Project4a Decision Tree

I directly modified the Testing.py, to run the following test, just run the Testing.py.

Q6: Performance

Dummy Set1:

Tree size: 3, Average Classification Rate 1.0

Analysis:

The tree size is very small, because in the training set the label depends only on the attribute 5. More specifically, the output label is LOGICAL NOT (Attribute 5). Therefore, the tree size is small.

The classification rate is 1.0 which is not very common in real case, it is because the test set have the exact same pattern (output is logical NOT of the attribute 5) as the training set.

Dummy Set2:

Tree size 11, Average Classification Rate 0.65

Analysis:

Compared to Set1, the tree size is larger. Because there is no "obvious" correlation between the attribute and the label, we use the information gain to choose the best split attribute at each stage and result in a size-11 tree.

As for the classification rate, it is only 0.65, because the training data set is too small, it only contains 20 examples, which is barely equal to the size of the test set! Therefore, the model easily overfits to the training set, therefore it performs bad on test set.

Connect4:

Tree size 41521, Average Classification Rate 0.757

Analysis:

The tree size is 41521, which is very large, because the state space (3 to the power of 43) for the game is very large. And the large tree size also tells us that our model is probably overfitting. Because the problem is very hard to be solved by decision tree, because of its large space and the representation of the problem (using the state of each cell) is too complicated and the individual cell can not tell us much information.

Although we have a lot of training examples, the classification rate is not that high, because the dataset cannot generalize well due to the poor representation of state. And each case is kind of unique under this representation, even change one attribute of the

42 attributes may result in a completely different game result. Therefore, the model cannot perform well.

Car:

Tree size 408, Average Classification Rate 0.941750

Analysis:

The tree size is within an acceptable range this time. The tree is not overfitting to the training data. And it is larger than the previous tree trained on dummy set because this time we have more complex data. Therefore, our tree need to consider more cases, which will result in a larger tree.

As for the classification rate, it is the highest among the four, because this time we have enough training data (1728 examples), and our tree is not overfitting to the training data (the tree size is not incredibly large, and as the classification rate can show, it is not overfitting).

Q7: Applications

Car:

Q: Explain how their Decision Trees could be used along with other software (another algorithm, or a user-facing GUI) to solve some problem:

A: I think the decision tree trained on the car dataset can be embedded in to an application which takes the input information from the user (Buying, Maint, Doors, Persons, Lug_boot, Safety), and evaluate a car to tell the user whether this car is worth to buy or not (unacc, acc, good, vgood).

Q: Suggest a similar dataset and analyze how having a classifier such as a Decision Tree could be useful to something like a website selling products:

A: Suppose we have a house dataset. The attribute can be:

Buying ability: vhigh, high, med, low.

rooms: 2, 3, 4, 5more.

beds: 2, 3, 4, 5more.

Price: vhigh,high, med, low

Size of the house: small, med, big.

Safety of the neighborhood: low, med, high.

And the class values are unacceptable, acceptable, good, very good.

It can be used in a house renting/selling website, where user will input the attribute, and

the website will evaluate the houses and return the list of houses based on the class values of each house.

Connect4:

Q: Explain how their Decision Trees could be used along with other software (another algorithm, or a user-facing GUI) to solve some problem:

A: We can create an APP with a game agent based on the decision tree. The decision tree will evaluate the game state and return the labels for each action in the action list of the game agent. It can be use as the Q-value of the game agent, or it can be used as heuristic function for A* search game agent.

Q: Suggest some way in which the classifier could be incorporated with one of the past algorithms we have learned about to make a better Connect4 playing bot.

A: As mentioned in previous question, it can be incorporated with A* search of the game agent as the heuristic function for each state.