

Homework 2: Decision Trees

Jef Harkay

October 26, 2011

1 Dataset

1.1 Information

Large Dataset Instances:	200
Small Dataset Instances:	50
Number of Attributes:	10
Attributes:	1, 0
Classifications:	True, False

1.2 Boolean Function

The boolean function that the user supplies must consist of the variables A_1, A_2, \dots, A_n . The boolean function that was used contained 10 attributes because we know for 200 instances, we won't run out of permutations of 1's and 0's. If we had, say, 5 attributes, then we could only have 32 (2^5) unique dataset values.

$$(A1 \text{ and } A2 \text{ and } A3 \text{ and } !(A4 \text{ or } !A5)) \text{ or } (!A6 \text{ and } A9 \text{ and } A10 \text{ or } !A7 \text{ and } A8)$$

Figure 1: Boolean function used for datasets 1, 3, and 4.

1.3 Attribute Noise

The way I distributed the noise for each attribute was by editing 1 line for every 5 lines. This means that I edited 40% of the dataset lines. At each line, I would change only 1 of the attributes (either from 0 to 1, or 1 to 0). The position of the attribute changed was chosen at random.

1.4 Irrelevant Attributes

For the second dataset, I excluded A_3, A_6 , and A_9 . These variables were chosen at random.

$$(A1 \text{ and } A2 \text{ and } !(A4 \text{ or } !A5)) \text{ or } (A10 \text{ or } !A7 \text{ and } A8)$$

Figure 2: Irrelevant attribute boolean function for dataset 2.

1.5 Class Noise

For the classification noise, I changed 1 line for every 10 lines, so this means I edited 20% of the dataset lines. I would either change the classification from True to False or False to True. In total, I think I changed 12 falses to trues.

1.6 Smaller Datasets

The smaller datasets were broken down by finding the first 50 entries and deleting the rest. The data was already random, so there was no need to randomize the data, then crop it down.

2 Results

Table 1: Unpruned Results			
Dataset	Right	Wrong	Accuracy
1 Large	197	3	98.5%
1 Small	47	3	94%
2 Large	198	2	99%
2 Small	49	1	98%
3 Large	189	11	94.5%
3 Small	41	9	82%
4 Large	164	36	82%
4 Small	39	11	78%

Table 2: Pruned Results			
Dataset	Right	Wrong	Accuracy
1 Large	198	2	99%
1 Small	47	3	94%
2 Large	197	3	98.5%
2 Small	49	1	98%
3 Large	191	9	95.5%
3 Small	42	8	84%
4 Large	177	23	88.5%
4 Small	38	12	76%

