2025 Autmun
Intro-to-Machine Learning
Homework 3

Release Date: 2025/11/04 15:30

Homework 3

- Deadline: 23:59, Nov. 18th (Tue), 2025
- **Coding** (60%): Implement ensemble methods with Numpy and **PyTorch.**
  - Submit your code in executable python files (.py).
  - Report the outcome and parameters by screenshots to the questions.
- **Handwritten Questions** (40%): Answer questions about ensemble methods.
  - Answer the questions in the report.
  - You **must use the template** and in **digital-typed** (**no handwritten scan**)
  - In English

Links

- Questions and Report template
- Sample code / Dataset

Coding Environment

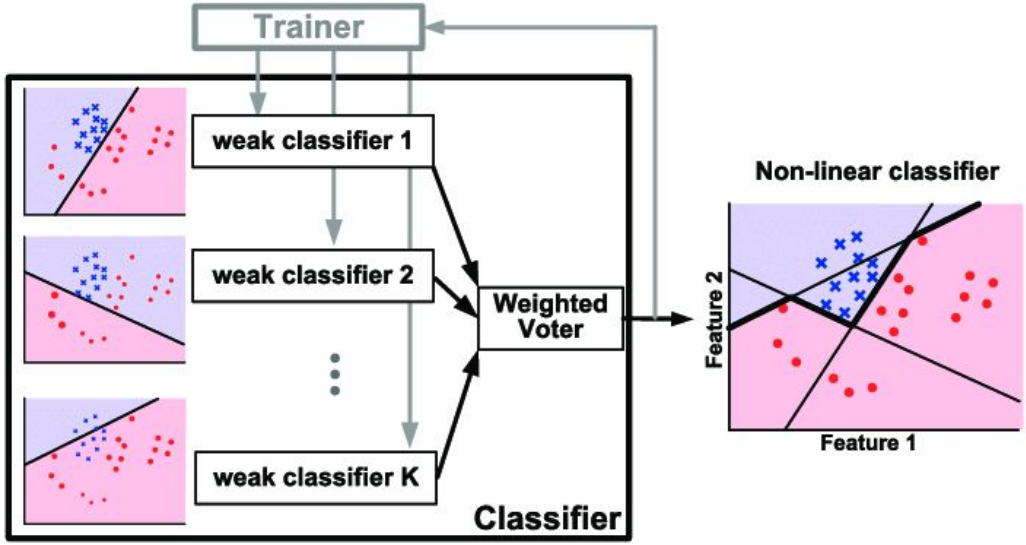
- Recommnedation: Python 3.9 or higher
- Tips
  - We recommend you to use **virtual environments** when implementing your homework assignments.
  - Here are some popular virtual environment management tools
    - uv
    - Poetry
    - Conda
    - Virtualenv

Numpy & PyTorch

- Numpy Tutorial: Link
- **PyTorch Tutorial**: Link
  - Allowed to use any optimizer
  - Not allowed to used built-in loss function (Please implement it by your self!)

Adaboost

- AdaBoost is a boosting technique in machine learning. It is called Adaptive Boosting as the weights are re-assigned to each instance, with higher weights assigned to incorrectly classified instances.



https://www.researchgate.net/figure/Illustration-of-AdaBoost-algorithm-for-creating-a-strong-classifier-based-on-multiple_fig9_288699540

Bagging

- Train multiple classifiers, each with a proportion of data.
- Data is sampled from the whole training set with a sampling with replacement strategy.
- Majority votes for the final prediction

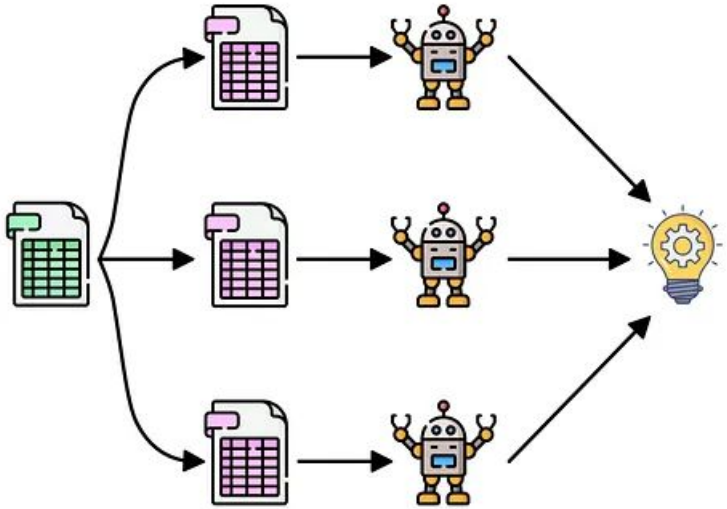


image credit: Fernando López

Decision Tree

- Decision tree is a non-parametric supervised learning algorithm which has a hierarchical, tree structure, which consists of a root node, branches, internal nodes and leaf nodes.

