CSCI 446 Artificial Intelligence Project 3 Design Report

Roy Smart Nevin Leh Brian Marsh

November 4, 2016

1 Introduction

Introduce Machine Learning Algorithm (MLA)!!!!

2 Datasets

2.1 Dataset Representation

We will define a *datum* to be a vector consisting of the class as the zeroth element and the associated attributes as the rest of the elements. Classes and attributes will be represented by an integer. Continuous data will be binned before it is inserted into each datum. The resolution of the bins will be a variable that will have to be tuned. Each dataset will be represented as a vector of datums.

2.2 Data Imputation

Imputation is the process of approximating missing values in the datasets. To our knowledge there is only one dataset that has real missing values: the Wisconsin Breast Cancer Database. The 1984 United States Congressional Voting Records Database appears to have missing values, but these can actually be interpreted as a stance on a particular issue. Since the breast cancer database has a small proportion of missing values, it is appropriate to simply eliminate datums with missing values. The authors assert that trying to train a MLA with imputed values would only create unnecessary bias in the network.

However, it is a common real-world problem to attempt to classify an unknown, incomplete datum, therefore we will perform imputation on the validation datasets. To approximate the missing values we will first try a hot-deck imputation, where missing attribute values of a given datum are constructed by selecting a random member of that datum's class and copying the value of the attribute. If the hot-deck is unsuccessful, we will attempt to fill in the missing values using a regression model developed in *Mathematica*.

2.3 Cross-validation

To partition the full datasets into test and training datasets, we will use 10-fold cross validation. This method partitions the data into ten *folds* and uses one fold for testing data and the remaining nine folds for the training dataset. This process is repeated nine more times until every fold has been used as a test dataset. We selected this method because it will allow the convergence measurement described in Section 5.3 to be applied over a larger range, which allows us to measure the rate of convergence for each MLA more accurately.

3 Machine Learning Algorithms

3.1 k-Nearest Neighbors

3.2 Naïve Bayes

3.3 TAN

3.4 ID3

3.4.1 Description

ID3(Iterative Dichotomiser 3) is an algorithm that creates a decision tree and uses it to make inductive inferences about a set of data. The decision tree approach to making these inferences is a fairly obvious one. Make a decision tree that correctly classifies all the individuals in the training data. Since the goal of this solution is to generalize classification, it makes sense to use the smallest tree that can still correctly classify all of the training data. The naive way of finding the shortest tree would be to generate all possible trees and select the shortest one[1]. The problem that one encounters is the very large, but finite, number of decision trees that are capable of correctly classifying all data. The computation required to do this is unfeasible large. ID3 provides a good, but not optimal, solution to this problem.

ID3 is an iterative process that generates a short tree by splitting the tree one attribute at a time. The first attribute to be used is the one with the most information gain. Information gain is a metric that tells you how well that attribute splits the data into the correct classes. The equation for finding the gain in an $n - ary$ classification system is as follows.

One bias that ID3 introduces is the fact that attributes that have a large amount of possibilities will naturally have a higher gain and, therefore, will be chosen first. To combat this we will use gain ratio instead. Gain ratio takes into account the number of possibilities for each algorithm.

Another problem is that the tree still might not be general enough and could be trimmed down further. To help with this we will implement reduced error pruning. The general idea is to fully create a tree and then prune branches while checking if the missing branch reduces the accuracy of its classifications of a validation set.

3.4.2 Design Implications

This algorithm will affect the structure of our code greatly. A special tree structure will have to be created that can contain subsets of the training set and also leaf values for when a classification is found. In addition the training set is going to have to be divided to get a validation set to perform reduced error pruning. The main body of the learning part of the algorithm will be in a while loop with the conditions above. The search part of the algorithm will just consist of transversing the tree according to the attributes belonging to the query.

4 Software Architecture

5 Experiment Design

5.1 Algorithm Accuracy

5.1.1 Precision

5.1.2 Recall

5.1.3 Confusion

5.2 Algorithm Time-Complexity

5.3 Algorithm Convergence

References

[1] J.R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.