CS405 Machine Learning

Lab #5 Decision Tree

(100 points)

1. Introduction

This lab we introduce the classical machine learning algorithm, decision tree as well as its ensemble learning algorithm, random forest (RF). Decision tree is an important non-parameter learning. Although it is simple and limited, it still can make great power in ensemble algorithm. In this lab we will learn decision tree from the beginning and implement in RF. The scikit learn package is used so you do not need to write your own decision tree from the bottom.

2. Decision tree

It is better to know the basic principal of decision tree before using it. Please refer the *Lab05_DecisionTree.pdf* or other documents. Now we assume you have been familiar about decision tree. Firstly, we load the scikit learn iris toy dataset as an example. For visualization, only the last 2 attributes are used.

```
import numpy as np
import matplotlib.pyplot as plt
```

```
from sklearn import datasets

iris = datasets.load_iris()
X = iris.data[:, 2:]  # choose 2 of attributes for visualization
y = iris.target
```

```
plt.scatter(X[y==0, 0], X[y==0, 1])
plt.scatter(X[y==1, 0], X[y==1, 1])
plt.scatter(X[y==2, 0], X[y==2, 1])
plt.show()
```

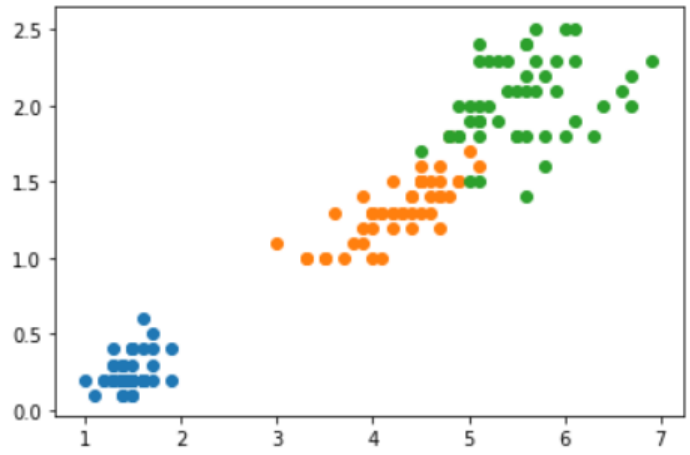

Fig. 1

Here we import the sklearn decision tree model and train.

```
from sklearn.tree import DecisionTreeClassifier

dt_clf = DecisionTreeClassifier(max_depth=2, criterion="entropy")
dt_clf.fit(X, y)
```

```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=2,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best')
```

Fig. 2

Define the visualize function.

```python
def plot_decision_boundary(model, axis):

    x0, x1 = np.meshgrid(
        np.linspace(axis[0], axis[1], int((axis[1]-axis[0])*100)),
        np.linspace(axis[2], axis[3], int((axis[3]-axis[2])*100))
    )
    X_new = np.c_[x0.ravel(), x1.ravel()]

    y_predict = model.predict(X_new)
    zz = y_predict.reshape(x0.shape)

    from matplotlib.colors import ListedColormap
    custom_cmap = ListedColormap(['#EF9A9A','#FFF59D','#90CAF9'])

    plt.contourf(x0, x1, zz, cmap=custom_cmap)
```

Fig. 3

```python
plot_decision_boundary(dt_clf, axis=[0.5, 7.5, 0, 3])
plt.scatter(X[y==0, 0], X[y==0, 1])
plt.scatter(X[y==1, 0], X[y==1, 1])
plt.scatter(X[y==2, 0], X[y==2, 1])
plt.show()
```

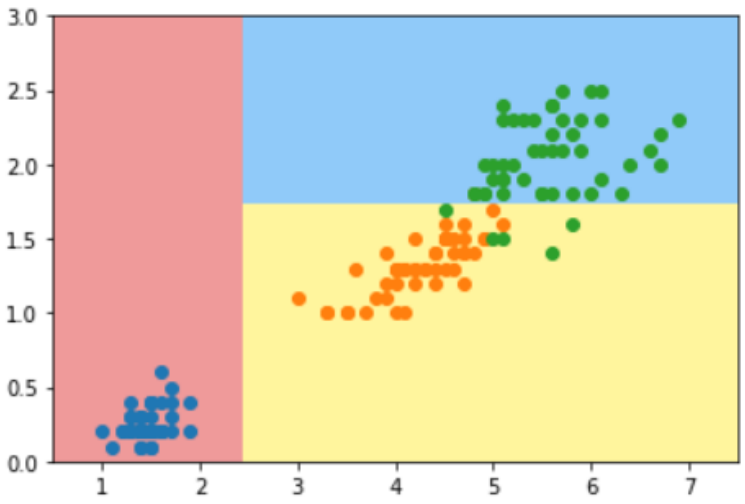

Fig. 4

The dataset is divided by some vertical and horizontal lines and that is a limit of decision tree, we can not generate a lean boundary in decision tree.

