

NYCU Pattern Recognition, Homework 2

109550090 李以恩

Part. 1, Coding (80%):

For this coding assignment, you are required to implement the Decision Tree and Random Forest algorithms using only NumPy. Afterward, you will need to train your model on the provided dataset and evaluate its performance on the validation data.

(30%) Decision Tree

Requirements:

- Implement the **Gini** and **Entropy** for measuring the "best" splitting of the data.
- Implement the Decision Tree algorithm ([CART, Classification and Regression Trees](#)) with the following 3 arguments:
- **Criterion**: The function to measure the quality of a split of the data. Your model should support "gini" for the Gini impurity and "entropy" for the information gain.
- **Max_depth**: The maximum depth of the tree. If Max_depth=None, then nodes are expanded until all leaves are pure. Max_depth=1 equals splitting data once.
- **Max_features**: The number of features to consider when looking for the best split. If None, then max_features=n_features.
- For more detailed descriptions of the arguments, please refer to [Scikit-learn](#).
- Your model should produce the same results when rebuilt with the same arguments, and there is no need to prune the trees.
- You can use the recursive method to build the nodes.

Criteria:

1. (5%) Compute the Entropy and Gini index of the array provided in the sample code, using the formulas on page 6 of the HW3 slide.

```
['+' '+' '+' '+' '+' '-']: entropy = 0.6500224216483541
['+' '+' '+' '-' '-' '-']: entropy = 1.0
['+' '-' '-' '-' '-' '-']: entropy = 0.6500224216483541

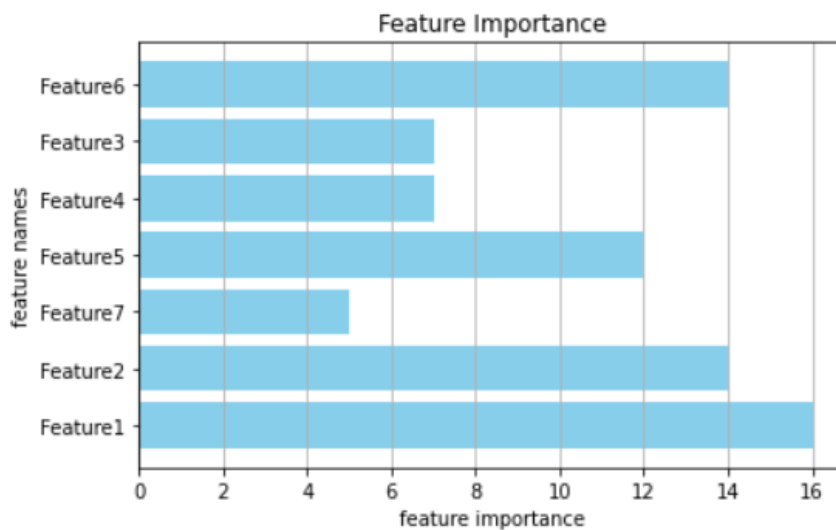
['+' '+' '+' '+' '+' '-']: gini index = 0.2777777777777777
['+' '+' '+' '-' '-' '-']: gini index = 0.5
['+' '-' '-' '-' '-' '-']: gini index = 0.2777777777777777
```

2. (10%) Show the accuracy score of the validation data using criterion='gini' and max_features=None for max_depth=3 and max_depth=10, respectively.

Q2-1 max_depth=3: 0.73125

Q2-2 max_depth=10: 0.865

3. (10%) Show the accuracy score of the validation data using `max_depth=3` and `max_features=None`, for `criterion='gini'` and `criterion='entropy'`, respectively.
- Q3-1 `criterion='gini': 0.73125`
- Q3-2 `criterion='entropy': 0.77`
4. (5%) Train your model using `criterion='gini'`, `max_depth=10` and `max_features=None`. Plot the [feature importance](#) of your decision tree model by simply counting the number of times each feature is used to split the data.



(20%) Random Forest

Requirements:

- Fix the random seed.
- Implement the Random Forest algorithm by using the CART you just implemented.
- The Random Forest model should include the following three arguments:
- **N_estimators**: The number of trees in the forest.
- **Max_features**: The number of features to consider when looking for the best split using the decision tree.
- **Bootstrap**: Whether to use bootstrap samples when building trees.
- For more detailed descriptions of the arguments, please refer to [Scikit-learn](#).
- Use majority voting to obtain the final prediction.

Criteria:

5. (10%) Show the accuracy score of the validation data using `criterion='gini'`, `max_depth=None`, `max_features=sqrt(n_features)`, and `bootstrap=True`, for `n_estimators=10` and `n_estimators=50`, respectively.

Q6-1 `n_estimators=10`: 0.88875

Q6-1 `n_estimators=50`: 0.895

6. (10%) Show the accuracy score of the validation data using `criterion='gini'`, `max_depth=None`, `n_estimators=10`, and `bootstrap=True`, for `max_features=sqrt(n_features)` and `max_features=n_features`, respectively.

