Eli

Random Forest Classification

4/30/16

FUNCTIONAL OVERVIEW

The project will be to implement a classifier which employs decision trees. The program will run from the command line interface. Data will be accepted in the form of text file prescribing to some (yet to be determined) formatting. Outputs will be saved as text files (also in a yet to be determined format). Users will be able to save the Decision Trees for later used as well. The is expected to be fairly robust allowing it to be trained and theoretically be used for many different classification problems.

DESIGN OVERVIEW

The user will provide an input file that contains a data set that is comprised different objects with a set of values along with an identifier of the objects class. My program will create a Random Forest (a set of decision trees) based of the sample data set to perform classification for other objects.

Once the Random Forest has been created it will be saved in memory. Users will then be able to feed in unclassified data sets. These data sets will be fed through the decision trees. A combination or mean of the classifications each tree in the Random Forest produces will be used to determine the classification of a given object. The results of the classification process will be saved in a text file. Objects to classify may also be passed in through the command line; in this case the resulting classification will be displayed on the command line.

PRIORITIZATION OF FEATURES

Features in a ranked list from highest priority to lowest:

1. Program can create at least make one functioning decision tree from a data set
2. User enters data set by naming a file contains the data
3. Classification objects is displayed on the command line
4. Implements a fully functional Random Forest
5. Program can output the classification of a data set to a text file
6. Handling both continuous and discrete attributes

DESIGN DETAILS

The classifier will be written in Java, and consist of two major parts: the implementation of the Random Forest and a system to parse input data from a text file to be used to create the decision trees.

Each tree in the Random Forest will be grown according to the following algorithm:

1. Select a random subspace of the input set S this will be the data used to generate the decision tree
2. If there are N input variables for each object in S specify a number M < N so that at each node, M parameters out of N are selected randomly and the best split on these variables are used to split the data at his node. M should be determined using the out-of-bag error rate (oob) so that the error of the Random Forest is minimized. M should also be held constant for all trees in the forest
3. Grow each tree to its largest extent, do not prune

Keeping the above algorithm in mind each individual decision tree will be grown using the ID3 algorithm.

Brief Summary of the ID3 algorithm:

1. Calculate the entropy of every variable contained in each object of set S
2. Split S into subsets using the variable for which entropy is minimum (or, equivalently, information gain is maximum)
3. Create a branch node containing information on how S was split at that point
4. Recurse on subsets using remaining variables.

More detail on how entropy is used to determine how to split data:

The information gain is based on the decrease in entropy after a dataset is split on an attribute. Constructing a decision tree is all about finding attribute that returns the highest information gain (i.e., the most homogeneous branches). Information gain can be defined as the difference between the entropy before splitting the data on an attribute and the entropy of that attribute. Entropy of a given attribute, A, is given by the following equation. Where m is the total number of classifications the given data set includes.

We also need to determine the entropy after splitting the data base of one attribute. This is given by the following equation, where is the probability of a given case of X.

The information gain of a given split S is defined by the following:

ID3 uses a greedy algorithm to find the best possible split, calculating the information gain for every possible split and then finding the minimum. This shouldn’t be an issue if the data sets being used define their attributes discreetly. Time efficiency arises if we are trying to deal with continuous attributes because there are too many ways the split the data at each node. To deal with this I am going to add a feature of the C4.5 algorithm. If we are dealing with continuous attributes, then I will define a threshold for the attributes so that continues attributes can be split into discrete sections.

I will create Class’s for an attribute or variable of an object to be classified (Attribute.java), a Class for discrete attributes (DiscreteAttribute.java extends Attribute), a Class for a continuous attribute (ContinuousAttribute.java), a Class for handling the entropy calculations necessary to create a decision tree (Entropy.java), a Class for and individual Node in a decision tree (Node.java), an individual decision tree (DecisionTree.java), and a Class for complete Random Forest (RandomForest.java).

TESTING

 How will you test your program?

 Give specific test cases, particularly “edge cases” that may cause trouble.

 How will you test with other users?

o List four specific questions you will ask them about your program.

GRADING RUBRIC

Write your own grading rubric (out of 40 points) that takes into account whether

a) you matched your functional specification (does your program works as planned)

b) is your design a good one? Is it efficient? What does efficient mean in this language or program?

c) is your code well documented and readable?

d) how well was your code tested?

e) what else could you be graded on that isn’t listed here? Dig deep, be creative.

Note that this does NOT include the grade for this design.

A good way to think about this is that our prior assignments had about 15% for style and documentation, 40-50%

for external correctness and the rest for internal correctness and efficiency.

PROPOSED IMPLEMENTATION SCHEDULE

Plan for this to be completed by the Code Review date. Include when you will do your learning (if necessary),

testing, and your presentation.

POTENTIAL SHOWSTOPPERS

 List any problems you think could come up in your project.

OPEN QUESTIONS

 Any other questions or concerns?

RESOURCES

List and resources you are currently using or expect to use during the course of this project.