3 Weka Screenshots

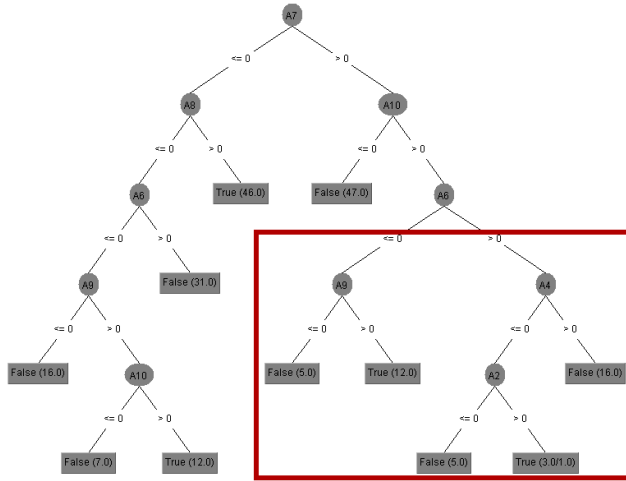


Table 3: Dataset 1 Large, unpruned tree

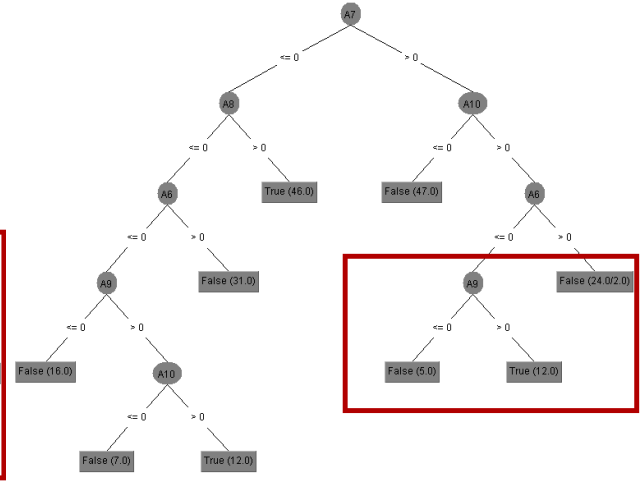


Table 4: Dataset 1 Large, pruned tree

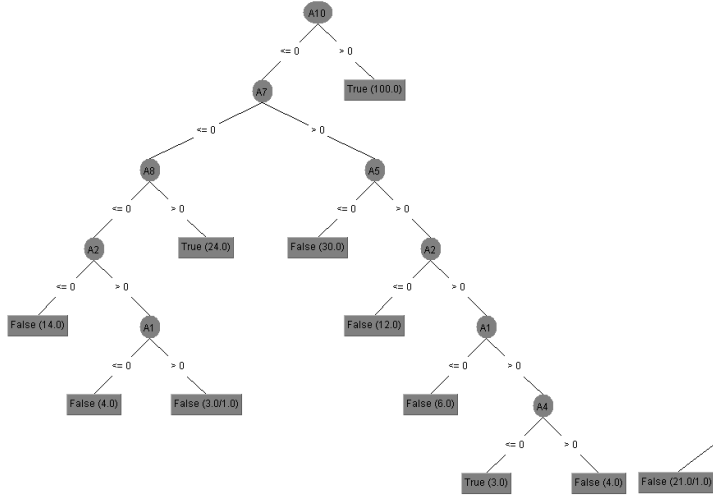


Table 5: Dataset 2 Large, unpruned tree

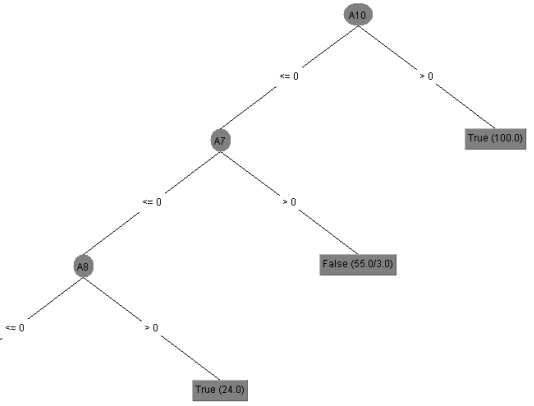


Table 6: Dataset 2 Large, pruned tree

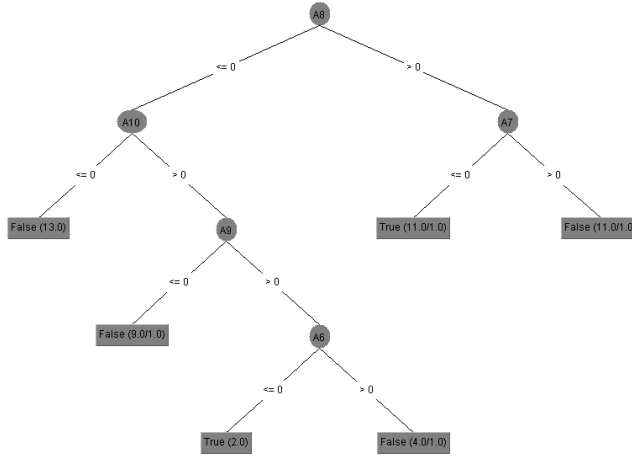


Table 7: Dataset 3 Small, unpruned tree

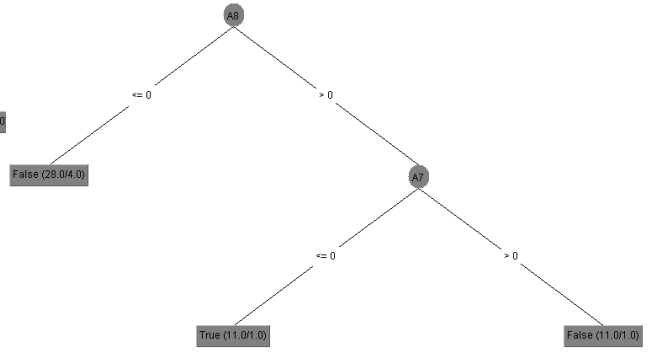


Table 8: Dataset 3 Small, pruned tree

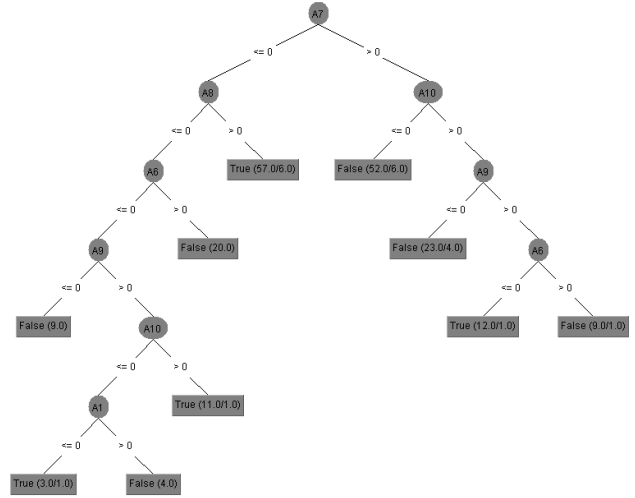
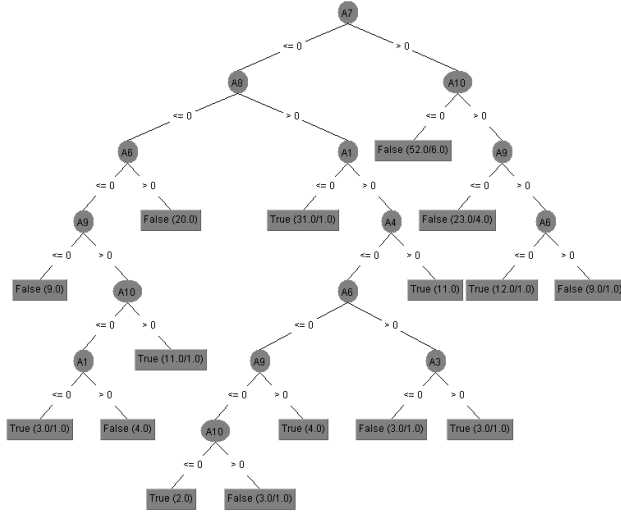


Table 9: Dataset 4 Large, unpruned tree

Table 10: Dataset 4 Large, pruned tree

4 Discussion

4.1 Normal Dataset

The unpruned tree (Table 3) came out looking pretty normal. The tree seems to favor the right side of the boolean function (A_6 - A_{10} side), but it makes sense that A_7 is at the top of the tree because of the OR between it and A_8 in the boolean function.

The nodes underneath A_6 have more weight for being false, so pruning the tree picks false and gets rid of nodes A_4 and A_2 . After the pruning is done, the left side of the boolean function has no say in the tree whatsoever. This either means that the randomness of the training values favored the right side of the boolean function, or the left side of the boolean function is just hard to evaluate to true.

4.2 Irrelevant Attribute Dataset

The first observation for the second dataset is that the irrelevant attributes stay irrelevant. A_3 , A_6 , and A_9 were not factored into the tree.

However, by comparing the unpruned (Table 5) and pruned (Table 6) trees, we see most of the branches can be trimmed off. This means the tree’s data created overfitting. This could be because we had to modify the boolean statement which gave more weight to certain attributes, or it could be a result of the randomly generated data.

4.3 Noise Attribute Dataset

Dataset 3 was interesting because the pruned tree (Table 8) shows that the unpruned tree (Table 7) contained overfitting. This proves that adding in random noise will lead to overfitting.

It's also interesting that the cross fold validation accuracies went down. Once the accuracies

start going down, you realize that the model is not a good generalized model, which means it will contain overfitting.

4.4 Noise Classification Dataset

The final dataset is just a mess. Not only is the unpruned tree (Table 9) overly complicated, but its cross validation accuracies are quite low (compared to the others). By getting rid of A_8 's right branches, the pruned tree (Table 10) only proves that the unpruned tree contained overfitting.

