376],
    [0.19537639],
    [0.07185977],
    [0.24954213],
    [0.50662504],
    [0.52943298],
    [0.52406997],
    [0.51999057],
    [-0.82318288]
]).reshape(-1,1)
```

- Using the provided code, plot the relationship between variables `x` and `y`.
- Attach the Python code used to generate the graph in your PDF.
- Display the graph you obtain.

## Step 2: Polynomial Feature Transformation

- Implement the poly_features function to transform input data X into polynomial features of degree K.

**(Polynomial Regression)**

$$\phi(x) = \begin{bmatrix} \phi_0(x) \\ \phi_1(x) \\ \vdots \\ \phi_{K-1}(x) \end{bmatrix} = \begin{bmatrix} 1 \\ x \\ x^2 \\ x^3 \\ \vdots \\ x^{K-1} \end{bmatrix}$$

**(Feature Matrix for Second-order Polynomials)**

$$\mathbf{\Phi} = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \vdots & \vdots & \vdots \\ 1 & x_N & x_N^2 \end{bmatrix}$$

- Attach the Python code used for the implementation.
- Explain the purpose and the effect of this transformation.

**Step 3: Fitting the Model Using Maximum Likelihood**

- Using the poly_features function from Step 2, transform both training and test datasets for a degree of polynomial 4.
- Apply the nonlinear_features_maximum_likelihood function to estimate model parameters (Theta) with Phi from training data.

$$\theta_{\mathrm{ML}} = (\Phi^\top \Phi)^{-1} \Phi^\top y$$

- Attach the Python code used for the implementation and show their result.

**Step 4: Model Evaluation Using RMSE**

- Implement the RMSE function to evaluate the accuracy of your model on both the training and test datasets.
- Calculate RMSE for models with polynomial degrees ranging from 0 to 15.
- Plot these RMSE values against the polynomial degree for both training and test datasets.
- Attach the Python code used for the implementation and generate the graph in your PDF.

**Step 5: Selecting the Best Model**

- Identify the polynomial degree that results in the lowest RMSE for the test dataset.
- Using the optimal degree, plot the model's predictions against the original data points.

```
X_test_all = np.linspace(-5, 5, 100).reshape(-1,1)
Phi_test_all = poly_features(X_test_all, optimal_k)
ypred_test_all = Phi_test_all @ theta_optimal
```

- Discuss why this particular model degree was optimal and how model complexity affects overfitting and underfitting.
- Attach the Python code used for the implementation and generate the graph in your PDF.

**Step 6: Conclusion**

- Summarize your findings regarding the polynomial regression model's performance.
- Discuss the trade-offs between model complexity (degree of the polynomial) and prediction accuracy.

# Q2. K-means

In this assignment, you will use a dataset from Kaggle to perform K-means clustering, commonly used for grouping similar data points together. This task will involve several clear steps:

**Step 1: Prepare Your Dataset**

- Choose a dataset from Kaggle that's good for grouping, like the Mall Customers Dataset (link)( https://www.kaggle.com/vjchoudhary7/customer-segmentation-tutorial-in-python). Clean up the dataset by removing any missing or unnecessary data.
- **Attach your data cleaning code** with your report.

**Step 2: Visualize Data Before Clustering**

- Create a graph to show how your data looks before you apply clustering. This helps you see what the data looks like normally.
- **Attach your plotting code** with your report.

**Step 3: Implement K-means Clustering and Determine Optimal Clusters**

- Use the K-means algorithm to find groups in your data. To figure out the best number of groups, use the elbow method: plot the within-cluster sum of squares (WCSS) against different numbers of clusters and look for the point where the line starts to flatten out.
- **Attach your K-means and elbow method code** with your report.

**Elbow Method Example Code:**

==========

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs

# Generate synthetic data
X, _ = make_blobs(n_samples=300, centers=4, cluster_std=0.60, random_state=0)

# Calculate WCSS for different numbers of clusters
```

```
wcss = []
for i in range(1, 11):  # Testing 1 to 10 clusters
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10,
random_state=0)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)  # inertia_ is the WCSS for the model

# Plotting the WCSS to observe the 'Elbow'
plt.figure(figsize=(10, 8))
plt.plot(range(1, 11), wcss, marker='o')
plt.title('Elbow Method For Optimal k')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.grid(True)
plt.show()
```

==========

### Step 4: Visualize Data After Clustering

- Make another graph to show your data with the groups found by K-means. This shows how the algorithm has organized the data.
- **Attach your post-clustering plotting code** with your report.

### Step 5: Interpret Your Results

- Explain what the results from the K-means clustering show about your data. Discuss how this information can help identify specific groups within the dataset.
- **Attach your interpretation and any additional code** you used to analyze the results.

**Submission Requirements:**

- Submit your completed assignment as a PDF containing all necessary code snippets and graphs.
- Attach corresponding .py and .ipynb files containing executable code.
- Ensure all files are clearly labeled and organized.
- Late submissions will be subject to deductions according to the policy.

**Evaluation Criteria:** Your assignment will be graded based on the accuracy of your implementation, the clarity of your presentation, and the completeness of your submission. Specific attention will be given to how well you follow these guidelines:

- **Step-by-Step Explanation:** This assignment requires clear, step-by-step explanations. You must explicitly label and explain each step (e.g., Step 1, Step 2, Step 3, Step 4). Failure to provide detailed explanations for each required step will result in a proportional deduction in your score. For example, if the assignment requires four steps and you complete only three, this will result in a score of 75 out of 100.