NYCU Introduction to Machine Learning, Homework 3

Deadline: Nov. 15, 23:59

109550064 陳姵帆

Part. 1, Coding (80%):

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])`

   Gini Index

```
print("Gini of data is ", gini(data))
✓ 0.2s

Gini of data is  0.4628099173553719
```

   Entropy

```
print("Entropy of data is ", entropy(data))
✓ 0.3s

Entropy of data is  0.9456603046006401
```

2. (10%) Implement the Decision Tree algorithm (CART, Classification and Regression Trees)

   2.1. Using Criterion='gini', showing the accuracy score of test data by Max_depth=3 and Max_depth=10, respectively.

```
clf_depth3 = DecisionTree(criterion='gini', max_depth=3)
clf_depth10 = DecisionTree(criterion='gini', max_depth=10)
✓ 0.2s

clf_depth3.fit(x_train_data, y_train_data)
print("clf_depth3 acc: ", get_acc(clf_depth3.predict(x_test_data), y_test_data))

clf_depth10.fit(x_train_data, y_train_data)
print("clf_depth10 acc: ", get_acc(clf_depth10.predict(x_test_data), y_test_data))
✓ 44.1s

clf_depth3 acc:  0.92
clf_depth10 acc:  0.93
```

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

```
clf_gini = DecisionTree(criterion='gini', max_depth=3)
clf_entropy = DecisionTree(criterion='entropy', max_depth=3)
```

✓ 0.2s

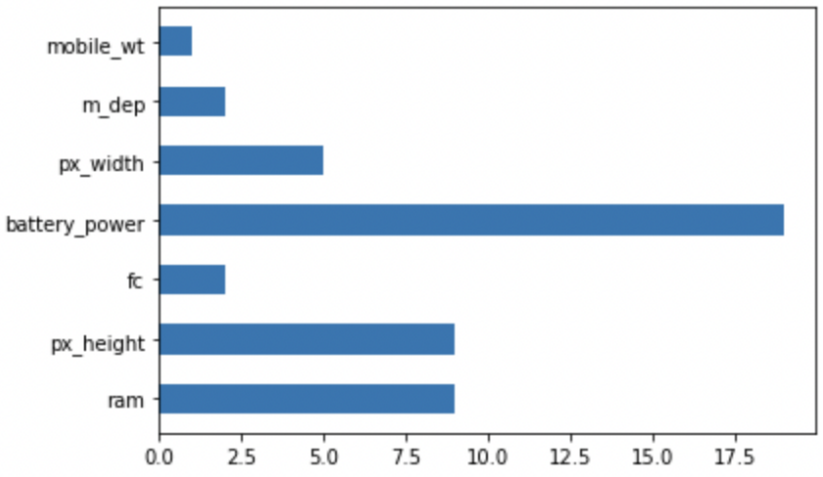
```
clf_gini.fit(x_train_data, y_train_data)
print("clf_gini acc: ", get_acc(clf_gini.predict(x_test_data), y_test_data))
clf_entropy.fit(x_train_data, y_train_data)
print("clf_entropy acc: ", get_acc(clf_entropy.predict(x_test_data), y_test_data))
```

✓ 40.3s

```
clf_gini acc:  0.92
clf_entropy acc:  0.9333333333333333
```

I Use the unique sorted value of the feature as the threshold to split

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)




```
ram :  9
px_height :  9
fc :  2
battery_power :  19
px_width :  5
m_dep :  2
mobile_wt :  1
```

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.

```
adaboost_10.fit(x_train_data, y_train_data)
print("adaboost_10 acc: ", get_acc(adaboost_10.predict(x_test_data), y_test_data))
```

[267] ✓ 1m 23.5s

```
adaboost_10 acc:  0.9433333333333334
```

```
adaboost_100.fit(x_train_data, y_train_data)
y_predict_adaboost_100 = adaboost_100.predict(x_test_data)
print("adaboost_100 acc: ", get_acc(adaboost_100.predict(x_test_data), y_test_data))
```

✓ 13m 55.3s

```
adaboost_100 acc:  0.97
```

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), Bootstrap=True, showing the accuracy score of test data by n_estimators=10 and n_estimators=100, respectively.

```
clf_10tree.fit(x_train_data, y_train_data)
print("clf_10tree acc: ", get_acc(clf_10tree.predict(x_test_data), y_test_data))
```