We can also see our trained model in a “tree” structure.

```
1  from sklearn import tree
2
3  tree.plot_tree(dt_clf)
```

```
[Text(133.92000000000002, 181.2, 'X[0] <= 2.45\nentropy = 1.585\nsamples = 150\nvalue = [50, 50, 50]'),
 Text(66.96000000000001, 108.72, 'entropy = 0.0\nsamples = 50\nvalue = [50, 0, 0]'),
 Text(200.88000000000002, 108.72, 'X[1] <= 1.75\nentropy = 1.0\nsamples = 100\nvalue = [0, 50, 50]'),
 Text(133.92000000000002, 36.23999999999998, 'entropy = 0.445\nsamples = 54\nvalue = [0, 49, 5]'),
 Text(267.84000000000003, 36.23999999999998, 'entropy = 0.151\nsamples = 46\nvalue = [0, 1, 45]')]
```

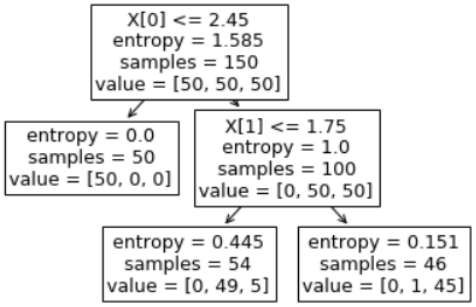

Fig. 5

Problem: Using Decision tree is quite easy, but it is very likely to overfit. Actually almost all the non-parameter learning have this problem. We can use pruning to optimize our trained decision tree. And we can also adjusting the super parameters to avoid overfitting.

Exercise 01:

When we create a decision tree in Fig. 2, we only use two of the arguments. Actually, there are so many arguments listed below and they are all helpful in adjusting the parameters to achieve the balance between bias and variance. Please check for the meaning of each arguments and explain how it effect the result about bias and variance.

3. Random forest

In *Lab05_DecisionTree.pdf* we have introduced that RF is bagging plus decision tree. Thus, in scikit learn, there are two ways to implement an

RF, one from the Bagging view and the other just the RF directly. But first, we create a random dataset for visualization.

```
# Create random dataset
X, y = datasets.make_moons(n_samples=500, noise=0.3, random_state=42)
```

```
plt.scatter(X[y==0, 0], X[y==0, 1])
plt.scatter(X[y==1, 0], X[y==1, 1])
plt.show()
```

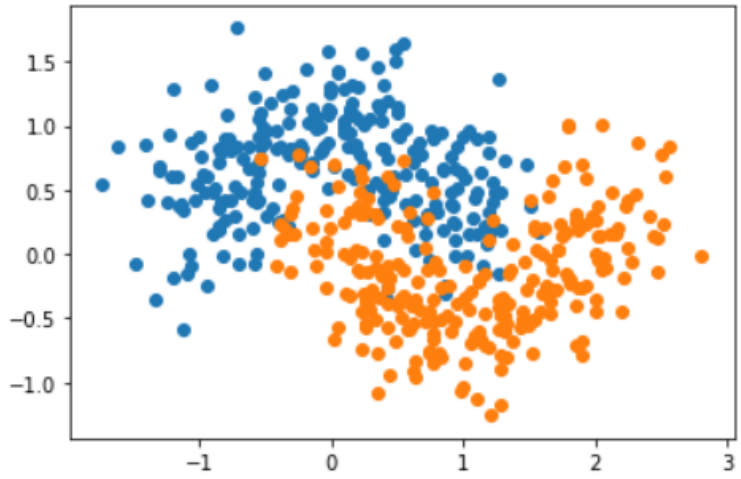

Fig. 6

Bagging view: use bagging algorithm and all base learning algorithms are decision trees.

```
# Use Random Forest from Bagging view

from sklearn.ensemble import BaggingClassifier

bagging_clf = BaggingClassifier(DecisionTreeClassifier(),
                                n_estimators=300,
                                max_samples=300,
                                bootstrap=True, # using bootstrap sampling method
                                oob_score=True, # use oob data for scoring
                                # n_jobs=-1  # use paralell computing
                               )
bagging_clf.fit(X, y)
```

Fig. 7

Because we use oob_score (out-of-bag data for score), so we do not need to split a separated testing set.

```
# Output oob score
bagging_clf.oob_score_
```

```
0.916
```

