北京航空航天大學
BEIHANG UNIVERSITY

Pattern Recognition and Machine Learning

Experiment Report

School of Automation Science and Electrical Engineering

Kun Xiao

14051166

July,2018

Experiment 3 Decision Tree Learning for Classification

3.1 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.

3.2 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, ... m.*

The entropy of this attribute is then:

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

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.

3.3 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 decsion 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 mesure we prune a decsion tree.

(5) To understand the concept of Soft Decsion Treees and why they are imporant extensions to classical decision trees.

3.4 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 decion tree. Building the tree from the root to leaves based on the principal discussed in section 3.2 by using Information Gain guided heuritics.

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 decsion tree gives a hard bounday 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 disvantages 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.

5. Results and Conclusion

5.1 数据集选择

鸢尾花卉数据集（英文：Iris flower data set）是一个经典的连续属性分类数据集，这里选择其为实验数据集。

5.2 准则选择与离散化技术

本实验按照报告要求选择的信息增益的方法。信息增益的代码实现分为两个函数，第一个函数求解信息熵，第二个函数求解信息增益。由于 IRIS 数据集的属性是连续的，因此需要对其进行离散化，这里使用二分法，处理后的信息增益变为：

$$Gain(D,a) = \max_{t \in T_a} Gain(D,a,t) = \max_{t \in T_a} Ent(D) - \sum_{\lambda \in \{-,+\}} \frac{|D_t^\lambda|}{|D|} Ent(|D_t^\lambda|)$$

函数程序如下：

```
    # 计算信息熵
    @staticmethod
    def calculateEntropy(dataset):
        flag2num = dict()
        total = len(dataset)
        entropy = 0
        for data in dataset:
            flag = data[-1]
            if flag in flag2num.keys():
                flag2num[flag] += 1
            else:
                flag2num[flag] = 1
        for flag in flag2num:
            p = flag2num[flag] / total
            entropy -= p * np.log2(p)
        return entropy

    # 计算信息增益
    @staticmethod
    def calculateEntropyGain(dataset, attribute, threshold):
        # 根据特征选择数据
        dataset.sort(key = lambda x: x[attribute])
        entropy = DecisionTree.calculateEntropy(dataset)
        # 找到划分点
        thres_pos = 0
        while (dataset[thres_pos][attribute] < threshold):
            thres_pos += 1
        entropy -= DecisionTree.calculateEntropy(dataset[:thres_pos]) * thres_pos / len(dataset)
        entropy -= DecisionTree.calculateEntropy(dataset[thres_pos:]) * (len(dataset) - thres_pos) / len(dataset)
        return entropy
```

