MARKS___

Decision Tree Learning for Classification

Pattern recognition: Decision Tree Learning for Classification

Professor: Dr. Zengchang Qin ,Dr. Yang Li

University: Beihang University

Student: Zenghui Zhou

Student Number:15231122

Final Date: June 8, 2018

Introduction

Decision tree induction is one of the simplest and yet most successful learning algorithms. A decision tree (DT) consists of internal and external nodes and the interconnections between nodes are called branches of the tree. An internal node is a decision-making unit to decide which child nodes to visit next depending on different possible values of associated variables. In contrast, an external node also known as a leaf node, is the terminated node of a branch. It has no child nodes and is associated with a class label that describes the given data. A decision tree is a set of rules in a tree structure, each branch of which can be interpreted as a decision rule associated with nodes visited along this branch.

Principle and Theory

Decision trees classify instances by sorting them down the tree from root to leaf nodes.This tree-structured classifier partitions the input space of the data set recursively into mutually exclusive spaces. Following this structure, each training data is identified as belonging to a certain subspace, which is assigned a label, a value, or an action to characterize its data points. The decision tree mechanism has good transparency in that we can follow a tree structure easily in order to explain how a decision is made.Thus interpretability is enhanced when we clarify the conditional rules characterizing the tree.

Entropy of a random variable is the average amount of information generated by observing its value. Consider the random experiment of tossing a coin with probability of heads equal to 0.9, so that P(Head) = 0.9 and P(Tail) = 0.1. This provides more information than the case where P(Head) = 0.5 and P(Tail) = 0.5. Entropy is used to evaluate randomness in physics, where a large entropy value indicates that the process is very random. The decision tree is guided heuristically according to the information content of each attribute. Entropy is used to evaluate the information of each attribute; as a means of classification. Suppose we have m classes, for a particular attribute, we

denoted it by pi by the proportion of data which belongs to class C_i where $i = 1, 2, \ldots m$.

The entropy of this attribute is then:

$$Entropy = \sum_{i=1}^{m} -p_i \log_2 p_i$$

We can also say that entropy is a measurement of the impurity in a collection of training examples: larger the entropy, the more impure the data is. Based on entropy, Information Gain (IG) is used to measure the effectiveness of an attribute as a means of discriminating between classes.

$$IG(S, A) = Entropy(S) = \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

where all examples S is divided into several groups (i.e. S_v for $v \in Values(A)$) according to the value of A. It is simply the expected reduction of entropy caused by partitioning the examples according to this attribute.

Objective

The goals of the experiment are as follows:

(1) To understand why we use entropy-based measure for constructing a decision tree.

(2) To understand how Information Gain is used to select attributes in the process of building a decision tree.

(3) To understand the equivalence of a decision tree to a set of rules.

(4) To understand why we need to prune the tree sometimes and how can we prune? Based on what measure we prune a decision tree.

(5) To understand the concept of Soft Decision Trees and why they are important extensions to classical decision trees.

Contents and Procedure

Stage 1:

(1) According to the above principle and theory in section 3.2, implement the code to calculate the information entropy of each attribute.

(2) Select the most informative attribute from a given classification problem. (e.g., we will be given the Iris Dataset from the UCI Machine Learning Repository)

(3) Find an appropriate data structure to represent a decision tree. Building the tree from the root to leaves based on the principal discussed in section 3.2 by using Information Gain guided heuristics.

Stage 2:

(1) Now consider the case of with continuous attributes or mixed attributes (with both continuous and discrete attributes), how can we deal with the decision trees? Can you propose some approaches to do discretization?

(2) Is there a tradeoff between the size of the tree and the model accuracy? Is there existing an optimal tree in both compactness and performance?

(3) For one data element, the classical decision tree gives a hard boundary to decide which branch to follow, can you propose a "soft approach" to increase the robustness of the decision tree?

(4) Compare to the Naïve Bayes, what are the advantages and disadvantages of the decision tree learning?

Stage 3：

Explore the questions in the previous section and design experiments to answer these questions. Complete and submit an experiment report about all experiment

results with comparative analysis and a summary of experiences about this experiment study.

