Making a SLIQ and Scalable Classifier for Data Mining

Introduction

Classification is an important problem in the field of data mining. It allows for the extraction of relations between attributes as well as give a method to assigning and classifying new data. With the explosion of data accumulating in databases, data mining techniques such as classification become critical in finding "passive data" and changing it into useful information like trends and tendencies.

Classification can be described by taking in a training set which has many records, multiple attributes, and a class label. The classifier's task is to analyze the input data and generate an accurate model for each class in the data. This model is then used to classify test data where the class labels are unknown. This can be used to better understand the data and assist in things such as medical diagnosis, target marketing, and treatment effectiveness.

One of these classification models is a decision tree. For every internal node in the tree, it will represent an attribute. Every branch represents the values of the attribute node, and the leaf nodes represent class labels or distributions. Every record or tuple can follow one of the paths down and get classified. An example of one such classifier is the ID3 algorithm, where it selects the root and subsequent nodes based on information gain and entropy.

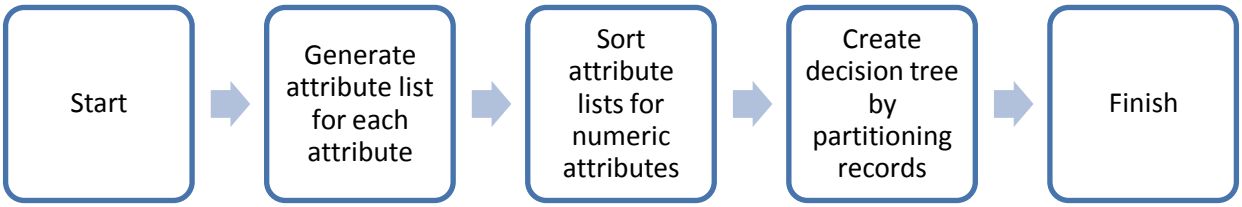
Although ID3 is an effective algorithm, it has a few shortcomings. The major source of contention comes from its lack of scalability. For every stage of constructing the decision tree, there is a very large amount of computation required. As well, ID3 requires all of the training data to reside in memory. Lastly, ID3 does not handle splitting standard indexes for range based attributes.

In order to deal with ID3's shortcomings, there is an algorithm called Supervised Learning In Quest, or SLIQ in short. SLIQ is capable of dealing with both numerical and categorical attributes. It can also build compact and accurate trees using a pre-sorting technique in the tree growing phase and a cheap pruning algorithm. It can also scale well because it doesn't require the

training data to be in memory. As such, the number of classes, attributes and records may vary and grow without too much penalty.

Design

SLIQ follows a general methodology as shown below:



After the data is loaded in, all records will be assigned a unique id and every attribute will have its own table. If the attribute happens to be numeric, they are sorted in descending order. After that, it will begin to generate the decision tree by partitioning records. It will do that by determining the best split using the attribute results and split the records at the examined node to generate child nodes. The determination is based on information entropy. It will then continue iterating through each leaf node and examine until every leaf contains records of one class.

SLIQ typically creates a new table for each attribute and a table for the class. (ie. < rid, attr1>, <rid, attr2>, ...<rid, attrN> and < rid, class, leaf>) Every iteration you need to match the rids in the attribute table to the class table. Sprint on the other hand creates a new table for each attribute with its associated class. (ie. <rid, attr1, class>, <rid, attr2, class>, ... , <rid, attrN, class>). This strategy doesn't require a rematching to the class.

Implementation

For implementation, we decided to use Python as our language because Python can allow us to prototype and get a program up and running fast as compared to a compiled language. Since Python is interpreted on the fly, we do not need to compile every time, so we can easily prototype and implement features quickly.

Our design only uses two tables - the attribute table and the class table. (ie. <rid, attr1, attr2, ..., attrN> and <rid, class, leaf>) We decided to do this approach because Python allows to very easily sort the attribute table by column. As well, we can use recursion to only deal with the records that are relevant to the leaf. We can easily use only the records belonging to a specific node to evaluate the split and not have to walk through the entire table. As opposed to the SLIQ style of a table per attribute, we saw that having an entire attribute table would be better because it is space efficient and the ability for Python to only look at the relevant attributes justified using a single table.

We have two files. sliq.py is a general purpose wrapper that will take in from standard input an arff file. It will then parse it into a list of lists (essentially a 2 dimensional array) and pass it into the classlist class. Classlist.py contains the classlist class and it implements the sliq algorithms. It will split into two tables - one for attributes and one for classes as well as prepare a root node. It will then perform the SLIQ algorithm via doSLIQ and iterate through the nodes until the algorithm is complete. Since SLIQ doesn't require the expensive and repetitive recomputation of the entropy every iteration, it already has a runtime better than ID3. The algorithm will go at worst case n times, and we can assume the partitioning at every step is essentially constant, so the runtime is O(n) in time and O(n*m) in space. where n is the number of records, and m is the number of attributes and classes.

Input	Seconds
contact-lenses	0.001
diabetes	54.242
glass	0.794
ionosphere	1.769
iris	0.008
labor	0.062
soybean	1.277
vote	0.212
weather	0.001
weather.nominal	0.001

Fig 1. SLIQ performance on the following arff inputs (from weka)

For collaboration, we decided to use Github as a repository for our code. You can find it at https://github.com/jujaga/SENG-474. Using a sample set of the arff files from the Weka package, we were able to successfully generate decision trees from the implemented SLIQ algorithm. As the SLIQ paper mentions, its accuracy relative to ID3 is similar. This means that SLIQ is a low-cost and scalable classifier that can generate relatively accurate trees. Mehta and his colleagues demonstrated with their empirical evaluation that SLIQ has good accuracy, produces small decision trees, and has a relatively short classification time.

Works Cited

M. Mehta, R. Agrawal and J. Rissanen. SLIQ: A Fast Scalable Classifier for Data Mining. *Extending Database Technology*, pp. 18-32, 1996.