Homework #2 | Advanced Machine Learning (MLR 570)
Due Date: 28/10/2024 | Dr. Mohamed AlHajri

Please show your calculations and don't provide only final answers

Question 1: Least Square *(Each question is 10 points)* *30 points*

Dataset Generation Code:

```
import numpy as np

import matplotlib.pyplot as plt

from numpy.linalg import inv


# Seed for reproducibility

np.random.seed(42)


# Generate independent variable (X)

X = np.linspace(1, 10, 100)

X = X[:, np.newaxis]  # Reshape X to be a 2D array (100x1)


true_beta = 3

y = true_beta * X.squeeze() + np.random.normal(0, 1, size=100)


# Add bias term (intercept) to X (i.e., X0 = 1)

X_b = np.c_[np.ones((100, 1)), X]  # X_b is now (100x2)
```

1. Derive the Least Squares algorithm, starting from the objective function and leading to the closed-form solution.
2. Discuss the advantages and limitations of the Least Squares method. What is the computational complexity?
3. Implement Least Squares using the derived equations on the provided dataset. Compare your results with sklearn's implementation.

Question 2: Weighted Least Square *(Each question is 10 points)* *30 points*

Dataset Generation Code:

```
import numpy as np

import matplotlib.pyplot as plt

from numpy.linalg import inv


# Seed for reproducibility

np.random.seed(42)


# Generate independent variable (X)

X = np.linspace(1, 10, 100)

X = X[:, np.newaxis]  # Reshape X to be a 2D array (100x1)


# Generate heteroscedastic data with extremely increased variance

true_beta = 3

y = true_beta * X.squeeze() + np.random.normal(0, X.squeeze()**4, size=100)


# Add bias term (intercept) to X (i.e., X0 = 1)

X_b = np.c_[np.ones((100, 1)), X]  # X_b is now (100x2)
```

1. Derive the Weighted Least Squares algorithm, starting from the objective function and leading to the closed-form solution.
2. Discuss the advantages and limitations of the Weighted Least Squares method. What is the computational complexity?
3. Implement Weighted Least Squares using the derived equations on the provided dataset. Compare your results with sklearn's implementation.

Question 3: Kernel Least Square *(Each question is 10 points)* *30 points*

Dataset Generation Code:

```
import numpy as np

import matplotlib.pyplot as plt

from sklearn.kernel_approximation import RBFSampler

from sklearn.kernel_ridge import KernelRidge


# Generate a non-linear dataset

np.random.seed(42)

X = np.linspace(0, 10, 100).reshape(-1, 1)  # Features

y = np.sin(X).ravel() + 0.1 * np.random.randn(100)  # Non-linear target with noise


# Augment the features matrix for Least Squares (add bias term)

X_aug = np.hstack([np.ones((X.shape[0], 1)), X])  # Add a bias term (column of ones)
```

1. Derive the kernel least square, starting from the objective function and leading to the closed-form solution.
2. Discuss the advantages and limitations of the kernel least square. What is the computational complexity?
3. Implement kernel least square using the derived equations on the provided dataset. Compare your results with sklearn's implementation.

Question 4: Kernel Ridge Regression *(Each question is 10 points)* *30 points*

Dataset Generation Code:

```
import numpy as np

import matplotlib.pyplot as plt

from sklearn.kernel_approximation import RBFSampler

from sklearn.kernel_ridge import KernelRidge


# Generate a non-linear dataset

np.random.seed(42)

X = np.linspace(0, 10, 100).reshape(-1, 1)  # Features

y = np.sin(X).ravel() + 0.1 * np.random.randn(100)  # Non-linear target with noise


# Augment the features matrix for Least Squares (add bias term)

X_aug = np.hstack([np.ones((X.shape[0], 1)), X])  # Add a bias term (column of ones)
```

1. Derive the kernel ridge regression, starting from the objective function and leading to the closed-form solution.
2. Discuss the advantages and limitations of the kernel ridge regression. What is the computational complexity?
3. Implement kernel ridge regression using the derived equations on the provided dataset. Compare your results with sklearn's implementation.

Question 5: Perceptron & SVM Algorithm *(Each question is 10 points)* *40 points*

Dataset:

```
X_perceptron = np.array([[2, 3], [3, 3], [4, 5], [1, 0], [2, 1], [1, 1]])

y_perceptron = np.array([1, 1, 1, -1, -1, -1])
```

1. Implement the Perceptron algorithm by hand using the dataset provided. Show each step, including the weight updates, and explain the results.

Download the provided dataset **X_.npy** and **y_.npy**

2. Implement the Perceptron algorithm in Python. Report the accuracy and the number of iterations required for convergence. (The number of iterations refers to how many times the algorithm checks the perceptron condition before it converges.)
3. Implement modifications to the Perceptron algorithm, such as adjustments to the learning rate or weight initialization, to reduce the number of iterations below 1000.