Fig. 8

See the result.

```
plot_decision_boundary(bagging_clf, axis=[-2, 3, -1.5, 2])
plt.scatter(X[y==0, 0], X[y==0, 1])
plt.scatter(X[y==1, 0], X[y==1, 1])
plt.show()
```

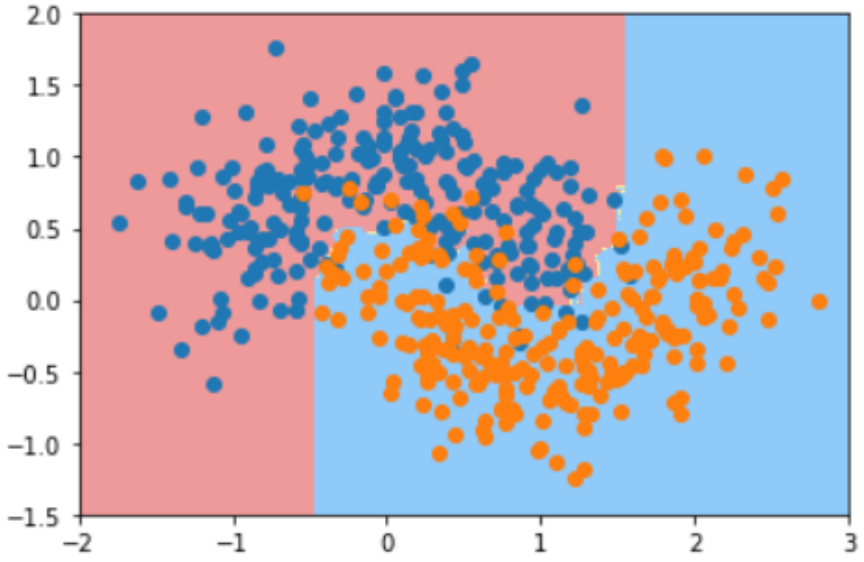

Fig. 9

RF view: scikit learn has built RF class

```
# Use Random Forest directly

from sklearn.ensemble import RandomForestClassifier

rf_clf = RandomForestClassifier(n_estimators=300,
                                random_state=666,  # random attributes subset
                                oob_score=True,
                                # n_jobs=-1
                               )
```

Fig. 10

There are many arguments for either base decision tree or whole ensemble algorithm. A good ensemble algorithm should make sure that every base ones are both accurate and diversiform. So it is better to get a

good enough base tree parameters before training your ensemble learning algorithm.

Than we see the results.

```
plot_decision_boundary(rf_clf, axis=[-2, 3, -1.5, 2])
plt.scatter(X[y==0, 0], X[y==0, 1])
plt.scatter(X[y==1, 0], X[y==1, 1])
plt.show()
```

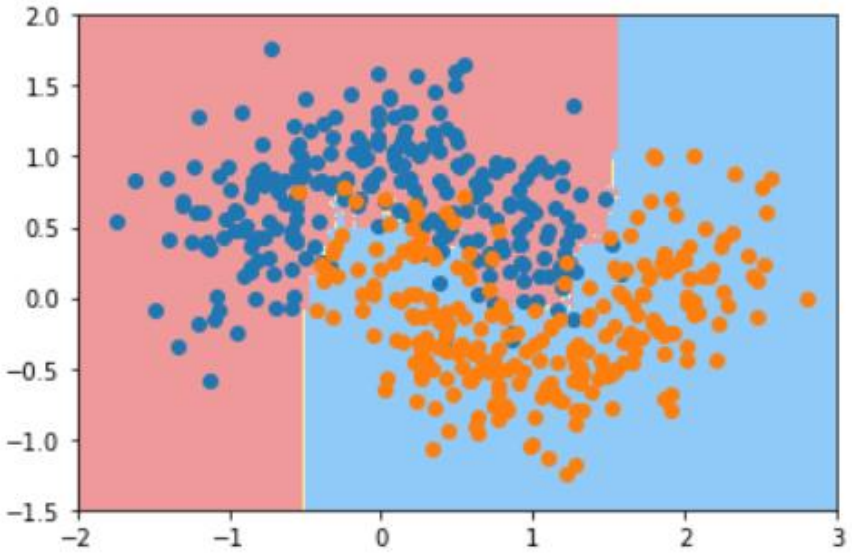

Fig. 11

Exercise 02:

Try to figure out the meaning of each arguments of BaggingClassifier, and discuss the effect to the bias and variance.

4. Lab requirement

Please answer the Exercise 01 and 02 questions and wright a report.

5. References

https://scikit-learn.org/stable/