Homework: Decision Tree Classification

XiangKai Wu 18377024

Abstract

Decision tree is an intuitive and effective learning algorithm. Starting from the basic principles and properties, this paper uses Iris data to code to successfully implement the decision tree algorithm, and presents the decision tree by using the data structure of Python dictionary. In addition, the method of sorting and bi-partition is used to transform continuous values into discrete values.

This paper also discusses the generalization performance of decision tree, and considers that there is over-fitting problem in the training of decision tree. In the experiment, the smallest acceptable difference of entropy is set, which successfully improves the accuracy of the decision tree on the testing dataset and reduces the number of layers of the decision tree. This paper also discusses the problem of soft decision tree. In the experiment, using linear fuzzy function in decision-making does not enhance the accuracy of the algorithm. How to reset the parameters of the decision tree and select the appropriate fuzzy function is the key to improve the performance of soft decision tree.

1 Introduction

Decision tree learning is one of the predictive modelling approaches used in statistics, data mining and machine learning. It uses a decision tree to go from observations about an item to conclusions about the item's target value. Tree models where the target variable can take a discrete set of values are called classification trees. In these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. Decision trees are among the most popular machine learning algorithms given their intelligibility and simplicity.

2 Stage 1: Complete Classical Decision Tree

2.1 Calculate the Information Entropy of Each Attribute

The equation of information entropy is

$$\mathrm{Ent}(D) = -\sum_k p_k \log_2 p_k$$

And code is completed in `DecisionTree.py`, function `get_shannon_ent`.

2.2 Decision Tree

Given a dataset, the decision tree can be built to do classification. And code is completed in `DecisionTree.py`. The idea of recursion is used.

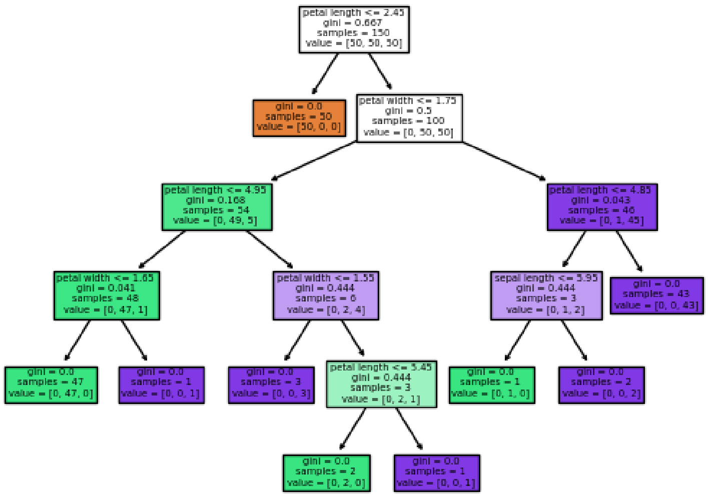
2.3 Data Structure to Represent a Tree

The data structure used to represent a tree can be dictionary. When we get the information entropy gain and corresponding feature and dividing point, we build another dictionary. And this can save a decision tree. Code is completed in `DecisionTree.py` function `my_ tree`.

And output dictionary is

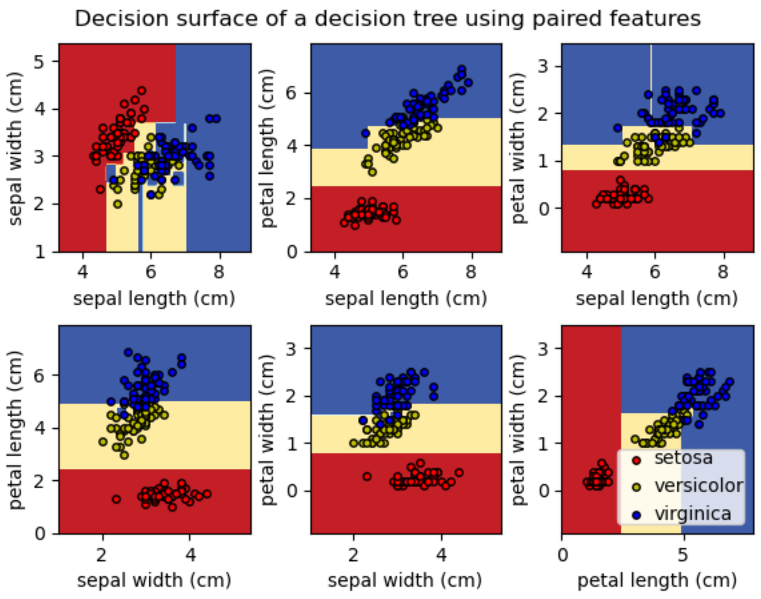
```
{'petal length': {'<2.45': 'Setosa', '>2.45': {'petal width': {'<1.75': {'petal length': {'<4.95': {'petal width': {'<1.65': 'Versicolour', '>1.65': 'Virginica'}},
 '>4.95': {'petal width': {'<1.55': 'Virginica', '>1.55': {'sepal length': {'<6.95': 'Versicolour', '>6.95': 'Virginica'}}}}}}, '>1.75': {'petal length': {'<4.85':
 {'sepal length': {'<5.95': 'Versicolour', '>5.95': 'Virginica'}}, '>4.85': 'Virginica'}}}}}}
```

