Artificial Intelligence Project 2
Decision Tree

Class 1, Team 6

December 21, 2014

1 Team Members

Name	Student ID	Job
Fan Ziyao	12330081	Team leader, implementation
Chen Yingcong	12330049	Implementation
Chen Xiongtao	12330040	Modeling, implementation
Huang Long	12330132	Implementation
Zhang Qiuyi	12330402	Implementation, documentation

2 Problem Description

The dataset contains the records of the United States congressional votes in 1984. There are 435 records, each containing the party (goal attribute) of the congressman/congresswoman and his/her votes on 16 bills. There are only two types of values for the goal attribute: 'democrat' and 'republican', and three types for other attributes: 'n' (can be considered as nay), 'y' (can be considered as yea), and '?' (can be considered as missing).

Given this dataset, the goal is to build a decision tree to classify a record (that is, to find out the party of this congressman/congresswoman) according to its known attributes (that is, his/her votes on these 16 bills). Both the training set and the testing set will be sampled from this dataset.

3 Algorithm and Implementation

For this project we implemented the C4.5 algorithm to build the decision tree. Since the dataset only have two effective values ('y' and 'n') for all attributes, there is no need to use the information gain ratio as the measure of attributes. The information gain alone is more suitable for this dataset.

To handle the missing data, when building the tree, we only consider the records with defined value for the attribute being measured. When classifying the record, we use the probability of each possible value to weigh the calculation of each branch.

The data set is split by sampling the training set with a given probability p. The rest of the data will be put in to the testing set.

3.1 Decision tree building

The algorithm for building the decision tree is described in Algorithm 1.

Since the decision tree built with this data set won't be very deep (after all, there are only 16 attributes, and the data set is fairly skewed), a recursive implementation is sufficient. Records with missing values of the chosen attribute will be ignored when calculating the information gain.

Algorithm 1 Decision tree bulding

```
function BUILD($data$, $attributes$)
    if $data$ is empty then
        return an empty node
    end if
    $gain_{best} = 0.0$, $attr_{best}$ = None
    $value_{best}$ = None, $sets_{best}$ = None
    for each attribute $a$ in $attributes$ do
        Divide $data$ based on values of $a$ into $set1$, $set2$
        Calculate the information gain of spliting $data$ into $set1$ and $set2$, ignoring missing values
        if The new information gain $> gain_{best}$ then
            Update $gain_{best}$, $attr_{best}$, $sets_{best}$, $value_{best}$
        end if
    end for
    if $attributes$ is not empty and $gain_{best} \neq 0.0$ then
        remove $attr_{best}$ from $attributes$
        $left\_branch$ = BUILD($set1$ in$sets_{best}$, $attributes$)
        $right\_branch$ = BUILD($set2$ in$sets_{best}$, $attributes$)
        Create a node $node$
        $node.attr = attr_{best}$, $node.value = value_{best}$
        $node.left = left\_branch$, $node.right = right\_branch$, $node.count$ = size of $data$
        return $node$
    else
        Create a node $node$
        $node.leaves$ = count of each parties in $data$
        $node.count$ = size of $data$
        return $node$
    end if
end function
```

Remark 1.

The *entropy* of a attribute V in the data set D is defined as:

$$H(V, D) = -\sum_k P(v_k) \log_2 \frac{1}{P(v_k)}$$

where v_k are values of V in D, $P(v_k)$ is the probability of v_k in D.

Remark 2.

When a data set D with goal attribute V is split into subsets D_k using attribute A, each with n_k records, the *information gain* is defined as:

$$Gain(A) = H(V, D) - Remainder(A)$$

where

$$Remainder(A) = \sum_k P(n_k) H(V, D_k)$$

$P(n_k)$ is the proportion of E_k in D.

3.2 Decision tree pruning

The algorithm for pruning the decision tree is described in Algorithm 2. For the same reason as the tree building, a recursive implementation is enough. Nonetheless, since the data set is quite skewed, there is seldom a need for pruning the tree.

Algorithm 2 Decision tree pruning

```
function PRUNE($tree$, $min\_gain$)
    if $tree.left$ is not a leave then
        PRUNE($tree.left$, $min\_gain$)
    end if
    if $tree.right$ is not a leave then
        PRUNE($tree.right$, $min\_gain$)
    end if
    if Both $tree.left$ and $tree.right$ are leaves then
        Temprorily combine $tree.left$ and $tree.right$ into $new\_node$
        $delta = \text{ENTROPY}(new\_node) - (\text{ENTROPY}(tree.left) + \text{ENTROPY}(tree.right))/2$
        if $delta < min\_gain$ then
            Replace $tree$ with $new\_node$
        end if
    end if
end function
```

3.3 Classification

The algorithm for classifying an observation is described in Algorithm 3. It follows all branches at any node for which a value is missing and multiply the weights along each path.