Q7-1 `max_features='sqrt'`: 0.88875

Q7-1 `max_features='All'`: 0.88125

(20%) Train your own model**Requirements:**

- Train your model (either Decision Tree or Random Forest).
- Try different parameters and feature engineering to beat the baseline.
- Save your test predictions in a CSV file.

Criteria:

7. (20%) Explain how you chose/design your model and what feature processing you have done in detail. Otherwise, no points will be given.

Points	Testing Accuracy
20 points	<code>acc >= 0.90625</code>
15 points	<code>acc > 0.89</code>
10 points	<code>acc > 0.85</code>
5 points	<code>acc > 0.8</code>
0 points	<code>acc <= 0.8</code>

1) Feature Engineering

From the data info, we can notice that the value range of Feature1 and Feature6 differs significantly from the other features. However, doing the normalization didn't achieve a better performance. I've also tried some feature engineering methods like in hw1 and hw2(e.g. polynomial feature engineering), but the performance was even worse under the same conditions. Therefore, I chose to do nothing in feature engineering.

	Feature1	Feature2	Feature3	Feature4	Feature5 \
count	800.000000	800.000000	800.000000	800.000000	800.000000
mean	843.891250	0.752642	0.747779	0.987290	0.874084
std	198.267589	0.085675	0.047478	0.004409	0.057975
min	531.000000	0.353000	0.583000	0.964000	0.578000
25%	706.000000	0.718000	0.713000	0.986000	0.837000
50%	796.000000	0.763000	0.756000	0.988000	0.882500
75%	953.250000	0.806000	0.784250	0.990000	0.916000
max	1720.000000	0.896000	0.841000	0.993000	0.980000

	Feature6	Feature7	Target
count	800.000000	800.000000	800.000000
mean	0.001722	0.995340	3.578750
std	0.000562	0.003971	1.858929
min	0.000650	0.972000	0.000000
25%	0.001210	0.994000	3.000000
50%	0.001710	0.997000	3.000000
75%	0.002122	0.998000	5.000000
max	0.003320	1.000000	6.000000

2) Model Selection

```
1 # Build and train your model
2 np.random.seed(6)
3 your_model = RandomForest(n_estimators=20, max_features=np.sqrt(X_train.shape[1]),
4                           bootstrap=True, criterion='entropy', max_depth=15)
5 your_model.fit(X_train, y_train)
6 print("your_model validation accuracy: ", accuracy_score(y_val, your_model.predict(X_val)))

-----Processing forest 0-----
-----Processing forest 1-----
-----Processing forest 2-----
-----Processing forest 3-----
-----Processing forest 4-----
-----Processing forest 5-----
-----Processing forest 6-----
-----Processing forest 7-----
-----Processing forest 8-----
-----Processing forest 9-----
-----Processing forest 10-----
-----Processing forest 11-----
-----Processing forest 12-----
-----Processing forest 13-----
-----Processing forest 14-----
-----Processing forest 15-----
-----Processing forest 16-----
-----Processing forest 17-----
-----Processing forest 18-----
-----Processing forest 19-----
your_model validation accuracy: 0.905
```

- **seed=6**: By trial and error, the best result I determined.
- **n_estimators=20**: By trial and error, the best result I determined.
Having more trees in a random forest does not necessarily result in better performance.
- **max_features=np.sqrt(X_train.shape[1])**: Taking part of the features while training is better than taking all the features.
- **bootstrap=True**: Using bootstrap can achieve better performance.
- **criterion='entropy'**: Using entropy criterion is better than gini.
- **max_depth=15**: Put a limitation on max_depth to avoid trees growing too complex in order to avoid overfitting problems.
-

Part. 2, Questions (30%):

Answer the following questions in detail:

a. Why does a decision tree tend to overfit the training set?

Overfit condition arises when the model memorizes the noise of the training data and fails to capture important patterns, meaning that the decision tree may fit the training data too closely, resulting in poor generalization performance on unseen data. Also, decision trees may become overly complex, especially when the tree is deep and has many decision nodes, which can lead to overfitting as the tree becomes too specialized to the training data.

b. Is it possible for a decision tree to achieve 100% accuracy on the training set?

Yes. This can happen when the decision tree is able to perfectly partition the feature space based on the training data, resulting in pure leaf nodes that correctly predict the labels of all training samples.

c. List and describe at least three strategies we can use to reduce the risk of overfitting in a decision tree.

- Pre-Pruning

Early stop of the growth of the decision tree, e.g. judging max_depth, min_samples_leaf, min_samples_split

- Post-Pruning

Allow the decision tree model to grow to its full depth, then remove the tree branches to prevent the model from overfitting, e.g. Cost complexity pruning (ccp).

- Random Forest

Random Forest is an ensemble technique for classification and regression by bootstrapping multiple decision trees. Random Forest follows bootstrap sampling and aggregation techniques to prevent overfitting.

For each statement, answer True or False and provide a detailed explanation:

d. In AdaBoost, weights of the misclassified examples go up by the same multiplicative factor.

