NYCU Introduction to Machine Learning, Homework 3

110550126 曾家祐

Part. 1, Coding (50%):

For this coding assignment, you are required to implement the Decision Tree and Adaboost algorithms using only NumPy. After that, train your model on the provided dataset and evaluate the performance on the testing data.

(30%) Decision Tree

Requirements:

- Implement gini index and entropy for measuring the best split of the data.
- Implement the decision tree classifier (CART, Classification and Regression Trees) with the following two arguments:
- criterion: The function to measure the quality of a split of the data. Your model should support "gini" and "entropy".
- 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 to splitting data once.

Tips:

- 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 gini index and the entropy of the array [0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1].

```
gini of [0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1]: 0.4628099173553719
entropy of [0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1]: 0.6554817739013927
```

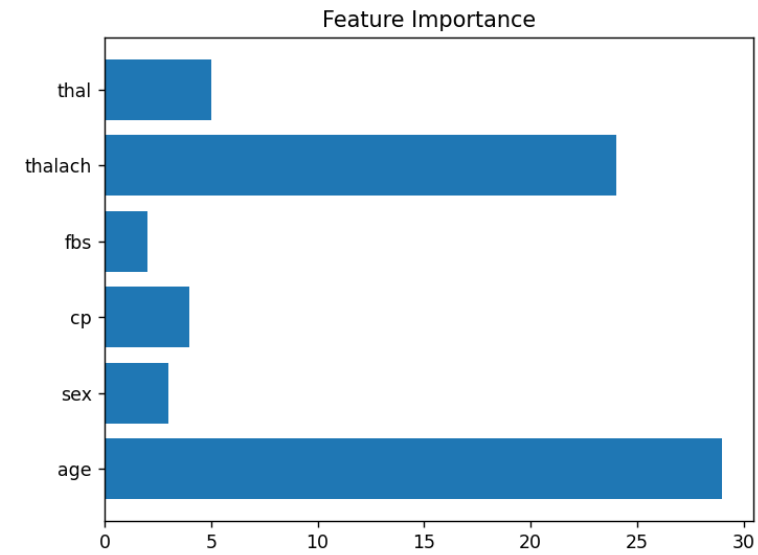
2. (10%) Show the accuracy score of the testing data using criterion="gini" and max_depth=7. Your accuracy score should be higher than 0.7.

```
Accuracy (gini with max_depth=7): 0.7049180327868853
```

3. (10%) Show the accuracy score of the testing data using criterion="entropy" and max_depth=7. Your accuracy score should be higher than 0.7.

```
Accuracy (entropy with max_depth=7): 0.7213114754098361
```

4. (5%) Train your model using criterion="gini", max_depth=15. Plot the feature importance of your decision tree model by simply counting the number of times each feature is used to split the data. Your answer should look like the plot below:



(20%) Adaboost

Requirements:

- Implement the Adaboost algorithm by using the decision tree classifier (max_depth=1) you just implemented as the weak classifier.
- The Adaboost model should include the following two arguments:
- criterion: The function to measure the quality of a split of the data. Your model should sup port "gini" and "entropy".
- n_estimators: The total number of weak classifiers.

Tips:

- You can set any random seed to make your result reproducible.

Criteria:

5. (20%) Tune the arguments of AdaBoost to achieve higher accuracy than your Decision Tre es.

```
218  # AdaBoost
219      print("Part 2: AdaBoost")
220      # Tune the arguments of AdaBoost to achieve higher accuracy than your Decision Tree.
221      ada = AdaBoost(criterion='gini', n_estimators=12)
222      ada.fit(X_train, y_train)
223      y_pred = ada.predict(X_test)
224      print("Accuracy:", accuracy_score(y_test, y_pred))
```

```
Part 2: AdaBoost
Accuracy: 0.8360655737704918
```

Part. 2, Questions (50%):

1. (10%) True or False. If your answer is false, please explain.
   a. (5%) In an iteration of AdaBoost, the weights of misclassified examples are increa sed by adding the same additive factor to emphasize their importance in subsequen t iterations.

      False.

      the weights of misclassified examples are increased, but the increase is not achieved by adding the same additive factor to all misclassified examples. Instead, the weights are adjusted by giving more emphasis to the examples that were miscla ssified in the previous iteration.

   b. (5%) AdaBoost can use various classification methods as its weak classifiers, such as linear classifiers, decision trees, etc.

