Assignment #3: Machine Learning

Total score: 100 + (bonus 30)

This assignment can be completed individually or by a team of a maximum of three members.

Due date: See the course page

1. Supervised learning for linear regression [40 points]

Write a Python program "**linear_reg.py**" that

(1) Loads a training dataset

X = [-3.0, -2.5, -2.0, -1.5, -1.0, 0.0, 1.0, 1.5, 2.0, 2.5, 3.0]

Y = [17.5, 12.9, 9.5, 7.2, 5.8, 5.5, 7.1, 9.7, 13.5, 18.4, 24.4],

(2) Displays a scatter plot for the dataset using "**matplotlib**" package for data visualization,

(3) Finds the function from the dataset using the "**LinearRegression()**" method in **scikit-learn** package. In supervised machine learning, the process of finding a function from a training dataset is called "**training**" and the entire process for analyzing a dataset and understanding the dataset by finding a function is called "**linear regression**". Linear regression is one of two common supervised learning methods. The function learned from the training can be used to predict $\hat{y}$ for any given x value if the function is reasonably accurate.

(4) Displays the coefficient vector of the equation $[w_0, w_1,\ldots,w_n]$ and writes the function in the form of a polynomial equation $y = w_0 + w_1x_1 + \cdots + w_nx_n$.

(5) Computes the training error using the Rooted Mean Squared Error $\boldsymbol{RMSE} = \sqrt{\frac{\sum_i (y_i - \hat{y}_i)^2}{N}}$ where y_i is actual y value in the training dataset, $\hat{y}_i$ is the predicted y value for each x value, and N is the total number of examples in the dataset.

(6) Generate five random numbers as a new data X in the range of [-5, 5], then predict the corresponding $\hat{y}$ for each new x value. Compute prediction errors using RMSE and briefly discuss how well the learned function fits the training dataset and whether or not the predicted values are acceptable.

(7) Load the dataset from "GasProperties.csv", split the dataset into training set (80%) and testing set (20%) and perform the linear regression to find the coefficients using the least square method $\hat{\boldsymbol{w}} = (\boldsymbol{X^T X})^{-1}\boldsymbol{X^T y}$. Note that this equation requires matrix transpose, inverse, and multiplication. This is a solution to the optimization problem:

$argmin_w J(w) = \frac{1}{N}\sum_i (y_i - w^T x_i)^2$. Compute the RMSE of the function for both training and testing errors.

2. **Supervised learning for classification** [30 points]

Write a Python program "**mlp_classifier.py**" that:

(1) Loads the "Iris flower" dataset directly from "**sklearn.datasets**" package.
The "**Iris flower**" dataset (or Fisher's Iris dataset) is a multivariate dataset used for testing the performance of many supervised learning algorithms. For more information about the dataset, refer to https://en.wikipedia.org/wiki/Iris_flower_data_set/ and the data repository (https://archive.ics.uci.edu/dataset/53/iris/).

(2) Splits 80% of the original dataset into training dataset and 20% into test dataset (20%).

(3) Train the decision tree algorithm to classify the dataset using a simple **MLPClassifier** in scikit-learn with only one hidden layer with 3 neurons but any other hyperparameters (e.g., activation functions, learning_rate, max_iter, batch_size, momentum, etc.) Compute the % accuracy for both training and testing error.

(4) Adjusts or changes any MLP hyperparameters (except for adding more layers or neurons) to enhance classification performance. Briefly explain if you were you able to improve the classification performance.

(5) Increase the network complexity by adding more layers and/or neurons until the new architecture shows the improved accuracy. Briefly explain if you were you able to improve the classification performance.

3. **Unsupervised learning for clustering** [30 points]

Write a Python program "**kmeans_clustering.py**" that

(1) Clusters the "**Iris flower**" dataset using the **K-means** with K=3 using **KMeans** in scikit-learn. Ignore the label column of the dataset for clustering since clustering is an unsupervised learning technique and does not require labels. Computes the **RMSE** for each cluster based on its centroids.

(2) Compares the clustering results from K-means with the classification results from "mlp_classifier.py", focusing on how well the clusters align with the actual classes in the

dataset. This evaluation process, known as **cluster quality validation**, helps determine whether data points within each cluster are similar to those in the actual dataset classes.

4. **Reinforcement learning: Agent navigation a Gridworld** [Bonus 30 points]

You are asked to complete the Python program "gridworld_policy_iteration.py" (posted on the course page), which simulates an agent navigating a Gridworld environment using **Policy Iteration**. The environment contains:

- A designated start state (red outline).
- Terminal/goal states (green outline).
- Obstacles (**black tiles**) that the agent cannot pass through.
- Reward flags (**gold tiles**) that give additional rewards.
- A visual display showing the grid and the value function heatmap, where blue shading represents the value function (**darker** = higher value).

Complete the following missing tasks in the program to make it fully functional and interactive:

(a) **Policy Evaluation**: Implement one sweep of policy evaluation, computing the value function $V(s)$ given a policy π.

(b) **Policy Improvement**: Update the policy π greedily for improvement based on the current value function $V(s)$.

(c) **Policy Iteration Loop**: Combine policy evaluation and improvement into a loop that repeatedly evaluates and improves the policy until it becomes stable.

What to submit

- A report file in word or PDF format that includes the name or names of the team members, and the % contribution of each member. If there is no agreement on individual contributions, briefly describe the tasks assigned and completed by each member. Different scores may be awarded based on individual contributions. If all members contributed equally, simply state "equal contribution." The report should also include descriptions of the answers (if questions are asked) and a portion of the screenshot demonstrating the functionality of each program.
- Each program file individually. DO NOT submit any zip file as Canvas cannot open them.
- Submit only one report file and the program file(s) for the entire team.

Grading criteria

- Does each program successfully perform the required tasks and generate the correct results?
- Does the report demonstrate that the team has a clear understanding of the relevant concepts, programming techniques, functional requirements?
- Does it show how the team applied this knowledge to successfully complete the tasks?
- Is the effort put into the project clearly reflected in both the report and the program files?