True. The adaboost algorithm was formulated for weak classifiers that predict ± 1 labels. The weights of all misclassified points will be multiplied by $\exp(-\alpha * y * h(x_i)) = \exp(A)$ before normalization, where A is the same for all misclassified examples.

- Update distribution:

$$D_{t+1}(i) = \frac{D_t(i) \exp[-\alpha_t y_i h_t(x_i)]}{Z_t}, \text{ } Z_t \text{ is for normalization}$$

e. In AdaBoost, weighted training error ϵ_t of the t_{th} weak classifier on training data with weights D_t tends to increase as a function of t.

True. In the course of boosting iterations, the weak classifiers are forced to try to classify more difficult examples. The weights will increase for examples which have wrong predictions from weak classifiers. The weighted training error ϵ_t of the t_{th} weak classifier therefore tends to increase.

f. AdaBoost will eventually give zero training error regardless of the type of weak classifier it uses, provided enough iterations are performed.

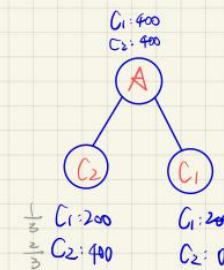
False. If the data is not separable by a linear combination of the weak classifiers, AdaBoost can't achieve zero training error.

Consider a data set comprising 400 data points from class C_1 and 400 data points from class C_2 . Suppose that a tree model A splits these into (200, 400) at the first leaf node and (200, 0) at the second leaf node, where (n, m) denotes that n points are assigned to C_1 and m points are assigned to C_2 . Similarly, suppose that a second tree model B splits them into (300, 100) and (100, 300). **Evaluate the misclassification rates for the two trees and hence show that they are equal.** Similarly, evaluate the

cross-entropy $Entropy = - \sum_{k=1}^K p_k \log_2 p_k$ and **Gini index**

$Gini = 1 - \sum_{k=1}^K p_k^2$ for the two trees. Define p_k to be the proportion of data

points in region R assigned to class k, where $k = 1, \dots, K$.

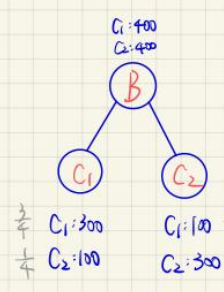


Tree A:
Root A splits into C2 and C1.
C2: $C_1: 200, C_2: 400$
C1: $C_1: 200, C_2: 0$

(a) Misclassification rate: $\frac{200}{800} + \frac{0}{800} = 0.25 \#$

(b) Entropy = $-\sum_{k=1}^K p_k \log_2 p_k$
 $= -\left[\frac{600}{800} \left(\frac{1}{3} \log_2 \frac{1}{3} + \frac{2}{3} \log_2 \frac{2}{3} \right) + \frac{200}{800} (1 \log_2 1 + 0 \log_2 0) \right]$
 $= -\left[\frac{3}{4} (0.333 \times -1.586 + 0.667 \times -0.584) + \frac{1}{4} \cdot 0 \right]$
 $= -\frac{3}{4} (-0.528 - 0.390) = 0.689 \#$

(c) Gini = $1 - \sum_{k=1}^K p_k^2$
 $= \frac{600}{800} \left[1 - \left(\left(\frac{1}{3} \right)^2 + \left(\frac{2}{3} \right)^2 \right) \right] + \frac{200}{800} [1 - (1^2 + 0^2)]$
 $= \frac{3}{4} \cdot \left[1 - \frac{5}{9} \right] = \frac{1}{3} = 0.333 \#$



Tree B:
Root B splits into C1 and C2.
C1: $C_1: 300, C_2: 100$
C2: $C_1: 100, C_2: 300$

(a) Misclassification rate: $\frac{100}{800} + \frac{100}{800} = 0.25 \#$

(b) Entropy = $-\sum_{k=1}^K p_k \log_2 p_k$
 $= -\left[\frac{400}{800} \left(\frac{3}{4} \log_2 \frac{3}{4} + \frac{1}{4} \log_2 \frac{1}{4} \right) + \frac{400}{800} \left(\frac{1}{4} \log_2 \frac{1}{4} + \frac{3}{4} \log_2 \frac{3}{4} \right) \right]$
 $= -\frac{1}{2} \cdot 2 \cdot (0.75 \times -0.415 + 0.25 \times -2)$
 $= 0.311 + 0.5 = 0.811 \#$

(c) Gini = $1 - \sum_{k=1}^K p_k^2$
 $= \frac{400}{800} \left[1 - \left(\left(\frac{3}{4} \right)^2 + \left(\frac{1}{4} \right)^2 \right) \right] + \frac{400}{800} \left[1 - \left(\left(\frac{1}{4} \right)^2 + \left(\frac{3}{4} \right)^2 \right) \right]$
 $= \frac{1}{2} \cdot 2 \cdot \left[1 - \frac{10}{16} \right] = \frac{3}{8} = 0.375 \#$