Experiment Results

Stage 1:

(1) I already write a function to calculate the information entropy of each attribute. The code are as follows. According to the test resule

```
def ComputeEntropy(dataset):
    """
    Compute the entropy.
    :param dataset: a 2D list,each row of which is like [some feature value, classtype]
    :return: the entropy computed
    """
    p1_num = 0
    p2_num = 0
    p3_num = 0
    for i in range(len(dataset)):
        if dataset[i][1] == 0:
            p1_num += 1
        if dataset[i][1] == 1:
            p2_num += 1
        if dataset[i][1] == 2:
            p3_num += 1
    p1 = p1_num / (p1_num + p2_num + p3_num)
    p2 = p2_num / (p1_num + p2_num + p3_num)
    p3 = p3_num / (p1_num + p2_num + p3_num)
    ent = 0
    if p1 != 0:
        ent = ent - p1*np.log2(p1)
    if p2 != 0:
        ent = ent - p2 * np.log2(p2)
    if p3 != 0:
        ent = ent - p3 * np.log2(p3)
    return ent
```

(2) I use the UCI Machine Learning Repository to test my code and it performance well. We chose 80% of the dataset as our train dataset and the others as our test dataset. The results are as follows:

```
D:\2018年春季课程\模式识别\experiment-3
Accuracy: 1.0
```

So there exist some overfit problems, so we set an alpha to try to amend the overfitting. The default value of alpha is 1. You can change it in the source code. If we change alpha to 0.4, the result is as follows

```
d:\2018年春季课程\模式识别\experiment-3
Accuracy: 0.6666666666666666
```

(3) In my code, I use the class type to represent my tree and according to different condition to give different nodes. The code is as follows.

```
class TreeNode(object):
    def __init__(self, classtype=None, leftnode=None, rightnode=None, feature=None, threshold=None):
        self.__classtype = classtype
        self.__leftnode = leftnode
        self.__rightnode = rightnode
        self.__feature = feature
        self.__threshold = threshold

    def set_classtype(self, classtype):
        self.__classtype = classtype

    def add_leftnode(self, leftnode):
        self.__leftnode = leftnode

    def add_rightnode(self, rightnode):
        self.__rightnode = rightnode

    def set_feature(self, feature):
        self.__feature = feature

    def set_threshold(self, threshold):
        self.__threshold = threshold

    def get_classtype(self):
        return self.__classtype

    def get_leftnode(self):
        return self.__leftnode
```

Stage 2:

(1) If the attributes are continuous or mixed, we can discreet it by ourselves. In my code, I pick the average of two adjacent sample value as the threshold to discrete the attributes.

```
thresholds.append((fea_value[i] + fea_value[i+1]) / 2)
```

(2) Our model is overfitting, and it is probable that every leaf only contains one data which means our tree is oversize though it has a good performance. So pre-pruning and post-pruning is necessary to be considered here. But owing to the lack of time, I don't carry out it in my model.

(3) When the distance of data and boundary is very short which means it is difficult for is to assert the data is exactly belonging to one class. So we should try to class the data in a higher dimension which means we need to have a class named 'unknow' which saves the data closed to the boundary.

(4) Decision tree is a better way to classify the data in a high dimension compared to Naïve Bayes. Also, decision tree is easier to be train. It needs a little data but has good performance. Decision tree has more advantages when the dataset is very large. The tree can be very small and have high accuracy.

Experiment Results

The Decision Tree is very useful to deal with the classification problem, it is more useful compared to the Naïve Bayes. In our real life, the decision is used everywhere and it can largely simplify our work.

In this experiment, I have classified the UCI Machining Learning Dataset. I become more familiar with the Decision Tree. In the process of experiment, I have met many questions. I can't understand why the continuous attributes can be used to classify the child nodes rather than abandon it. But I successfully work out these questions with the help of my classmates. I have made a great progress. The way I prevent overfitting is also under his help.

Appendix

Code Address:

https://github.com/Charlie2048/pattern-recognition/tree/Decision-Tree