Download the provided dataset **X_train.npy, X_test.npy, y_train.npy, y_test.npy**

4. Implement the perceptron algorithm and support vector machine in python. Report the accuracy on the training and test data of the perceptron and svm algorithm. Draw the decision boundary and mention why there are differences between the perceptron and svm.

Question 6: k-Nearest Neighbor (k-NN) [Classification] *(Each question is 10 points)* *30 points*

Dataset:

```
X = np.array([[2, 3], [1, 1], [4, 4], [5, 5], [6, 7], [7, 8]])

y = np.array([1, 1, -1, -1, -1, 1])
```

1. Predict the class of the point (3, 4) using k-NN for k=1,3,5 by calculating the Euclidean distance manually and performing majority voting.
2. Implement the k-NN algorithm on the dataset above in python using scikit-learn for k=1,3,5
3. Implement the k-NN algorithm on the iris dataset for k=1,3,5

```
iris = load_iris()

X = iris.data

y = iris.target # Split the data into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

Question 7: k-Nearest Neighbor (k-NN) [Regression] *20 points*

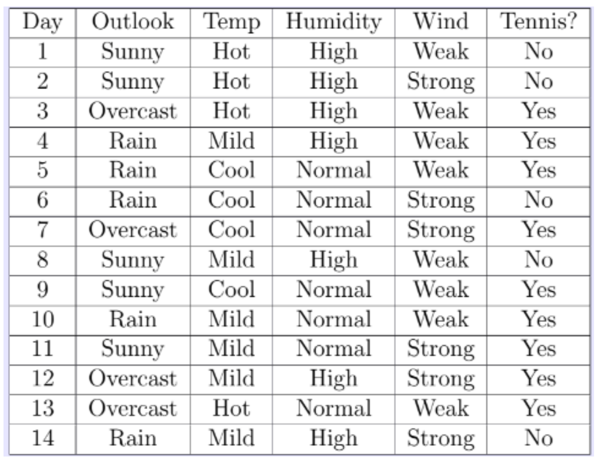
Dataset:

```
np.random.seed(42)

n_samples = 1000

X_car_travel = np.random.rand(n_samples, 3) * 100  # Features: traffic density, road type, and distance

y_car_travel = (X_car_travel[:, 0] * 0.5 + X_car_travel[:, 1] * 0.3 + X_car_travel[:, 2] * 1.2) +
np.random.randn(n_samples) * 5  # Simulated travel time



# Standardize the features

X_car_travel_scaled = scaler.fit_transform(X_car_travel)



# Split the data into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X_car_travel_scaled, y_car_travel, test_size=0.2,
random_state=42)
```

1. Implement k-NN regression in Python and explore the impact of different values of k (e.g., k = 1, 3, 5, 7) on performance. In addition to using the arithmetic mean to predict the regression output, also implement the geometric and harmonic means as alternative aggregation methods. Compare the performance of the three means (MSE, R^2) across different values of k by evaluating the model on the dataset provided. Include an analysis of when one mean might outperform the others. Provide both numerical results and visualizations to support your findings.
2. (Bonus) Explore 5 different real datasets and compare the performance of the arithmetic, geometric and harmonic mean in terms of the MSE and R^2. Please provide an explanation of when does the geometric or harmonic mean provide a better performance. *(10 points)*

Question 8: Decision Tree *(Each question is 10 points)* *30 points*

Dataset:



Day	Outlook	Temp	Humidity	Wind	Tennis?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

1. Construct the decision tree using the information gain and show your detailed calculations.
2. Construct the decision tree using the gini coefficient and show your detailed calculations.
3. In this question, you will implement three decision tree algorithms and compare their performance on the Yeast dataset:
    a. **ID3 (Information Gain)** – Implemented using sklearn based on **entropy** (information gain).
    b. **CART (Classification and Regression Trees)** – Implemented using sklearn based on **Gini coefficient**.
    c. **Decision Tree with Tsallis Entropy** – A custom decision tree where the split is based on **Tsallis entropy**, with a tunable parameter q. Explore different values of q and report the best performance for q in the range of [0.3,8].

```
from sklearn.datasets import fetch_openml

from sklearn.model_selection import train_test_split

import pandas as pd

# Load the yeast dataset from OpenML

yeast_data = fetch_openml(name="yeast", version=1)

X = pd.DataFrame(yeast_data.data, columns=yeast_data.feature_names)

y = yeast_data.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

4. (Bonus) **Optimal q Selection**: Propose and implement a method for selecting the optimal value of q based on the dataset properties and explore it for 10 different datasets. *(10 points)*