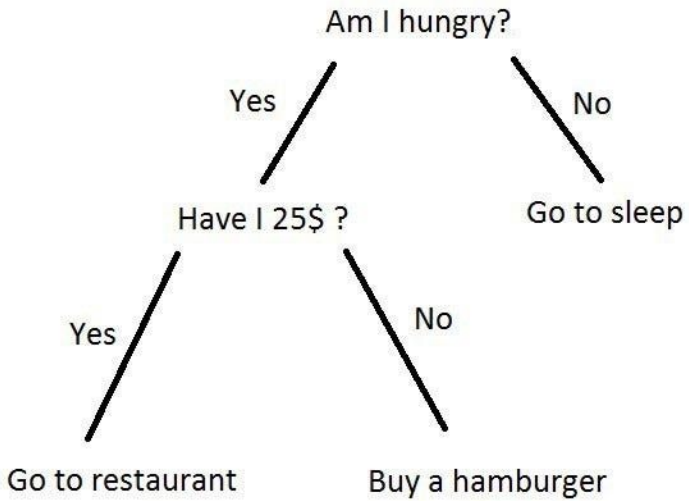


image credit: Rishabh Mall

Dataset and Environment

- Loan Dataset (Binary classification)
  - Already split into training and testing set
- Features
  - You need to preprocess the dataset by yourself.
    - Normalization, encoding, etc.
- Target
  - 0 = Rejected
  - 1 = Approved

AdaBoost (20%)

Accuracy	Score (pt)
>= 0.82	5
>= 0.78	3
>= 0.75	2
< 0.7	0

- Requirements
  - Implement the AdaBoost method with **weak linear classifiers**
    - No non-linear activation function in intermediate layers (Only allowed for output layer)
  - Plot the AUC curve for each weak classifier
  - Plot the feature importance
- Grading Criteria
  - (5%) **Show your accuracy** of the testing data with 10 estimators. (n_estimators=10)
  - (5%) **Plot the AUC curves** of **each weak classifier**.
  - (10%) **Plot the feature importance** of the AdaBoost method.
    - Also, **paste the snapshot of your implementation** of the feature importance estimation.
    - **Explain how you compute** the feature importance of the AdaBoost method shortly (< 100 words)
      - e.g, how to select, what is the key difference with Bagging

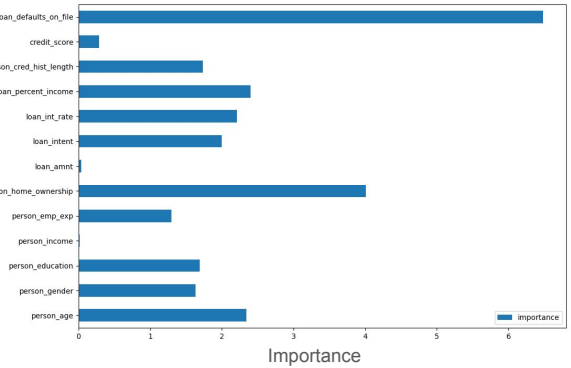
Bagging (20%)

Accuracy	Score (pt)
>= 0.82	5
>= 0.78	3
>= 0.75	2
< 0.7	0

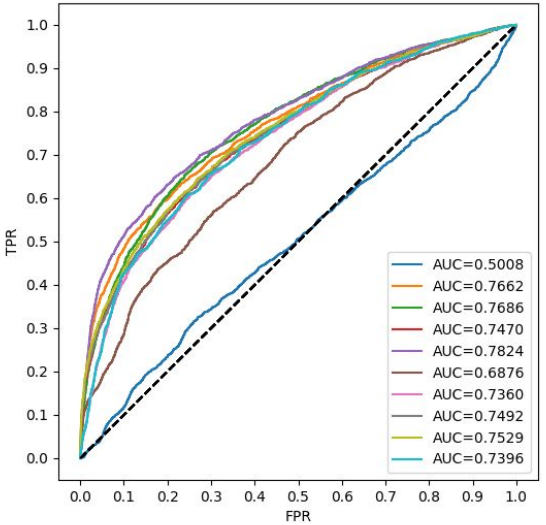
- Requirements
  - Implement the Bagging method with **weak linear classifiers.**
    - No non-linear activation function in intermediate layers (Only allowed for output layer)
  - Plot the AUC curve for each weak classifier
  - Plot the feature importance
- Grading Criteria
  - (5%) **Show your accuracy** of the testing data with 10 estimators. (n_estimators=10)
  - (5%) **Plot the AUC curves** of **each weak classifier**.
  - (10%) **Plot the feature importance** of the Bagging method.
    - Also, **paste the snapshot of your implementation** of the feature importance estimation.
    - **Explain how you compute** the feature importance of the Bagging method shortly (< 100 words)
      - e.g, how to select, what is the key difference with AdaBoost

