Report

Qinyun Song

My codes can be divided into four parts. The first part is the class handling the people. The second part is the classes handling the features. The third part is the structure handling the tree. And the fourth part is the main program. By doing this, I can encapsulate many operations to the first three parts. Thus makes my main program much simpler. I am going to introduce these four parts.

1 People

To handling the people, I designed a class called People. It contains the following functions:

1. Recording the name and the label of each person.
2. Calculating the entropy of these people.
3. Find the most common label among these people.

Everytime I need the information about a group of people, this class can calculate information for me.
This class is implemented in the file *people.h, people.cpp*.

2 Feature

2.1 BasicFeature

I noticed that, the feachers all need similar functions, like calculating the information gain, dividing the current people. So I first implement a base class called BasicFeature. It contains the following three functions:

1. Returns the information gain.
2. Divide a group of people.
3. Find the subgroup one person belongs to.

The three functions above are all defined as virtual functions. The reason doing that is that, I can control different features only using the pointers to this base class. This is done by polymorphism.
This class is implemented in the file *basicfeature.h*.

2.2 fSum

This class is designed to handle one feature: the sum of the characters in one's name. It derived the function from the base class and it will calculate the corresponding result based on its own conditions.
This class is implemented in file *fSum.h, fSum.cpp*.

2.3 fLength

This class is used to control the feature: length of the name. It is also derived from the base class and will implement the required three functions based on its own requirements.
This class is implemented in the file *flength.h, flength.cpp*.

2.4 fThirdChar

This class is used to divide the person by the fourth character in their names. It derived the base class. It contains 27 classes. The first 26 classes correspond to the 26 letters. And the last class is used to contain other characters. It will also implement the required functions.
It is defined in the file *fthirdchar.h, fthirdchar.cpp*

2.5 fOdd

This class is defined to control the feature: The sum of the first name of one person is odd or even. It also derived the base class and will calculate the results for the required functions.
This class is implemented in the file *fodd.h, fodd.cpp*.

2.6 AllFeatures

This class is used to handle different features. Since we may need same kind of information from different features, I use this class as the entrance of sending and getting the require information. It records four pointers in the base class type pointing to the above four different features. So for the required three functions, it will choose the desired feature and return the number from that feature.
This class is implemented in the file *allfeatures.h, allfeatures.cpp*.

3 Node

This class is defined to build the tree structure of the desicion tree. It contains the following information of a node in the tree:

1. A vector of nodes of its subtrees. Since it may not be a binary tree, I used a vector to hold pointers to all its subtrees.

2. A number to record what features are not used till now. So at each node, I can enumerate all the unused features to build the tree.

3. A number to record the finally chosen feature. So that when I want to test one name, I can find the corresponding featuer and calculate which group it belongs.

4. The label of the node. This is only used when the node is the leave of the tree so that it can return the result of the prediction.

This class is defined in the file *node.h, node.cpp*

4 Main Program

The main program is used to build a new decision tree and predict the label of names. It will first read all the names for training and use a recurrent function to build the tree. During the recursion, it will first enumerate all possible features and find the best one. Then it will divide the people to several groups and create subtrees for each of the group. At last, it will call the function to build the tree of its subtrees.
For the testing, it will also read all the names and search in the decision tree for each name. It will find the corresponding label of the name and compare with the correct label. It will also report the accuracy of the whole test data.
This is implemented in the file *DT.cpp*