NYCU Introduction to Machine Learning, Homework 3

Deadline: Nov. 15, 23:59

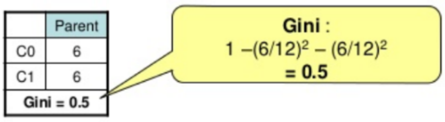
Part. 1, Coding (80%):

In this coding assignment, you need to implement the Decision Tree, AdaBoost and Random Forest algorithm by using only NumPy, then train your implemented model by the provided dataset and test the performance with testing data. Find the sample code and data on the GitHub page https://github.com/NCTU-VRDL/CS_CS20024/tree/main/HW3

Please note that only NumPy can be used to implement your model, you will get no points by simply calling sklearn.tree.DecsionTreeClassifier.

1. (5%) Gini Index or Entropy is often used for measuring the "best" splitting of the data. Please compute the Entropy and Gini Index of this array np.array([1,2,1,1,1,1,2,2,1,1,2]) by the formula below. (More details on page 5 of the hw3 slides, 1 and 2 represent class1 and class 2, respectively)

$$Gini = 1 - \sum_j p_j^2$$



$$Entropy = -\sum_j p_j \log_2 p_j$$

- If all classes are the same in one node

$$entropy = -1\log_2 1 = 0$$

- If the classes are half-and-half

$$entropy = -0.5\log_2 0.5 - 0.5\log_2 0.5 = 1$$

```
print("Gini of data is ", gini(data))
```

```
Gini of data is  0.4628099173553719
```

```
print("Entropy of data is ", entropy(data))
```

```
Entropy of data is  0.9456603046006401
```

2. (10%) Implement the Decision Tree algorithm (CART, Classification and Regression Trees) and train the model by the given arguments, and print the accuracy score on the test data. You should implement **two arguments** for the Decision Tree algorithm, 1) **Criterion**: The function to measure the quality of a split. Your model should support "gini" for the Gini impurity and "entropy" for the information gain.
2) **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 split data once
   2.1. Using Criterion= 'gini', showing the accuracy score of test data by Max_depth=3 and Max_depth=10, respectively.

```
0.9166666666666666
0.9366666666666666
```

above: Max_depth = 3

below: Max_depth = 10

2.2. Using Max_depth=3, showing the accuracy score of test data by Criterion= ‘gini’ and Criterion=’ entropy’ , respectively.

```
0.9166666666666666
0.93
```

Above: gini

Below: entropy

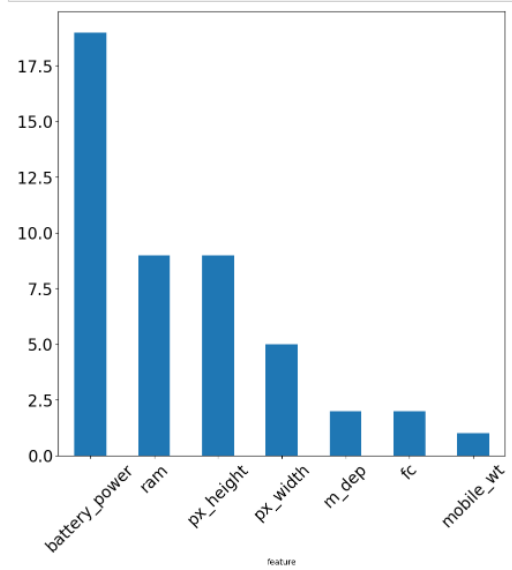
Note: Your decision tree scores should be over 0.9. It may suffer from overfitting, if so, you can tune the hyperparameter such as `max_depth`

*Note: You should get the same results when re-building the model with the same arguments, **no need to prune the trees***

Note: You can find the best split threshold by both methods. First one: 1) Try N-1 threshold values, where the i-th threshold is the average of the i-th and (i+1)-th sorted values. Second one: Use the unique sorted value of the feature as the threshold to split

Hint: You can use the recursive method to build the nodes

3. (5%) Plot the feature importance of your Decision Tree model. You can use the model from Question 2.1, max_depth=10. (You can use simply counting to get the feature importance instead of the formula in the reference, more details on the sample code. **Matplotlib** is allowed to be used)



4. (15%) Implement the AdaBoost algorithm by using the CART you just implemented from question 2. You should implement one argument for the AdaBoost.

   1) **N_estimators**: The number of trees in the forest.

   4.1. Showing the accuracy score of test data by n_estimators=10 and n_estimators=100, respectively.

   ```
   0.9466666666666667
   0.9733333333333334
   ```

   Above: n_estimators = 10

   Below: n_estimators = 100

5. (15%) Implement the Random Forest algorithm by using the CART you just implemented from question 2. You should implement three arguments for the Random Forest.

   1) **N_estimators**: The number of trees in the forest.

