pyC45 Documentation

Introduction

pyC45 is a simple and light C4.5 decision tree package for python which contains only one file "pyC45.py". It provides the user a simple and efficient interface to train a C4.5 decision tree and use it to do predictions or classifications. The trained decision tree is saved as an XML file so that it can be read and easily understood. C4.5 is one of the most popular algorithms in Data Mining and Machine Learning as the so called "Big Data Era" is coming. It is developed by 张驰昱 in Tsinghua University, in winter, 2013.
The source code is available on GitHub now.
https://github.com/zhangchiyu10/pyC45/

Principle

There are so many papers and documents which describe the algorithm in detail and so many advanced improvements of the efficiency of the algorithm has been achieved by Quilan and many other scientists. Thus, I don't want to waste time to repeat the detail but only to list several key steps and important features of the C4.5 algorithm.

1. Entropy
   We define the entropy of the set as follows, which indicates the diversity of the set element values. The smaller the entropy is, the purer the set is. If the set has only one value, its entropy should be 0.
   For distributed set:

$$\mathrm{Entropy(X)} = -\sum_{i\in X, i\neq "?"} \mathrm{P(x=i)log(P(x=i))}$$

   For continuous set:

$$\mathrm{Entropy\_c(X,i)} = -P(x<i)\mathrm{log}(P(x<i)) - P(x\geq i)\mathrm{log}(P(x\geq i))$$

2. Gain
   The gain of the attribute indicates the contribution to the classification by the attribute.
   For distributed set:

$$\mathrm{Gain(attribute)} = Entropy(category) - \sum_{k} P(attribute = k)Entropy(category_{attribute=k})$$

   For continuous set:

$$\text{Gain\_c(attribute)} = Entropy(category) - \min_{i\in\text{attribute},i\neq"?"} \{E \mid E = -P(\text{attribute} < i)Entropy(category_{attribute<k}) - P(\text{attribute} \geq i)Entropy(category_{attribute\geq k})\}$$

3. Gain Ratio

   If the selection of the attribute is only based the gain of attribute, it may often prefer the attributes which has more values, which is unreasonable. Thus we introduce Gain ratio.

   For distributed set:

$$\text{Ratio} = \frac{\text{Gain(attribute)}}{Entropy(\text{attribute})}$$

   For continuous set:

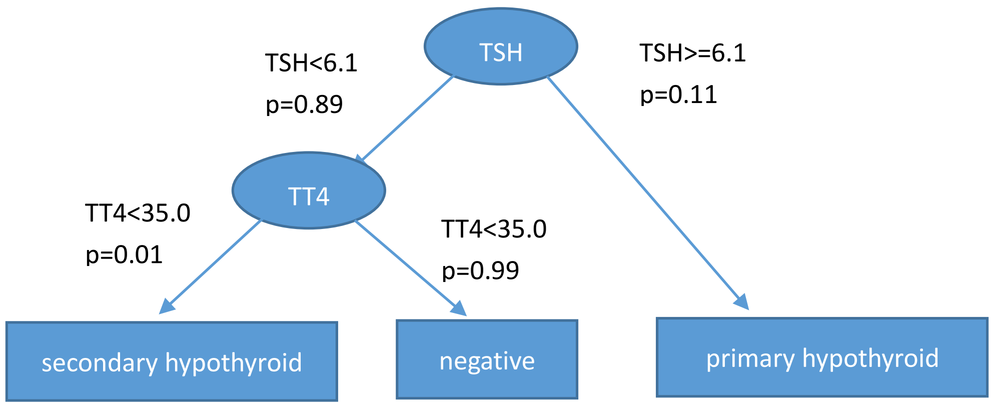
$$\text{Ratio} = \frac{\text{Gain(attribute)}}{Entropy\_c(\text{attribute, i})}$$

$$i = \min_{i\in\text{attribute},i\neq"?"} \{i \mid E = -P(\text{attribute} < i)Entropy(category_{attribute<k}) - P(\text{attribute} \geq i)Entropy(category_{attribute\geq k})\}$$

4. Missing value

   In the training process, when we calculate the entropy and the information gain ratio of an attribute, we only use the observed values.

   In the testing process, things become a little complicated. We actually list every possible result of the prediction and give every result a probability based on the training set. For instance:



As the decision tree showed avove, If both TSH and TT4 miss values, than the possible prediction are:

Secondary hypothyroid: 0.89*0.01≈0.01

Negative: 0.89*0.99≈0.88

Primary hypothyroid: 0.11
Thus, we select Negative as the prediction.

Usage of the functions

Function:
pyC45.train(training_obs,training_cat,xmlfile)

Description:
Train the decision tree and save it as a readable XML file.

Input:

Input name	type	description
training_obs	two-dimensional string array	Every element is an observation of the attributes, except that training_obs[0] is the column names.
training_cat	one-dimensional string array	Every element is an observation of the categories, except that training_cat[0] is the column name.
xmldir	string	The directory of the XML file to be saved

Output:
True/False

Function:
pyC45.predict (xmlfile,testing_obs)

Description:
Parse the decision tree XML file and predict the result of the testing set.

Input:

Input name	type	description
xmldir	string	The directory of the decision tree XML file
testing_obs	two-dimensional string array	Every element is an observation of the attributes, except that training_obs[0] is the

		column names.

Output:
True/False

● Demo

(See demo.py)

The error rate on “training_set.csv” is `0.87 %`
The error rate on “testing_set.csv” is `0.15 %`

Wow, the error rate is unbelievably low.
In fact, it’s reasonable. We use little cutting-leaf technique and the probability-based strategy of processing the missing value on testing set is convincible.

● XML file

The decision tree is like follows:

```
<DecisionTree>
  <TSH flag="l" p="0.89" value="6.1">
    <TT4 flag="l" p="0.001" value="35.0">secondary hypothyroid</TT4>
    <TT4 flag="r" p="0.999" value="35.0">negative</TT4>
  </TSH>
  <TSH flag="r" p="0.11" value="6.1">
    <FTI flag="l" p="0.289" value="65.0">
      <T4U flag="l" p="0.981" value="1.5">
        <thyroid_surgery flag="m" p="0.038" value="t">
          <age flag="l" p="0.5" value="50.0">primary hypothyroid</age>
          <age flag="r" p="0.5" value="50.0">negative</age>
        </thyroid_surgery>
        <thyroid_surgery flag="m" p="0.962" value="f">primary
hypothyroid</thyroid_surgery>
      </T4U>
      <T4U flag="r" p="0.019" value="1.5">negative</T4U>
    </FTI>
    <FTI flag="r" p="0.711" value="65.0">
      <on_thyroxine flag="m" p="0.195" value="t">negative</on_thyroxine>
      <on_thyroxine flag="m" p="0.805" value="f">
        <TT4 flag="l" p="0.953" value="160.0">
          <thyroid_surgery flag="m" p="0.029"
value="t">negative</thyroid_surgery>
          <thyroid_surgery flag="m" p="0.971" value="f">compensated
```

```
hypothyroid</thyroid_surgery>
        </TT4>
        <TT4 flag="r" p="0.047" value="160.0">negative</TT4>
      </on_thyroxine>
    </FTI>
  </TSH>
    </DecisionTree>
```

Tag elements	**Description**
Tag name	The tag name indicates the attribute name.
flag	If the flag=”m”, the attribute has distributed values. If the flag=”l”, it means the attribute has continuous value and it is the left child node. If the flag=”r”, it means the attribute has continuous value and it is the right child node.
value	Attribute value
p	The prior probability of the node based on the training set.