

2024 Autumn Intro-to-Machine Learning Homework 3

Release Date: 2024/11/05 15:00

Homework 3

- Deadline: 23:59, Nov. 19th (Tue), 2024
- **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

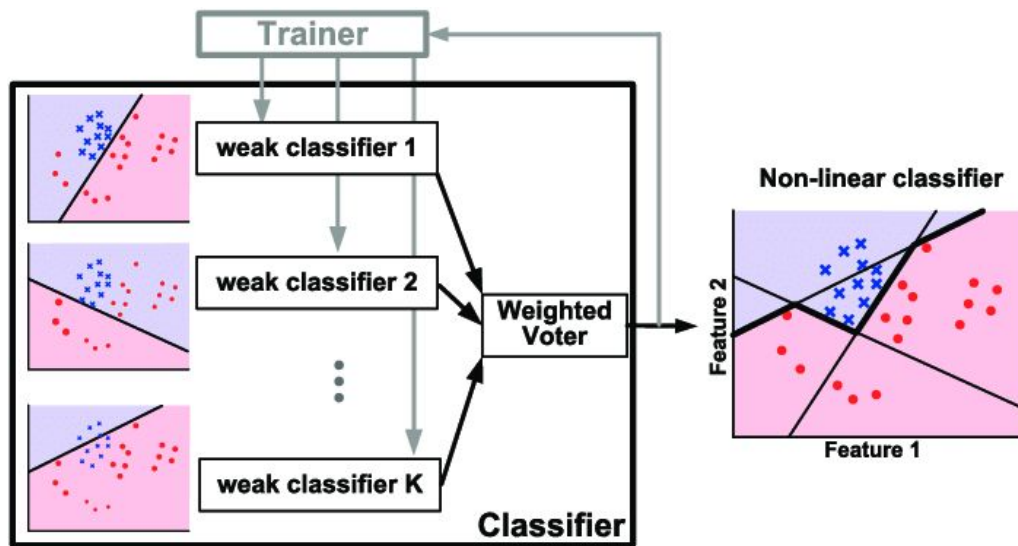
- Recommendation: Python 3.9 or higher
- Tips
 - We recommend you to use **virtual environments** when implementing your homework assignments.
 - Here are some popular virtual environment management tools
 - [Poetry](#)
 - [Conda](#)
 - [Virtualenv](#)

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.



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

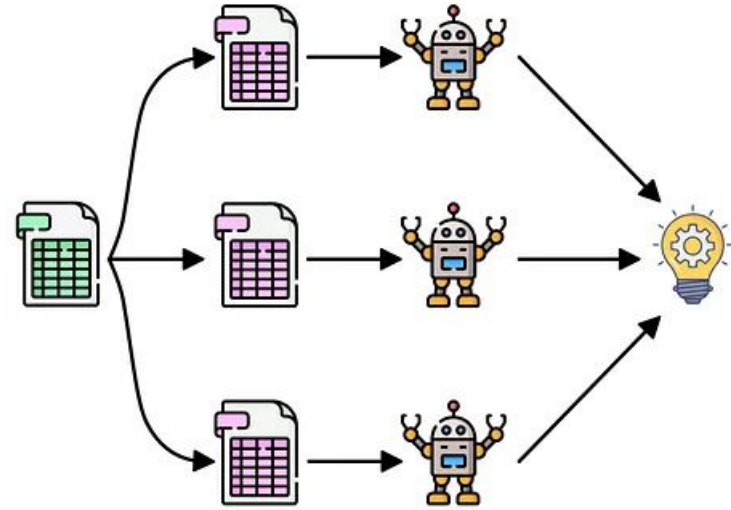
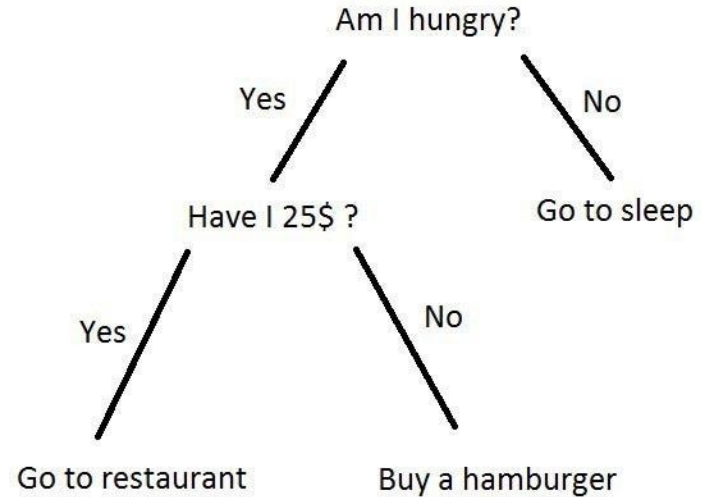


image credit: [Fernando López](#)

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.