We use the toolbox of package scikit-learn to verify the answer, and the answer is absolutely the same.



Code is completed in `DecisionTreeTest.py` (The code is taken from the official website scikit-learn). That indicates the correctness of my idea.

And the classification answer is



3 Stage 2: Explorative Problem

3.1 Discretization

When facing the case of with continuous attributes or mixed attributes (with both continuous and discrete attributes), the simplest strategy is bi-partition.

Given a data set D and a continuous attribute a, suppose that a has n different value. Sorting can be done and recorded as $a^1, a^2, \cdots, a^n$. D can be divided into D_- and D_+ based on dividing point t, where D_- is the set in which the attribute a value of the samples is smaller than t, and D_+ is the set in which the attribute a value of the samples is larger than t.

Hence, for continuous attribute a, we can consider a dividing point set which includes $n-1$ elements:

$$T_a = \left\{ \frac{a^i + a^{i+1}}{2} | 1 \leq i \leq n-1 \right\}$$

That means, we take the middle point of the interval $[a^i, a^{i+1})$ as the dividing point. And then, we can consider these values in the view of discrete values.

Actually, any value in the interval $[a^i, a^{i+1})$ is can be chosen as a dividing point, and it does not have any differences.

And the information gain can be expressed as

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

This strategy is used in Stage 1 to get classification of the dataset, due to all attributes of the Iris Dataset is continuous.

3.2 Optimal Tree in both Compactness and Performance

Considering a extreme case, a dataset D contains two kinds of elements A and B, the number of them are n (very large) and 1 respectively. One more classification can divide D into accurate two category, but we consider this case is over-fitting, because the difference of entropy is very small when we delete a small sample.

Thus, if the difference of entropy is smaller than 0.1, for example, we do not divide this node any more, and the class of this node only depends on number of each class.

To compare the model accuracy of two methods, we take 80% of the dataset randomly as training set and 20% of the dataset randomly as testing set.

If we consider the smallest acceptable difference of entropy 0.1, then the decision tree is like this:

```
{'petal length': {'<2.7': 'Setosa', '>2.7': {'petal width': {'<1.75': {'petal length': {'<4.95': 'Versicolour', '>4.95': {'petal width': {'<1.55': 'Virginica',
 '>1.55': 'Versicolour'}}}}, '>1.75': 'Virginica'}}}}
```

While the classical decision tree is as followed, which is much more complex when using the same training dataset.

```
{'petal length': {'<2.7': 'Setosa', '>2.7': {'petal width': {'<1.75': {'petal length': {'<4.95': {'sepal length': {'<4.95': 'Virginica', '>4.95': 'Versicolour'}},
 '>4.95': {'petal width': {'<1.55': 'Virginica', '>1.55': {'sepal length': {'<6.95': 'Versicolour', '>6.95': 'Virginica'}}}}}}, '>1.75': {'petal length': {'<4.85':
 {'sepal length': {'<6.05': 'Versicolour', '>6.05': 'Virginica'}}, '>4.85': 'Virginica'}}}}}}
```

The accuracy of this decision tree on testing dataset is 100% while the accuracy of classical decision tree on testing dataset is 93.3%. That indicates that a simpler decision tree may take better performance in both compactness and accuracy.