Algorithm 3 Classification

```
function CLASSIFY($tree$, $observation$)
    if $tree$ is a leave then
        return $tree.leaves$
    end if
    $value = observation[tree.attr]$
    if $value$ is missing then
        $result\_left = \text{CLASSIFY}(tree.left, observation)$
        $result\_right = \text{CLASSIFY}(tree.right, observation)$
        $weight\_left = tree.left.count/tree.count$
        $weight\_right = tree.left.count/tree.count$
        for each pair of $goal, count$ in $result\_left$ do
            $result[goal] += count * weight\_left$
        end for
        for each pair of $goal, count$ in $result\_right$ do
            $result[goal] += count * weight\_right$
        end for
        return $result$
    else
        if $value = tree.left.value$ then
            return CLASSIFY($tree.left$, $observation$)
        else
            return CLASSIFY($tree.right$, $observation$)
        end if
    end if
end function
```

4 Experiment Result

To evaluate the implementation, we sample the data with probability p and use the sample as training set. The rest of the data set will be the testing set. One of the decision tree built with $p = 0.2$ is visualized in Figure 1. In addition, we take training sets sampled with various p to generate the learning curve for evaluation, as shown in Figure 2. With training sets of different sizes, the precisions of cross validation fluctuate between 0.9 and 1.0, with mean $\mu = 0.9462$ and standard variation $\sigma = 0.0175$.

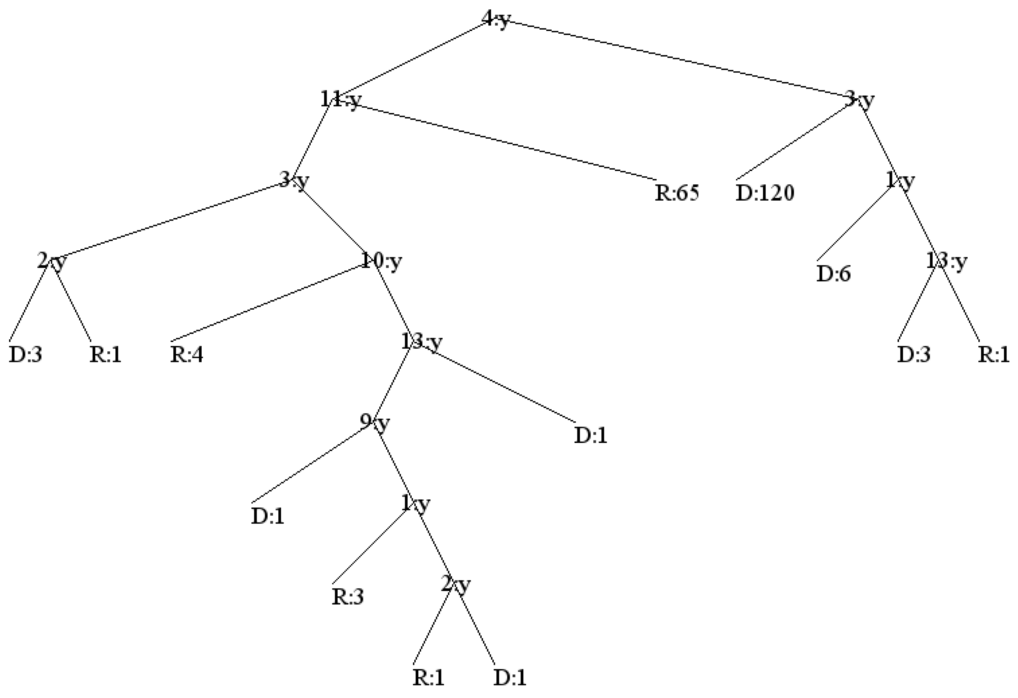

Figure 1: One of the decision trees built with training set sampled with $p = 0.5$

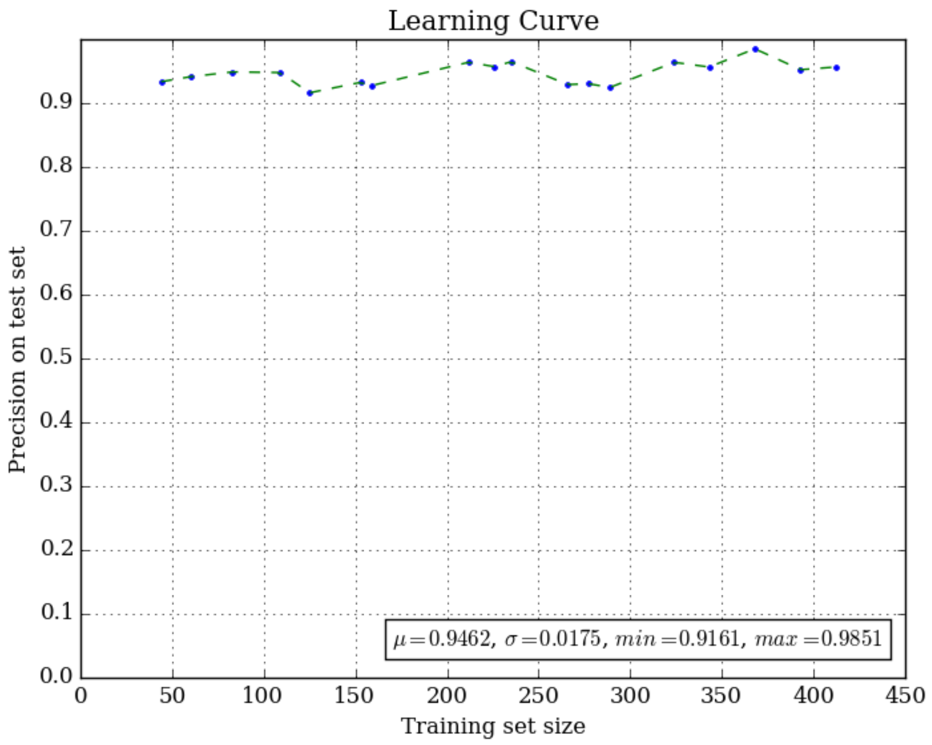

Figure 2: Learning Curve