   2) **Max_features**: The number of features to consider when looking for the best split

   3) **Bootstrap**: Whether bootstrap samples are used when building trees

   5.1. Using Criterion= 'gini' , Max_depth=None, Max_features=sqrt(n_features), Boo tstrap=True, showing the accuracy score of test data by n_estimators=10 and n_est imators=100, respectively.

   ```
   0.9266666666666666
   0.9433333333333334
   ```

   Above: n_estimators=10

   Below: n_estimators=100

   5.2. Using Criterion= 'gini' , Max_depth=None, N_estimators=10, Bootstrap=True, showing the accuracy score of test data by Max_features=sqrt(n_features) and Max _features=n_features, respectively.

   ```
   0.9133333333333333
   0.9566666666666667
   ```

   Above: Max_features=sqrt(n_features)

   Below: Max_features=n_features

   Note: Use majority votes to get the final prediction, you may get different results when re-building the random forest model

6. (20%) Tune the hyperparameter, perform feature engineering or implement more po werful ensemble methods to get a higher accuracy score. Please note that only the e nsemble method can be used. The neural network method is not allowed.

Accuracy	Your scores
acc > 0.975	20 points

0.95 < acc <= 0.975	15 points
0.9 < acc <= 0.95	10 points
acc < 0.9	0 points

Score: 0.97

Part. 2, Questions (30%):

1. Why does a decision tree have a tendency to overfit to the training set? Is it possible for a decision tree to reach a 100% accuracy in the training set? please explain. List and describe at least 3 strategies we can use to reduce the risk of overfitting of a decision tree.

   A: For each split in decision tree, it always finds the smallest entrophy weighted mean for the children, and the amount of segmentation is depending on the data amount and class amount, thus segmentation amount is limited, so we could always find a best way to split the data by calculating entropy, so it can get the most accurate splitting result on training data, thus it tends to overfitting.

   Yes, it is possible for a decision tree to reach a 100% accuracy in the training set, if you can make sure that no data from different class will have the same value. Since if so, no classifier could split those data because they all look alike.

   Ways to reduse the risk for overfitting:

   Fixed depth ,so it won’t split too many times to get the most accurate answer for training data.

   Tree pruning, like fixed depth, we discard the nodes that we think not important, like a desition node only split 2 datas.

   Bagging: We randomly sample n repeatable datas from the original data, so we could get multiple data sets from the original one, then for each data, we build their own classifier, finally, we get the predict answer by voting from multiple datasets classifier, since each sampled datasets are randomized, some data of the original one won’t be selected, thus it won’t overfitting.

   This part consists of three True/False questions. Answer True/False for each question and briefly explain your answer.

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

True, since the weights is decided by below sentence, for all wrong samples it gets -1 on yh(x).

$$\exp[-\alpha_t y_i h_t(x_i)]$$

b. In AdaBoost, weighted training error $\boldsymbol{\varepsilon}_t$ of the t_{th} weak classifier on training data with weights D_t tends to increase as a function of t.

False, logically, you would expect to have the error decrease while training. But it is possible for some training error to increase since we use different t for each round by applying the d trained by the t before. But the overall trend for error is to decrease.

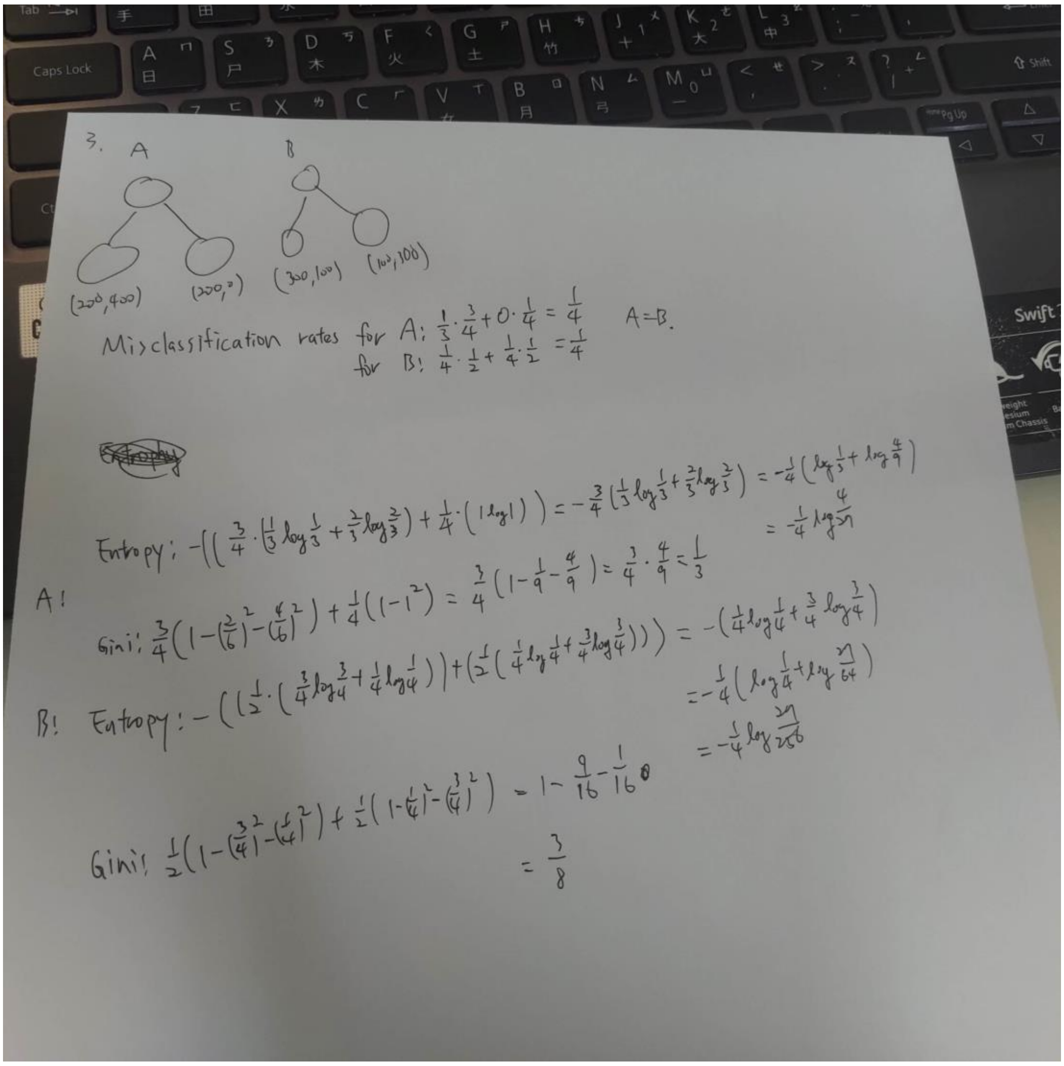
c. AdaBoost will eventually give zero training error regardless of the type of weak classifier it uses, provided enough iterations are performed.

