Module: Internet of Things **2024/25**

Lab Session 9: Machine Learning

Lecturers: Dr Mohammad Shojafar, Dr Chuan H Foh, Dr Ahmed Elzanaty

9.1 Introduction

Machine Learning (ML) is revolutionizing the Internet of Things (IoT) by enabling devices to learn from data, make intelligent decisions, and adapt to changing conditions in real-time. At the intersection of these two powerful technologies, ML enhances the capabilities of IoT devices, allowing them to process vast amounts of data generated by sensors, devices, and systems. By leveraging machine learning, IoT devices can predict outcomes, optimize operations, and provide insights without human intervention. This synergy is driving advancements in industries such as healthcare, smart cities, manufacturing, and agriculture, where IoT devices are increasingly becoming smarter and more autonomous, leading to more efficient and effective solutions.

However, IoT devices often face limitations such as constrained processing power, limited memory, and low energy capacity, which restrict their ability to run complex machine learning models. As a result, there is a growing need for simple, efficient models that can operate within these constraints while still providing valuable insights and predictions. Lightweight models ensure that IoT devices can function effectively in real-time scenarios without draining resources, making it possible to deploy machine learning solutions even in resource-limited environments.

In this experiment, you will practice using some simple ML models to classify IoT network traffic and detect any malicious actions.

Equipment and materials:

- ❑ Lab PC or your own computer
- ❑ Jupyter Notebook and Python development environment
- ❑ Sample codes available on instructor's github

9.2 Learning Objectives

We expect you to know the following background:

- How to use Jupyter Notebook, including editing codes, running cells, troubleshooting codes and environment setting;
- How to write codes in Python and use various ML related packages such as numpy, scikit-learn, pandas, and torch;

In this lab session, you will learn the following:

- How to preprocess dataset and its importance;
- How to analyze the ML outputs;
- How to tune ML models;

9.3 Ensemble Learners

Ensemble learners are a machine learning technique that combines multiple models, often referred to as "learners" or "base models" to improve the overall performance and accuracy of predictions. The idea behind ensemble learning is that by aggregating the predictions of several models, the ensemble can achieve better results than any individual model alone.

In this experiment, you shall try the following five models: Adaboost, LightGBM, Random Forest, CatBoost and XGBoost. We are using Bot-IoT dataset created by Koroniotis *et al.* The information about the dataset is given at https://ieee-dataport.org/documents/bot-iot-dataset.

*Note that understanding the dataset is unnecessary.

Please follow the steps below carefully to perform the experiments.

Step 1: Download the sample codes from github (https://github.com/cfoh)

- Open a Terminal and change directory to your Desktop by:
  - `cd ~/Desktop`
- In Desktop folder, download the sample codes by:
  - `git clone https://github.com/cfoh/IoT-Network-Intrusion-Detection-with-Ensemble-Learners.git`
- A folder named `IoT-Network-Intrusion-Detection-with-Ensemble-Learners` will be created under Desktop. This will be your working folder. Change directory to this folder by:
  - `cd IoT-Network-Intrusion-Detection-with-Ensemble-Learners`

Step 2: Unpack the dataset in the working folder

- In the working folder, unzip the dataset files by:
  - `unzip "*.zip"`
- Launch Jupyter Notebook, navigate to the working folder and open `CombineData.ipynb`.
- Run `CombineData.ipynb` notebook. Several notes:
  - You can do this cell by cell or simply run all cells.
  - If you see "`ModuleNotFoundError: no module named 'pandas'`" error, then install the missing package inside Jupyter Notebook by
    - Create a blank cell on top of the file
    - Type `` `pip install pandas` `` in the top cell
    - Run the cell to install the missing package `pandas`
    - The installation may take a while, it is important to wait until the execution is completed.
    - [optional] You may choose to remove the top cell as the missing package is already installed in your environment. While rerunning the installation again will not destroy the environment, it will waste computing resources.
    - Now, run the remaining cells.
  - After the execution, you should see the dataset `MergedData.csv` in the folder with size about 590MB.

Step 3: Run a model, say `RandomForest.ipynb`

- Load `RandomForest.ipynb` in Jupyter Notebook and run all cells. Observe any error produced by any cell:
    - For ModuleNotFoundError, use `pip install` command to install the missing package as explained in Step 2.
    - For any FutureWarning, simply ignore them.
    - If you encounter an error related to `base_estimator`, change it to `estimator`.
- Note that depending on the load of the computer, Random Forest model may take 3 to 5 minutes to complete the training.
- After the execution, you can read the outputs and analyze the model performance. Please write down the outputs, analyze the results and share your understanding below:

Step 4: Repeat Step 3 with all other models

- Repeat Step 3 with the following codes:
    - `AdaBoost.ipynb` for Adaptive Boosting which is a popular ensemble learning algorithm that combines multiple weak learners, typically decision trees with a single split (also known as decision stumps), to form a strong classifier.
    - `CatBoost.ipynb` for Categorical Boosting which is a powerful open-source gradient boosting algorithm developed by Yandex. It is particularly well-suited for handling categorical features.
    - `LightGBM.ipynb` for Light Gradient-Boosting Machine (GBM). It is high-performance, open-source gradient boosting framework developed by Microsoft and is designed to be highly efficient, fast, and scalable.
    - `XGBoost.ipynb` for eXtreme Gradient (XG) Boosting. XGBoost is one of the popular models used in competitions as it often outperforms other algorithms in machine learning competitions.

Step 5: Compare the performance of various models in terms of training time, accuracy and other metrics

- With all the captured results, can you identify which model has the shortest training time, which has the highest accuracy, and which is the best performing model after your analysis?

9.4 Deep Learners

Deep learners, or deep learning models, are a subset of machine learning algorithms that use artificial neural networks with multiple layers to automatically learn representations of data. It is also a popular model for classification. In this experiment, your task is to develop a neural network model using Pytorch that can achieve at least 0.99 accuracy with the given dataset.

The following are some guidance:

- Copy the sample code `ffnn.py` given in the following GitHub repo:
  - https://github.com/cfoh/FFNN-Examples/tree/main/classification-pytorch
- Make corresponding changes to the code, such as:
  - Load the appropriate dataset
  - Perform any preprocessing of data if necessary
  - Set the `run_mode` in the code to 2
  - Remove codes for `further testing` as they are given to test IRIS dataset and not appropriate for our IoT dataset
  - Tune the model to improve performance
  - Tune the training loop to improve the training efficiency