] ✓ 46.8s

```
clf_10tree acc:  0.9166666666666666
```

```
clf_100tree.fit(x_train_data, y_train_data)
print("clf_100tree acc: ", get_acc(clf_100tree.predict(x_test_data), y_test_data))
```

] ✓ 8m 37.1s

```
clf_100tree acc:  0.9433333333333334
```

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.

```
    clf_random_features.fit(x_train_data, y_train_data)
    print("clf_random_features acc: ", get_acc(clf_random_features.predict(x_test_data), y_test_data))
✓ 51.9s

clf_random_features acc:  0.9266666666666666
```

```
    clf_all_features.fit(x_train_data, y_train_data)
    print("clf_all_features acc: ", get_acc(clf_all_features.predict(x_test_data), y_test_data))
✓ 3m 25.4s

clf_all_features acc:  0.9533333333333334
```

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 powerful ensemble methods to get a higher accuracy score. Screenshot your tests score on the report. Please note that only the ensemble method can be used. The neural network method is not allowed.

```
    my_model = train_your_model(x_train_data, y_train_data)
    my_pred = my_model.predict(x_test_data)
    print("adaboost_170 acc: ", get_acc(my_pred, y_test_data))

adaboost_170 acc:  0.9833333333333333
```

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

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.

   (1) 因為 decision tree 只要你想要的話, 是可以把訓練資料集分到完全是純的(不純就再 split 下去, 且不純應該要有 feature 可以協助 split 下去), 所以會有 overfit to the training set 的可能性。

(decision tree 不像 random forest 會遇到兩筆資料之於 random 挑選出 feature 值都相同但是 label 不同的情況(若挑出 feature 數量小於所有 feature 數量), 因為 decision tree 是拿所有的 feature 下去考慮, 若兩筆資料 "所有 feature" 值都相同, 那他們的 label 應該也要相同)

(2) Yes, it is possible. 如同 (1) 所說, 只要切到所有 leaf node 拿到的 data 都是同一個 label (純的) 就代表我們對 train dataset 有 100% accuracy 了。(不同 label 的資料都有好好分開)

(3) [1] pre-pruning

邊訓練邊限制樹的長相, 可以限制 decision tree 的最大深度(max_depth)、限制一個葉節點必須有多少樣本(min_samples_leaf)、葉節點的最大數量(max_leaf_nodes)、在 split 時評估的特徵的最大數量(max_features)。提高 min* 超參數或降低 max* 超參數都可以限制 decision tree 的自由度＆協助不要讓 decision tree overfit。

[2] post-pruning

一開始不限制樹的生長, 等 train 完確定樹的架構後, 在對樹剪枝。

[3] Ensemble - Random Forest

Creating multiple trees, with each tree trained slightly differently and with the random factor. Thus, it is hard to make the ensemble of these randomly different trees to overfit on the same dataset.

2. 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.

   A: True, 分錯的 data weight 計算都是 (original weight) * exp(+alpha), 而 alpha 對同一輪的資料來說是一樣的, 因此 exp(+alpha) 對同一輪的資料來說都是一樣的 (乘上同一個數 = go up by the same multiplicative factor)

   b. 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.

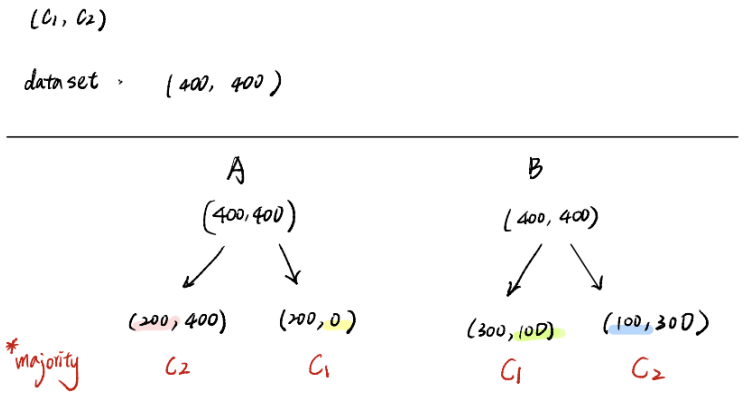
   A: True, In the course of boosting iterations the weak classifiers are forced to try to classify more difficult examples. The weights will increase for examples that are repeatedly misclassified by the weak classifiers. The weighted training error ε_t of the t_{th} weak classifier on the training data therefore tends to increase.