True, if a classifier's accuracy is lower than 50%, the classifier would be useless, since random guessing could be better than that, so a classifier's accuracy should be over 50%, by applying multiple classifier, accuracy will eventually increase to 100% if enough training times given.

2. 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 <u>misclassification rates</u> for the two trees and hence show that they are equal.** Similarly, evaluate the cross-entropy $\boldsymbol{Entropy} = -\sum_{k=1}^{K} \boldsymbol{p_k \log_2 p_k}$ and Gini index $Gini = 1 - \sum_{k=1}^{K} p_k^2$ for the two tre

es. Define p_k to be the proportion of data points in region R assigned to class k, where

k = 1, . . . , K.



3. A B

(200,400) (200,0) (300,100) (100,300)

Misclassification rates for A: $\frac{1}{3}\cdot\frac{3}{4}+0\cdot\frac{1}{4}=\frac{1}{4}$ A=B.

for B: $\frac{1}{4}\cdot\frac{1}{2}+\frac{1}{4}\cdot\frac{1}{2}=\frac{1}{4}$

~~Entropy~~

A:

Entropy: $-\left(\left(\frac{3}{4}\cdot\left(\frac{1}{3}\log\frac{1}{3}+\frac{2}{3}\log\frac{2}{3}\right)+\frac{1}{4}\cdot(1\log 1)\right)=-\frac{3}{4}\left(\frac{1}{3}\log\frac{1}{3}+\frac{2}{3}\log\frac{2}{3}\right)=-\frac{1}{4}\left(\log\frac{1}{3}+\log\frac{4}{9}\right)$

$=-\frac{1}{4}\log\frac{4}{27}$

Gini: $\frac{3}{4}\left(1-\left(\frac{2}{6}\right)^2-\left(\frac{4}{6}\right)^2\right)+\frac{1}{4}(1-1^2)=\frac{3}{4}\left(1-\frac{1}{9}-\frac{4}{9}\right)=\frac{3}{4}\cdot\frac{4}{9}=\frac{1}{3}$

B: Entropy: $-\left(\left(\frac{1}{2}\cdot\left(\frac{3}{4}\log\frac{3}{4}+\frac{1}{4}\log\frac{1}{4}\right)\right)+\left(\frac{1}{2}\left(\frac{1}{4}\log\frac{1}{4}+\frac{3}{4}\log\frac{3}{4}\right)\right)\right)=-\left(\frac{1}{4}\log\frac{1}{4}+\frac{3}{4}\log\frac{3}{4}\right)$

$=-\frac{1}{4}\left(\log\frac{1}{4}+\log\frac{27}{64}\right)$

$=-\frac{1}{4}\log\frac{27}{256}$

Gini: $\frac{1}{2}\left(1-\left(\frac{3}{4}\right)^2-\left(\frac{1}{4}\right)^2\right)+\frac{1}{2}\left(1-\left(\frac{1}{4}\right)^2-\left(\frac{3}{4}\right)^2\right)=1-\frac{9}{16}-\frac{1}{16}$

$=\frac{3}{8}$