NYCU Introduction to Machine Learning, Homework 3

Deadline: Nov. 28, 23:59

Part. 1, Coding (50%):

(30%) Decision Tree

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.

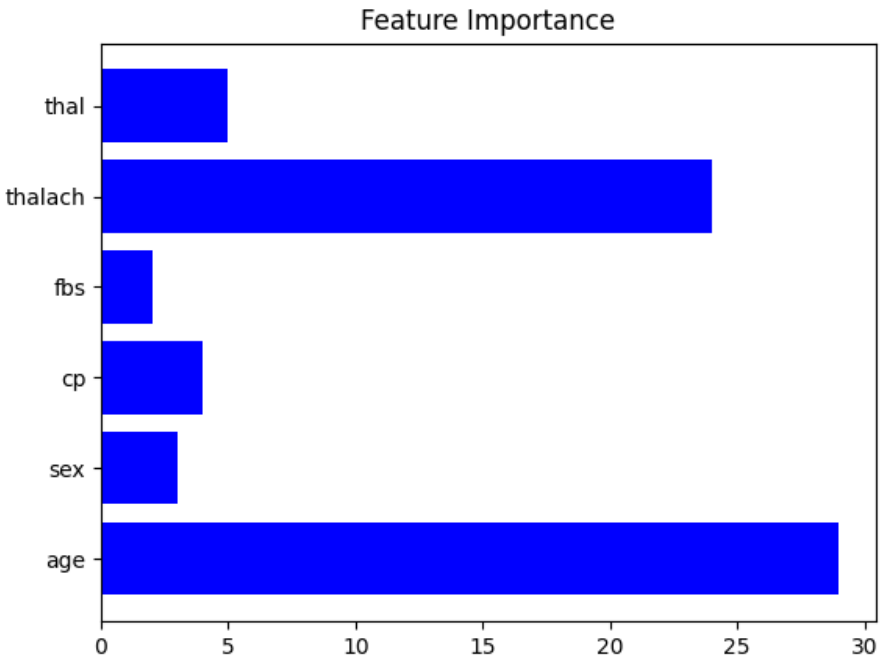
```
Accuracy (gini with max_depth=7): 0.7213114754098361
```

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.

```
284    tree = DecisionTree(criterion='gini', max_depth=15)
285    tree.fit(X_train, y_train)
286    tree.plot_feature_importance_img(["age","sex","cp","fbs","thalach","thal"])
```



(20%) Adaboost

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

Trained using criterion ‘entropy’ and 33 weak classifiers, hyper parameters shown in below image.

```
ada = AdaBoost(criterion='entropy', n_estimators=33)
ada.fit(X_train, y_train)
y_pred = ada.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
```

```
Part 2: AdaBoost
Accuracy: 0.8032786885245902
```

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 increased by adding the same additive factor to emphasize their importance in subsequent iterations.

False, for those misclassified samples we multiply its weight instead of adding the same additive factor. After the weight renewal it is then used to train the next weak classifier, giving the previously misclassified samples a higher chance of correct classification.

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

True, when referring to the choice of Adaboost weak classifiers there is only one constraint, it has to perform better than guessing with pure luck, providing a prediction accuracy slightly higher than 50% would be enough. Therefore, the above mentioned weak classifiers, linear classifiers, decision trees, are all valid choices for weak classifiers.

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

Similar to manipulating the epoch parameter of a linear regression model, raising or lowering the number of weak classifiers in an Adaboost Model impacts its performance in several aspects.

1. Overfitting and Underfitting
   Insufficient amount of weak classifiers will lead to underfitting . The model may not have enough capacity to capture the complexity of the underlying relationship in the data. On the other hand, overfitting may occur when the number of classifiers is more than enough. The model may then become too specific to training data and relatively weak to classifying future unseen data.
2. Computational cost
   Computational cost depends heavily on the number of classifiers, since it decides the size of the outer loop a model has to iterate to find the suitable weight for each classifier. Thus, the higher the number of

weak classifiers the higher the computational cost needed to fit the model.

3. Memory Requirements
   The number of features in the adaboost model depends on the number of weak classifiers it need. We would need more memory space to store the additional classifiers if the number of classifiers were raised, and less space if weak classifiers were removed.
4. Training Time
   Due to the adaptive attribute of adaboost(subsequent weights depend on previous prediction reslut), it is not feasible to paralyze the fitting process of adaboost. Therefore, the training time heavily depends on the number of classifiers utilized, which in the adaboost case indicates the number of iterations the model has to be trained on. It would be time consuming to fit an adaboost model with high classifier number, and fitting for models with low classifier number will be significantly faster.

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?

No, The student's proposal to set m = 1 in random forests is likely to be counterproductive and may not lead to improved accuracy nor reduced variance. The typical recommendation is to set m to a value smaller than the total number of features. This introduces diversity among the trees, which can help reduce overfitting and increase the robustness of the model. Below are several reasons not to set m = 1:

1. Lack of Diversity:

   With m = 1, each decision tree would be constructed using only a single random feature at each node. This would result in trees that are more similar to each other, rather than diverse, because the possibility of feature choice will be decreased.

2. Increased Correlation:

   The random forest algorithm relies on the principle of decorrelation among the trees. When m is set to 1, the trees become more correlated, as they are all

making decisions based on the same feature at each split. This diminishes the ensemble's ability to reduce variance and make robust predictions.

3. Loss of Information:

Using only one feature at each split might lead to the loss of valuable information contained in other features. The model may not capture complex interactions between different features, limiting its ability to understand the underlying patterns in the data. Potentially harming the accuracy of a random forest.

4. (15%) The formula on the left is the forward process of a standard neural network while the formula on the right is the forward process of a modified model with a specific technique.

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 the two equations comes from the Bernoulli distributed random variable r in the left hand side equation. By multiplying random variable r with y, input to the NN network could be set to 0 when r takes the value 0 in the Bernoulli distribution. This technique is called drop out, which is commonly use in NNs to prevent overfitting. The main idea behind drop out is to randomly set a fraction of the input units to 0 at each update during training.

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

Dropout can be viewed as a form of model averaging. Ensemble methods, like Random Forests, create multiple models and combine their predictions to achieve better generalization performance. Similarly, dropout in neural networks can be viewed as a form of creating an ensemble of different subnetworks.
The dropout process can be seen as training multiple networks simultaneously, with each network representing a different combination of neurons. These subnetworks collectively contribute to the overall learning process.
Ensemble methods often incorporate regularization to prevent overfitting. Dropout serves a similar purpose by introducing noise and preventing the network from relying too heavily on specific neurons.

In summary, dropout in neural networks can be understood as a mechanism for creating an implicit ensemble of subnetworks during training. The diversity introduced by dropout helps prevent overfitting, encourages robust feature learning, and parallels the principles of ensemble methods, contributing to improved generalization performance.

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

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

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

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

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

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