CS4013/5013: Assignment 5

Fall 2025

In this assignment, we will implement a few machine learning techniques in Python from scratch. Note that you cannot use existing implementations of these techniques e.g., the Python Scikit-Learn library includes a least square regression technique that we can directly use as follows. The above practice is not allowed. Instead, you need to implement the details of 'model.fit',

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(sample_train,label_train)
label_pred = model.predict(sample_test)
er = 1 - accuracy_score(label_test, label_pred)
```

'model.predict', etc. You can use advanced math functions though, such as those for computing matrix multiplication, matrix inverse, matrix decomposition and distance functions.

Task 1. Implement Linear Regression (Programming Task)

In Topic 1, a linear regression model and an iterative learning algorithm are introduced. Implement this algorithm and use it to learn a linear regression model. Evaluate model's prediction performance (in terms of mean square error) versus the number of iterations, and plot the results as a curve in Figure 1 (x-axis is the number of updates and y-axis is mean square error). A template is given to you. Implement and evaluate your algorithm based on it.

Goal: your curve should decrease and converges at some point.

Task 2. Implement Logistic Regression (Programming Task)

In Topic 1, a logistic regression model and iterative learning algorithm are introduced. Implement this algorithm and use it to learn a logistic regression model. Evaluate model's prediction performance (in terms of classification error) versus the number of iterations, and plot the results as a curve in Figure 2 (x-axis is the number of updates and y-axis is classification error). A template is given to you. Implement and evaluate your algorithm based on it.

Goal: your curve should decrease and converges at some point.

Task 3. Implement Kmeans Clustering

In Topic 2, a Kmeans clustering algorithm is introduced. Implement it and plot the clustering results in a two-dimensional or three dimensional space, where points assigned to the same cluster have the same color and points assigned to different clusters have different colors.

Note: To plot the clustering results in a 2D/3D space, we need to first reduce the feature number of data points to 2/3. In the given template, feature reduction has been done for you (using PCA). Treat the reduced 2/3 features as the coordinates of the plotted 2D/3D space.

You need to generate three figures.

– Figure 3 shows clustering results for $k = 2$ in a 2D space.

– Figure 4 shows clustering results for $k = 3$ in a 2D space.

– Figure 5 shows clustering results for $k = 2$ or $k = 3$ in a 3D space.

Task 4. Implement kNN Classification (Programming Task)

In Topic 2, a k-Nearest Neighbor classification algorithm is introduced. Implement it. Evaluate its prediction performance (in terms of classification error) versus k, and plot the results as a curve in Figure 6 (x-axis is k and y-axis is classification error). A template is given to you. Implement and evaluate your algorithm based on it.

Goal: You should aim to see higher error when k is too small or too large.

Task 5. Decision Tree (Theory Task)

In Topic 3, a decision tree model and learning algorithm is introduced. Let us practice its learning process in theory. The following table contains the records of ten students, described by five features (F1, F2, F3, F4, F5) and one binary label 'Grade'. Suppose we want to learn a decision tree model from this table. Your task is to identify the best feature to split root node. You need to elaborate arguments, especially how to estimate the label entropy of split child nodes based on each candidate feature.

Note: you are encouraged to write all mathematical arguments using Latex/Overleaf but it is also ok to hand write them and upload a scanned copy – however, the grader can take points off for unreadable writings.

Student ID	F1	F2	F3	F4	F5	Grade (A/B)
01	1	1	0	0	1	B
02	1	0	0	1	1	B
03	1	0	1	0	0	B
04	0	0	1	1	0	A
05	1	1	0	1	0	B
06	0	1	0	1	1	A
07	0	1	1	1	1	B
08	1	1	0	0	1	B
09	1	1	1	0	1	A
10	1	0	1	0	0	A

Task 6. Practice Random Forest (Programming Task)

In Topic 4, a random forest model is introduced. In this model, we know the number of decision trees m will impact the model's prediction performance (in terms of classification error).

Please empirically evaluate the impact of m on prediction model and plot the results as a curve in Figure 7 (x-axis is m and y-axis is classification error). You do not need to implement random forest from scratch. You can directly call a random forest function from scikit learn.

Goal: You should aim to see higher error when m is too small. Error may converge or bounce back when m becomes very large.

Task 7. Practice Ethical Machine Learning (Programming Task)

In Topic 5, we have studied the fairness problem in machine learning. You may also find more materials in some surveys such as

– M Hort et al. Bias mitigation for machine learning classifiers: A comprehensive survey, ACM Journal on Responsible Computing 2024.

– S Caton et al, Fairness in machine learning: a Survey, ACM Computing Surveys, 2024.

– D Pessach et al, A review on fairness in machine learning, ACM Computing Surveys 2022.

– N Mehrabi et al, A survey on bias and fairness in machine learning, ACM Computing Surveys 2021.

In the template, you are given a basic logistic regression model and its 'unfair' prediction across two groups. Implement a technique to improve prediction fairness (defined as the gap between two groups' errors). You can implement an existing technique or develop one by yourself.

Briefly explain your technique and report its classification errors in Table 1.

	Basic Logistic Regression	Your Method
Group 1 Error		
Group 2 Error		
Error Gap		

Table 1:

Submissions Instructions

You should generate 7 figures, 1 table and two sets of written answers (one for task 5 and one for task 7). You should place them all (sorted based on task number) in a pdf file named 'hw5.pdf' and upload it to Canvas through the submission page for hw5.

You also need to upload the code that generate the figures and table, including

– hw5_task1.py for Figure 1

– hw5_task2.py for Figure 2

– hw5_task3.py for Table 3/4/5 (set it to figure 4)

– hw5_task4.py for Figure 6

– hw5_task6.py for Figure 7

– hw5_task7.py for Table 1

You can generate the pdf file using any tool, although you are encouraged to generate it using Overleaf. A latex template 'hw5_Latex.txt' will be provided to you.