Homework 1
CS 6375: Machine Learning

Vibhav Gogate

Due: 11:59 p.m. , September, 16 via E-learning

Inducing Decision Trees

In this homework you will implement and test the decision tree learning algorithm (See Mitchell, Chapter 3). It is acceptable to look at Java code for decision trees in WEKA. However, you cannot copy code from WEKA.

You can use either C/C++, Java or Python to implement your algorithms. Your C/C++ implementations should compile on Linux gcc/g++ compilers.

- Download the 15 datasets available on the class web page. Each data set is divided into three sub-sets: the *training set*, the *validation set* and the *test set*. Data sets are in CSV format. Each line is a training (or test) example that contains a list of attribute values separated by a comma. The last attribute is the class-variable. Assume that all attributes take values from the domain {0,1}.

  The datasets are generated synthetically by randomly sampling solutions and non-solutions (with solutions having class "1" and non-solutions having class "0") from a Boolean formula in conjunctive normal form (CNF). I randomly generated five formulas having 500 variables and 300, 500, 1000, 1500 and 1800 clauses (where the length of each clause equals 3) respectively and sampled 100, 1000 and 5000 positive and negative examples from each formula. I am using the following naming convention for the files. Filenames $train*$, $test*$ and $valid*$ denote the training, test and validation data respectively. $train\_c[i]\_d[j].csv$ where i and j are integers contains training data having j examples generated from the formula having i clauses. For example, the file with filename $train\_c500\_d100$ contains 100 examples generated from the formula having 500 clauses. **Note:** Do not mix and match the datasets. For example, do not train using $train\_c500\_d100.csv$ and test using $test\_c500\_d5000.csv$.

- Implement the decision tree learning algorithm we discussed in class. The main step in this algorithm is choosing the next attribute to split on. Implement the following two heuristics for selecting the next attribute.

    1. Information gain heuristic (See Class slides, Mitchell Chapter 3).
    2. Variance impurity heuristic described below.

       Let K denote the number of examples in the training set. Let K_0 denote the number of training examples that have $class = 0$ and K_1 denote the number of training examples that have $class = 1$. The variance impurity of the training set S is defined as:

       $$VI(S) = \frac{K_0}{K}\frac{K_1}{K}$$

       Notice that the impurity is 0 when the data is pure. The gain for this impurity is defined as usual.

       $$Gain(S, X) = VI(S) - \sum_{x \in Values(X)} \Pr(x) VI(S_x)$$

       where X is an attribute, S_x denotes the set of training examples that have $X = x$ and $\Pr(x)$ is the fraction of the training examples that have $X = x$ (i.e., the number of training examples that have $X = x$ divided by the number of training examples in S).

- Implement the reduced-error post pruning algorithm described in class (see also Mitchell Chapter 3). Use the validation data to do the pruning.

- Implement depth-based pruning by using maximum depth $d_{\max}$ as a hyper-parameter, namely in your decision tree prune all nodes having depth larger than $d_{\max}$. We will assume that $d_{\max}$ takes values from the following set: {5,10,15,20,50,100}. Recall that we tune the hyper-parameters using the validation set.

- Read the Wikipedia article on on Random Forests. If you are interested, you can read the original paper (Breiman L (2001). "Random Forests". Machine Learning. 45 (1): 532.); it is pretty readable. Use either scikit-learn or WEKA implementation of Random Forests and apply it on the 15 datasets. Use default options in Weka or Scikit learn.

- For each algorithm and dataset combination, train using the training set, tune using the validation set and test the accuracy using the test set. To be clear, you will evaluate the following algorithms:

  - Naive Decision tree learner with Entropy as the impurity heuristic
  - Naive Decision tree learner with Variance as the impurity heuristic.
  - Decision tree learner with Entropy as the impurity heuristic and reduced error pruning
  - Decision tree learner with Variance as the impurity heuristic and reduced error pruning
  - Decision tree learner with Entropy as the impurity heuristic and depth-based pruning
  - Decision tree learner with Variance as the impurity heuristic and depth-based pruning
  - Random Forests

  Record the classification accuracy for each dataset and algorithm in a table and then answer the following questions:

  1. Which impurity heuristic (Entropy/Variance) yields the best classification accuracy? How does increasing the number of examples and/or the number of clauses impact the (accuracy of the) two impurity heuristics. Explain your answer.
  2. Which overfitting avoidance method (reduced error pruning/ depth-based pruning) yields the best accuracy? Again, how does increasing the number of examples and/or the number of clauses impact the (accuracy of the) two overfitting avoidance methods. Explain your answer.
  3. Are random forests much better in terms of classification accuracy than your decision tree learners? Why? Explain your answer.

What to Turn in

- Your code and a Readme file for compiling the code. Also a read me file for running your code. We should be able to run all the 6 algorithms that you will implement from the command line and therefore include precise documentation.
- Your report containing the table described above as well as answers to the questions.