Data mining Assignment

#2 Decision Tree

2013011402 jangho park

date: 2016-05-20

2016

* Purpose

1. Background

Decision Tree is one of the most interpretable classification methods. However, decision tree suffers due to some problems such as overfitting problem and attribute selection problem. To overcome these shortcomings, we will use the bagging method, which is one of the ensemble methods, and also information gain is used.

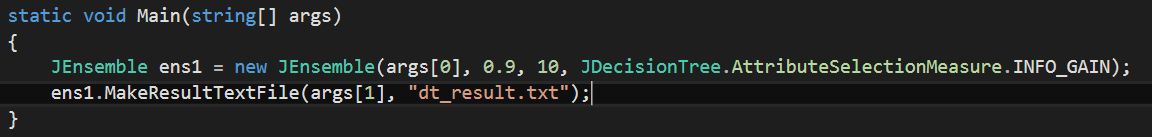
1. Our goal

Given training data, this program will make ensemble, which is composed of decision trees. After that, test input file will be given to this ensemble. Then “dt\_result.txt” will be made.

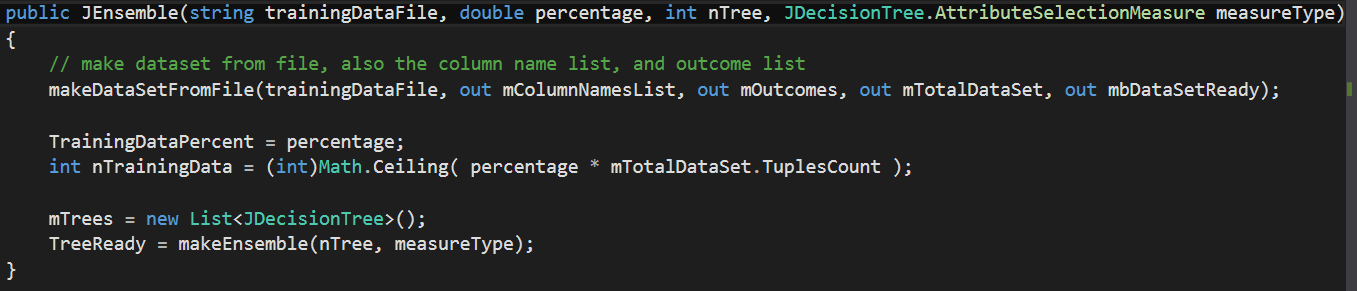
1. Data Structures and Its Public Method Interfaces
   1. **JTuple**: It represents a tuple in training data set or test data set. It contains dictionary data structure, key means attribute name and value means its corresponding value.
      1. JTuple(): constructor, it gives you a single newly allocated tuple object.
      2. JTuple(JTuple srcTuple): copy constructor
      3. ClassLabel: The class label of this tuple.
      4. Operator[]: It is used to access the value of given attribute name.
   2. **JDataSet**: It is a collection of tuples. It will be used to make a decision tree.
      1. InsertTuple(JTuple tuple): Insert tuple to this JDataSet
      2. JDataSet(): constructor
      3. JDataSet(JDataSet sourceDataSet): Copy Constructor.
      4. Operator[n]: It is used to retrieve JTuple object which is located at index ‘n’.
      5. SameClass: It is the property indicating whether tuples of JDataSet is of same class.
      6. TuplesCount: It retrieves the # of tuples in this JDataSet.
      7. MajorityClass: It indicates what is the majority class.
      8. TuplesList: It is the list of tuples in this JDataSet.
   3. **JTreeNode**: It represents a single tree node. There are two types of this node. One is ‘TEST’ node, and the other is ‘RESULT’ node. ‘TEST’ node means this node is used to test input tuple such as ‘age>=30’. On the other hand, ‘RESULT’ node contains the class label such as ‘YES’ or ‘NO’.
      1. JTreeNode(JTreeNodeType type, string value): Constructor. ‘type’ is the type of this node such as ‘RESULT’, or ‘TEST’.
      2. JTreeNode(): Constructor 2.
      3. Operator[outcome]: It is used to get the value of attribute which is given by outcome.
      4. NodeType: It is the property which indicates the type of this node.
      5. Indicator: It is the value of the node. It depends on the type of the node. When the node type is ‘TEST’, it can be the expression like ‘age>=30’. Otherwise, it is the class label.
      6. TuplesCount: # of tuples which are managed by this tree node.
   4. **JDecisionTree**: It represents a decision tree that is made from JDataSet. It contains root node of decision tree. A tuple can be tested using this single decision tree.
      1. JDecisionTree(columnNameList, OutcomesPerAttribute, trainingDataSet): Constructor. ‘columnNameList’ is the list of each attribute name. ‘OutcomesPerAttribute’ is the dictionary whose key means the attribute name and the value is the set of all possible corresponding attribute values. ‘trainingDataSet’ is the data set used to make this tree.
      2. MakeTree(measureType): It makes the decision tree. ‘measureType’ indicates what type of attribute selection measure will be used such as information gain or gain ratio.
      3. TestTuple(tuple): ‘tuple’ is tested using this decision tree and it returns the class label name.
      4. MadeComplete: It is the property which indicates whether this decision tree is made or not.
   5. **JEnsemble**: It is an ensemble object which manages a collection of decision tree. If a tuple can be examined using this object.
      1. JEnsemble(trainingDataFile, percentage, nTree, measureType): This is the constructor. ‘trainingDataFile’ is the name of the training text file. ‘percentage’ means what percent of training data file is used to make each decision tree. ‘measureType’ is used to designate attribute selection method such as information gain, and gain ratio.
      2. DecisionTreeCount: It is the # of decision tree that are included in this ensemble.
      3. TrainingDataPercentage: It indicates what percent of training data set will be used to train each decision tree.
      4. TrainingTuplesCount: It is the # of training tuples to train each decision tree.
      5. TreeReady: It indicates whether all the decision trees are made or not.
      6. DataSetReady: It indicates if the data set is ready or not. It is needed to be checked before making each decision tree.
2. Interconnection Between Components.



1. Procedure
2. In Main(), allocate new JEnsemble Object. Here we gave input training data name(args[0]), percent(0.9), # of trees(10), and attribute selection measure is information gain.



1. Then the constructor calls makeDataSetFromFile() in order to make data set. After that call makeEnsemble(). This function make complete ensemble object.



1. After the allocation of JEnsemble object. You can call its member function whose name is MakeResultTextFile(). This function takes 2 arguments. The first argument is the name of test file, and the other argument is the name of the output text file.
2. Results
   1. When we only use information gain, not use bagging.



* 1. When we use bagging and attribute selection measure is information gain.



🡪 # of trees = 10000, percentage = 90%, Information gain.

* + I also tested using gain ratio method, but it’s very similar to the above. Therefore, I choose information gain as the main measure.
  + It’ll take some times to get result file because the time complexity of bagging method.