Dataset and Environment

- Loan Dataset (Binary classification)
 - Already split into training and testing set
- Features (#features = 13)
 - age, gender, education, income, etc.
 - You may need to preprocess the dataset by yourself. (e.g., map string to integer)
- Target
 - target (0 = loan rejected, 1 = loan approved)
- Required packages: `numpy`, `pandas`, `matplotlib`, `loguru`, `flake8`, `scikit-learn`, PyTorch

AdaBoost (20%)

- Requirements

- Implement the AdaBoost method with weak linear classifiers
- Plot the AUC curve for each weak classifier
- Plot the feature importance

- Grading Criteria

- (10%) Show your accuracy of the testing data with 10 estimators. (n_estimators=10)
- (5%) Plot the AUC curves of each weak classifier.
- (5%) Plot the feature importance of the AdaBoost method. Also, you should snapshot the implementation of the feature importance estimation. (Note: one axis is feature name and the other axis is its corresponding importance value)

Accuracy	Score (pt)
≥ 0.80	10
≥ 0.75	8
≥ 0.7	5
< 0.7	0

Note

For the linear classifier, no non-linear activation for the middle layer, but allowed for outputs

Bagging (20%)

- Requirements

- Implement the Bagging method with weak linear classifiers
- Plot the AUC curve of each weak classifier
- Plot the feature importance

- Grading Criteria

- (10%) Show your accuracy of the testing data with 10 estimators. (n_estimators=10)
- (5%) Plot the AUC curves of each weak classifier.
- (5%) Plot the feature importance of the Bagging method. Also, you should snapshot the implementation of the feature importance calculation. (Note: one axis is feature name and the other axis is its corresponding importance value)

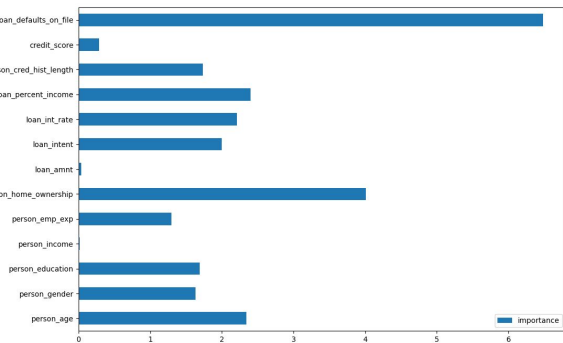
Accuracy	Score (pt)
≥ 0.80	10
≥ 0.75	8
≥ 0.7	5
< 0.7	0

Note

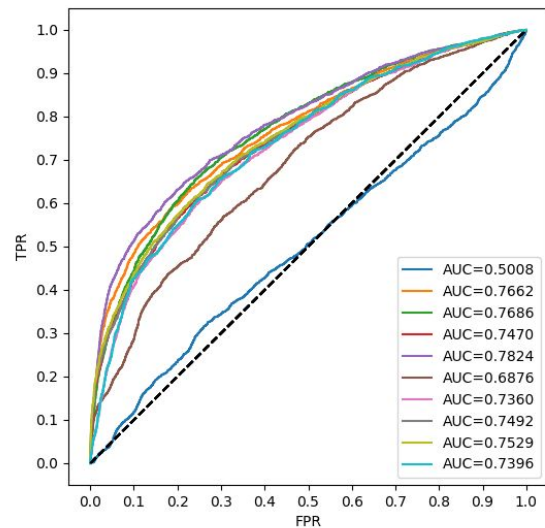
For the linear classifier, no non-linear activation for the middle layer, but allowed for outputs

Plot examples

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



```
def compute_feature_importance(self) -> t.Sequence[float]:  
    feature_importance = []  
    for feature in self.features:  
        importance = self.compute_importance(feature)  
        feature_importance.append(importance)  
    return feature_importance
```



Decision Tree (15%)

Accuracy	Score (pt)
≥ 0.90	10
≥ 0.80	5
< 0.70	0

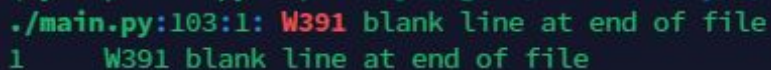
- Requirements
 - Implement entropy for measuring the best split of the data.
 - Implement the decision tree classifier with the argument ``max_depth``
- Tips
 - Your model should produce the same results when rebuilt with the same arguments.
- Grading Criteria
 - (5%) Compute the gini index and the entropy of the array [0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1].
 - (5%) Show your accuracy of the testing data with a max-depth = 7
 - (5%) Plot the feature importance of the decision tree.

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)**



```
./main.py:103:1: W391 blank line at end of file
1      W391 blank line at end of file
```

Example that shows warnings.

Handwritten Questions (40%)

2-1 (10%) What are Bagging and Boosting, and how are they different? Please list their difference and compare the two methods in detail.

2-2 (15%) What criterion do we use to decide when we stop splitting while performing the decision tree algorithm? Please list at least three criteria.

Handwritten Questions (40%)

2-3 (15%) A student claims to have a brilliant idea to make random forests more powerful: since random forests prefer trees which are diverse, i.e., not strongly correlated, the student proposes setting $m = 1$, where m is the number of random features used in each node of each decision tree. The student claims that this will improve accuracy while reducing variance. Do you agree with the student's claims? Clearly explain your answer.

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

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!