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

A: False, 不一定，如果 training data 無法被我們選用的 weak classifier 中的幾個 weak classifier 線性組合出來的分類器好好分開的話，那就不能達到 zero error。

3. 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, . . . , K.

1. misclassification rates



① misclassification rate

A: C_1 but misclassified to C_2 = 200

C_2 but misclassified to C_1 = 0

misclassification rate $\frac{200+0}{800} = \frac{1}{4}$

B: C_1 but misclassified to C_2 = 100

C_2 but misclassified to C_1 : 100

misclassification rate $\frac{100+100}{800} = \frac{1}{4}$

A & B's misclassification rate is the same.

2. entropy

② entropy before split

$-\left[\frac{400}{800}\log_2\frac{400}{800} + \frac{400}{800}\log_2\frac{400}{800}\right]$

$= 1$

A: after split

left node :

$-\left[\frac{200}{600}\log_2(\frac{200}{600}) + \frac{400}{600}\log_2(\frac{400}{600})\right]$

$= -\left[\frac{1}{3}\log_2(\frac{1}{3}) + \frac{2}{3}\log_2(\frac{2}{3})\right]$

$\fallingdotseq 0.918\,2958$

right node :

$-\left[\frac{200}{200}\log_2(\frac{200}{200}) + \frac{0}{200}\log_2(\frac{0}{200})\right]$

$= 0$

left & right weighted :

$\fallingdotseq 0.918 \times \frac{600}{800} + 0 \times \frac{200}{800}$

$= 0.6885$

$\fallingdotseq 1 - 0.6885 = 0.3115$ gain (before - after)

B: $-\left[\frac{400}{800}\log_2\frac{400}{800} + \frac{400}{800}\log_2\frac{400}{800}\right]$ before split

$= 1$

left node : after split

$-\left[\frac{300}{400}\log_2(\frac{300}{400}) + \frac{100}{400}\log_2(\frac{100}{400})\right]$

$= -\left[\frac{3}{4}\log_2(\frac{3}{4}) + \frac{1}{4}\log_2(\frac{1}{4})\right]$

$\fallingdotseq 0.811\,2781$

right node :

$-\left[\frac{100}{400}\log_2(\frac{100}{400}) + \frac{300}{400}\log_2(\frac{300}{400})\right]$

$= -\left[\frac{1}{4}\log_2(\frac{1}{4}) + \frac{3}{4}\log_2(\frac{3}{4})\right]$

$\fallingdotseq 0.811\,2781$

weighted

$\fallingdotseq 0.811 \times \frac{400}{800} + 0.811 \times \frac{400}{800}$

$= 0.811$

$\fallingdotseq 1 - 0.811 = 0.189$ gain

$0.3115 > 0.189$, A win

(info gain 比較大)

3. Gini

③ Gini

before split

$1-\left(\frac{400}{800}\right)^2-\left(\frac{400}{800}\right)^2=\frac{1}{2}$

A:

after split

left node:

$1-\left(\frac{200}{600}\right)^2-\left(\frac{400}{600}\right)^2=0.\overline{4}$

right node:

$1-\left(\frac{200}{200}\right)^2-\left(\frac{0}{200}\right)^2=0$

weighted

$=0.\overline{4}\times\frac{600}{800}+0\times\frac{200}{800}$

$=0.\overline{3}$

gain

$\frac{1}{2}-0.\overline{3}=\frac{1}{6}\doteq 0.1\overline{6}$

B:

before split

$1-\left(\frac{400}{800}\right)^2-\left(\frac{400}{800}\right)^2=\frac{1}{2}$

after split

left node:

$1-\left(\frac{300}{400}\right)^2-\left(\frac{100}{400}\right)^2=0.375$

right node:

$1-\left(\frac{100}{400}\right)^2-\left(\frac{300}{400}\right)^2=0.375$

weighted

$0.375\times\frac{400}{800}+0.375\times\frac{400}{800}$

$=0.375$

gain

$\frac{1}{2}-0.375=0.125$

$0.1\overline{6}>0.125$ A win

Or simply say the impurity after split, A's impurity after split o.3- is smaller than B's impurity after split 0.375, thus A win

---

"entropy" and "Gini" 標準

下衡量，都是A切的比較好 #