Plot examples

Examples of figures and code snippet. (Only for reference, not the answer)




```
def compute_feature_importance(self) -> t.Sequence[float]:
    feature_importance = []
```



Note: You can call sklearn to compute AUC

Decision Tree (15%)

Accuracy	Score (pt)
>= 0.90	5
>= 0.80	2
< 0.80	0

- Requirements
  - Implement entropy for measuring the best split of the data.
  - Implement the decision tree classifier with the argument `max_depth`
- Tips
  - Your model should produce the same results when rebuilt with the same arguments.
- Grading Criteria
  - (5%) Compute the gini index and the entropy of the array **[0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1]**.
  - (5%) Show your accuracy of the testing data **with a max-depth = 7**
  - (5%) Plot the feature importance of the decision tree. (No code snapshot for this question)

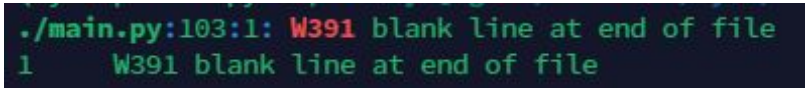
Python Coding Style Guide Reference

1. PEP8
2. Google Python Style

Additional Requirements (5%)

Code Check and Verification: **Lint** the code (5%)

- Code linting: `$ flake8 main.py`
  - -5pt per warning / error (any issue will make you get no point)
  - **Paste the screenshot even there is no output for fully passed linting. (Prove you execute the code linting)**



Example that shows warnings.

Handwritten Questions (40%)

2-1 (15%)

In the AdaBoost algorithm, each selected weak classifier h_t is assigned a weight

$$\alpha_t = \frac{1}{2} ln \frac{1-\epsilon_t}{\epsilon_t}$$

(a) What is the definition of ϵ_t in this formula?

(b) If a weak classifier h_t performs only as well as "random guessing" (i.e., ϵ_t = 0.5), what will α_t be?

(c) Based on your answer in (b), briefly explain how this α_t weight formula ensures the robustness of AdaBoost.

Handwritten Questions (40%)

2-2 (15%)

Random Forest is often considered an improvement over Bagging (Bootstrap Aggregating) when used with Decision Trees as the base learners.

(a) Briefly explain the potential problem that can exist among the T classifiers (C_1, ..., C_T) produced by Bagging when using decision trees.

(b) To mitigate the problem from (a), Random Forest introduces a **second source of randomness** in addition to Bagging. Specifically describe what this second source of randomness is and how it works.

Handwritten Questions (40%)

2-3 (10%)

Is it always possible to find a smallest decision tree that codes a dataset with no error in polynomial time? If not, what algorithm do we usually use to search a decision tree in reasonable time? Also, list the criterion that we stop splitting the decision tree and leaf nodes are created (List two).

Report

- Please follow the report template format. (-5pts if not use the template)
- Link

Submission

- Compress your **code** and **report** into a **.zip file** and submit it to E3.
- Report should be written in English. (-5 pts if not English)
- <STUDENT ID>_HW3.zip
  - main.py
  - src/
  - setup.cfg
  - <STUDENT ID>_HW3.**pdf** (NO .doc, .docx or others format)
- Don’t put the data (e.g. train.csv / test.csv) into submission file

- Any format / submission issue: -5 pts (cumulative)

Other rules

- **Late Policy**: A penalty of **20 points** per additional late day. (-20pt / delayed.day)
  - For example, If you get 90 points but delay for two days, your will get only 50 points!

- **No Plagiarism**: You should complete the assignment by yourself. Students engaged in plagiarism will be penalized heavily. Super serious penalty.
  - e.g. -100pt for the assignment or failed this course, etc
  - Report to academic integrity office

AI-Assistant

- Not recommended but no forbidden
- Copy-and-Paste answers from the AI-Assiant will be seen as Plagiarism
  - However, you can have your own answer first then rephrase it by AI-Assiant.
- Some questions might be parts of final exam, make sure you understand the concept

FAQs

- If you have other questions, ask on **E3 forum** first! We will reply as soon as possible.
  - Also, feel free to write email to TAs (And remember to cc all TAs).

Have Fun!