      True.

2. (10%) How does the number of weak classifiers in AdaBoost influence the model's perfor mance? Please discuss the potential impact on overfitting, underfitting, computational cost, memory for saving the model, and other relevant factors when the number of weak classifi ers is too small or too large.

   Too many weak classifiers:

   1. overfitting:

      If the number of weak classifiers is too large, the model may overfit the tra ining data. The model may start to memorize the training data, capturing n oise and outliers instead of learning the underlying patterns.

   2. computing cost:

      Training a large number of weak classifiers increases the computational co st of the AdaBoost algorithm. Each iteration involves updating weights and training a new weak classifier, and this can become computationally expen sive.

   3. memory for saving the model:

Storing a large number of weak classifiers in memory can be resource-intensive. The model size increases, and the memory requirements for deployment also go up.

4. Diminishing Returns:

   After a certain point, adding more weak classifiers may not significantly improve performance. The marginal gain in accuracy may decrease, and the computational cost continues to rise.

Too less weak classifiers:

1. underfitting:

   If the number of weak classifiers is too small, the model may underfit the training data. It might not have enough capacity to capture the underlying patterns in the data.

2. computing cost:

   When the number of weak classifiers is too small, the AdaBoost algorithm may converge quickly during training since it doesn't have many iterations to adjust the weights and focus on difficult-to-classify examples. As a result, the computational cost may be lower compared to a larger number of weak classifiers.

3. memory for saving the model:

   With fewer weak classifiers, the memory requirements for storing the ensemble model are lower. The model is simpler and requires less memory for deployment.

4. bias may dominated model:

   A model with too few weak classifiers tends to have higher bias, meaning it may oversimplify the underlying relationships in the data. This can result in poor generalization.

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 pr

oposes 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.

I think this is not a good idea. The student's proposal to set m = 1, where m is the number of random features used in each node of each decision tree in a random forest, is likely to have significant drawbacks and is not generally advisable.

Directly set m = 1 can limit the expressiveness of individual trees, and the decision trees m ay not capture the full complexity of the data. The ensemble may lack the ability to learn i ntricate relationships between features, potentially leading to suboptimal performance.
A random forest with m=1 is more susceptible to noise in the data, as each tree is making decisions based on only one feature at each split. This can result in trees that are more sens itive to outliers or irrelevant features.

In summary, Directly set m = 1 is not always a good idea, I think m can be a hyperparamet er, and we can tune it based on different dataset to reach good result.

4. (15%) The formula on the left is the forward process of a standard neural network while th e formula on the right is the forward process of a modified model with a specific techniqu e.
    a. (5%) According to the two formulas, describe what is the main difference between the two models and what is the technique applied to the model on the right side.

    The main difference between two models is y^l, on the right side we changes y^l into $y^l r^l$,r^l is Bernoulli(p). By using Bernoulli distribution, the model will ramdomly drop som e units in the training, set the output to 0 for some neural. This technique is called Dro pout, it helps to prevent overfitting.

b. (10%) This technique was used to deal with overfitting and has many different exp lanations; according to what you learned from the lecture, try to explain it with res pect to the ensemble method.

Ensemble method is combining the predictions of multiple models to improve overall performance.

Dropout can be seen as implicitly conducting **model averaging**. During training, in eac h iteration, due to the random dropout, we train a **slighty different model** base on differ ent subset. This can be seen as we train on different data and features. In testing, we di dn't dropout any neural. This can be seen as an **average over these different models.**

In summary, dropout in neural networks introduces stochasticity during training, effect ively creating an ensemble of subnetworks within a single model architecture. This int ernal ensemble effect mimics the principles of ensemble learning, leading to improved generalization, robustness, and resistance to overfitting.

$$z^{(l+1)} = w^{(l+1)} y^l + b^{(l+1)}$$

$$y^{(l+1)} = f(z^{(l+1)})$$

$$r^l = Bernoulli(p)$$

$$\tilde{y}^l = r^l y^l$$

$$z^{(l+1)} = w^{(l+1)} \tilde{y}^l + b^{(l+1)}$$

$$y^{(l+1)} = f(z^{(l+1)})$$