5.3 决策树生成与可视化

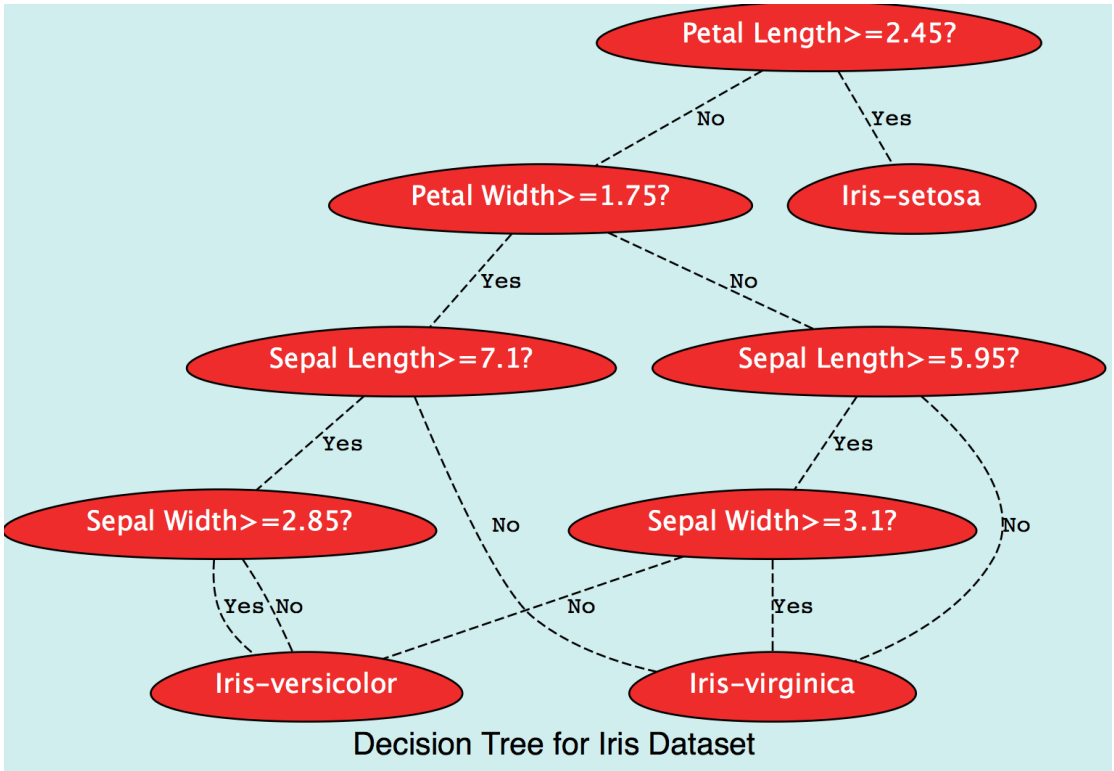
计算过按照每个属性分类所对应的信息增益后，选择信息增益最大的对应的属性作为此节点的分类属性，注意连续属性不同于离散属性，作为划分属性的属性之后仍然可以作为划分属性。

该决策树使用面向对象编程，枝节点具有四个属性。整个决策树以递归的方式生成。

```
# 把枝节点以类的描述，叶节点用0,1,2的整数表示，对应不同的IRIS种类
class BranchNode():
    def __init__(self, attribute, threshold, left, right):
        self.attribute = attribute
        self.threshold = threshold
        self.left = left
        self.right = right
```

```
# 递归生成决策树
@staticmethod
def createTree(dataset, attributes):
    flagList = [data[-1] for data in dataset]
    if flagList.count(flagList[0]) == len(flagList):
        return flagList[0]
    if len(attributes) == 0:
        return DecisionTree.majorityFlag(dataset)
    (attributeSelected, threshold) = DecisionTree.selectAttribute(dataset, attributes)
    dataset_l = [data for data in dataset if data[attributeSelected] < threshold]
    dataset_h = [data for data in dataset if data[attributeSelected] >= threshold]
    attributes.remove(attributeSelected)
    if len(dataset_l) > 0:
        left = DecisionTree.createTree(dataset_l, attributes)
    else:
        left = DecisionTree.majorityFlag(dataset)
    if len(dataset_h) > 0:
        right = DecisionTree.createTree(dataset_h, attributes)
    else:
        right = DecisionTree.majorityFlag(dataset)
    node = DecisionTree.BranchNode(attributeSelected, threshold, left, right)
    attributes.add(attributeSelected)
    return node
```

程序使用 graphviz 库实现了决策树的可视化，如下图所示：



5.4 过拟合与剪枝

训练过程可能把节点分的过细（决策树复杂度高），导致了过拟合；而若分的粗糙（决策树复杂度低），将导致训练集和测试集的分类正确率均不佳，这就是决策树复杂度与训练效果的 tradeoff 问题。

为了防止过拟合，可以使用剪枝处理，从而提高模型的泛化能力。上图中 Sepal Width ≥2.85 均归于一类，就是利用了剪枝处理。

5.5 决策树与朴素贝叶斯的比较

优点：

1．可解释性强，易于调试

2. 需要的训练数据两较小

3. 在绝大多数场合都能有比较不错的分类效果

4. 对于缺失值不敏感

缺点：

1. 离散化技术导致对于连续属性分类的准确度一般不如朴素贝叶斯

2. 容易过拟合

3. 回归问题的难度较大

6 源代码

https://github.com/robin-shaun/BUAA-Pattern-Recognition-and-Machine-Learning/Exp3