4.5 Conclusion

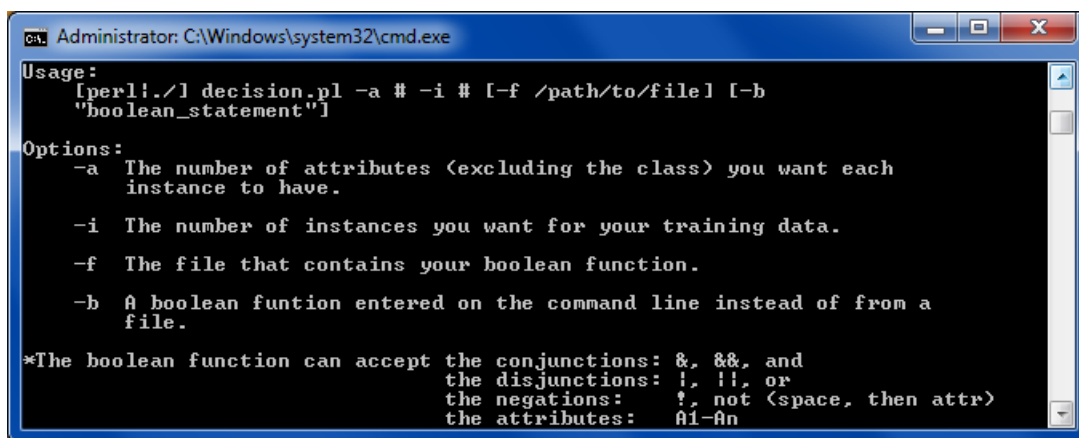
It has become apparent that by adding a little noise, whether if it's to the classification or attributes, the decision tree can become overly complex. This overfitting makes sense because you're training on falsified data. If any new, correct data comes in, the tree will have to create more branches because the noise data is going against what the correct data is saying.

Because the irrelevant data tree didn't include the irrelevant data, it's hard to conclude whether the overfitting was caused directly by the irrelevant data, or if it was some other problem.

5 Using Data Generator

To run decision.pl, you can check its usage statement by just executing the program. A couple of things to keep in mind:

- Put your boolean function in a file or enter it on the command line. Not both.
- The boolean function file must only contain the boolean function... nothing else.
- The boolean function must contain a space between variables, conjunctions, and disjunctions.
- The number of attributes you enter does not include the classification attribute.
- The boolean function must have the variables A_1 - A_n , where n is your highest number.



```
Administrator: C:\Windows\system32\cmd.exe
Usage:
[perl!./] decision.pl -a # -i # [-f /path/to/file] [-b
"boolean_statement"]

Options:
-a The number of attributes (excluding the class) you want each
   instance to have.
-i The number of instances you want for your training data.
-f The file that contains your boolean function.
-b A boolean funtion entered on the command line instead of from a
   file.

*The boolean function can accept the conjunctions: &, &&, and
the disjunctions: |, ||, or
the negations:   !, not (space, then attr)
the attributes:  A1-An
```

Figure 3: Usage statement for decision.pl

6 Code

```
use warnings;
use strict;
use Pod::Usage;
use Getopt::Long;
use File::Basename;

my $attr;
my $inst;
my $file;
my $bool;
my $relation = "boolean";
GetOptions( "a=i" => \$attr,
            "i=i" => \$inst,
            "f=s" => \$file,
            "b=s" => \$bool);
pod2usage(1) unless $attr and $inst and ($file or $bool);

if ($file) {
    my $path;
    my $suf;
    ($relation, $path, $suf) = fileparse($file, '\.[^\.]*');
    open my $fh, '<', $file or die "Can't open that file!  $!\n";
    $bool = <$fh>;
    chomp($bool);
}

# Relation for the ARFF file is the name of your boolean function file.
print '@RELATION ', "$relation\n\n";
for (my $i = 1; $i <= $attr+1; $i++) {
    if ($i == $attr) {
        print '@ATTRIBUTE class ', "\t{True,False}\n";
    }
    else {
        print '@ATTRIBUTE A', $i, "\tNUMERIC\n";
    }
}

my $range = 2;
print "\n", '@DATA', "\n";
for (my $i = 0; $i < $inst; $i++) {
    my $cur_bool = $bool;
    for (my $j = 1; $j <= $attr; $j++) {
        my $val = int(rand($range));
        print "$val,";
    }
}
```



```

    # Replacing the attribute variable with its randomly generated value.
    $cur_bool =~ s/\bA$j\b/$val/g;
}
# Evaluating the newly replaced boolean function.
if (eval($cur_bool)) {
    print "True\n";
}
else {
    print "False\n";
}
}

__END__

=head1 SYNOPSIS

[perl|./] decision.pl -a # -i # [-f /path/to/file] [-b "boolean_statement"]

=head1 OPTIONS

=item B<-a>

The number of attributes (excluding the class) you want each instance to have.

=item B<-i>

The number of instances you want for your training data.

=item B<-f>

The file that contains your boolean function.

=item B<-b>

A boolean funtion entered on the command line instead of from a file.

=begin text

*The boolean function can accept the conjunctions: &, &&, and
                                the disjunctions: |, ||, or
                                the negations:      !, not (space, then attr)
                                the attributes:     A1-An

=end text

```

=head1 DESCRIPTION

This program creates a random dataset with the user's supplied number of instances and attributes. Each attribute and classification will be either a 0 or 1.

=cut