If we consider the smallest acceptable difference of entropy 0.3, then the decision tree is like this:

```
{'petal length': {'<2.45': 'Setosa', '>2.45': 'Virginica'}}
```

While the classical decision tree is as followed, which is much more complex when using the same training dataset.

```
{'petal length': {'<2.45': 'Setosa', '>2.45': {'petal width': {'<1.75': {'petal length': {'<4.95': {'petal width': {'<1.65': 'Versicolour', '>1.65': 'Virginica'}},
 '>4.95': {'sepal length': {'<6.4': 'Virginica', '>6.4': 'Versicolour'}}}}, '>1.75': {'petal length': {'<4.85': {'sepal length': {'<5.95': 'Versicolour', '>5.95':
 'Virginica'}}, '>4.85': 'Virginica'}}}}}}
```

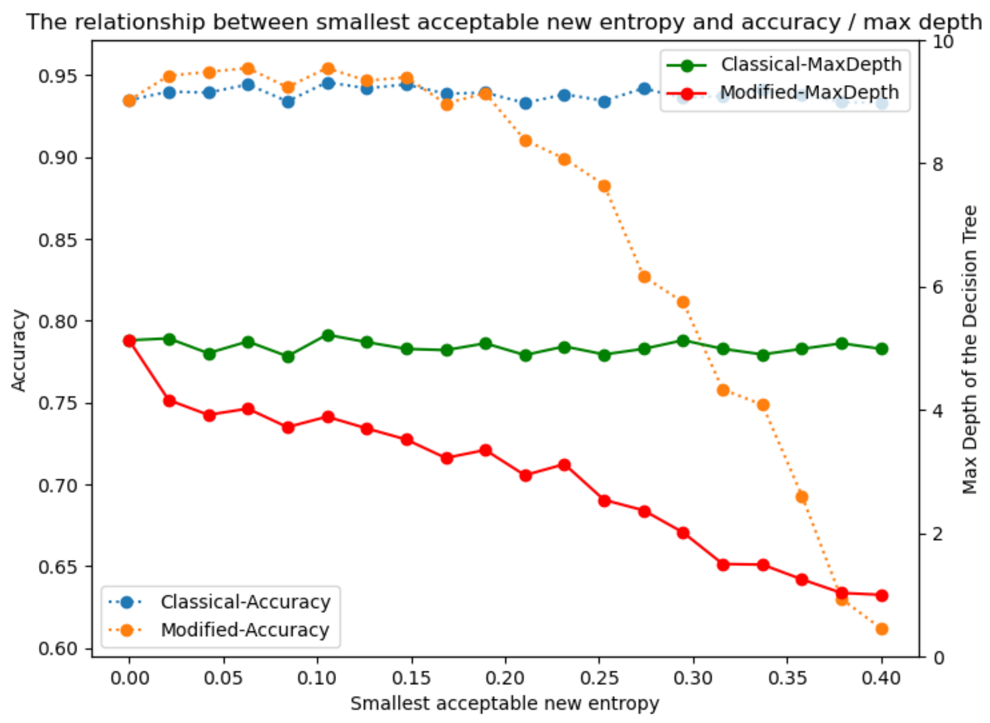
The accuracy of this decision tree on testing dataset is 70% while the accuracy of classical decision tree on testing dataset is 93.3%. If the decision tree is too simple, the accuracy will decrease for sure.

Code about how to build such a decision tree is completed in `ModifiedDecisionTree.py`.

When acceptable difference of entropy is small, the tree is quite complex. When acceptable difference of entropy becomes large, the accuracy of the decision tree decreases sharply. So a worth-thinking question is how to choose a appropriate acceptable difference of entropy in order to get both higher accuracy and less depth of the decision tree.

A experiment can be conducted. For a certain acceptable difference of entropy, we calculate the accuracy and the maximal depth of classical decision tree and modified decision tree, using the same training dataset and testing dataset. To ensure the validity of the experiment, each experiment will be conducted 100 times repeatedly.

And we can draw a figure to show the relationship between smallest acceptable difference of entropy and accuracy and maximal depth of the decision tree:



Code about drawing such a figure is completed in `plot_image.py` function `plot_new_entropy_accuracy()`.

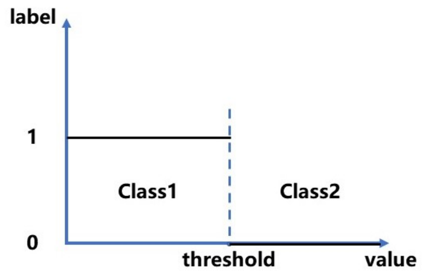
It can be observed that when smallest acceptable difference of entropy is between 0 and 0.15, the accuracy of the modified decision tree is a little higher than the accuracy of the classical decision tree.

And the maximal depth of modified decision tree is less than the classical decision tree. When smallest acceptable difference of entropy is larger than 0.15, the accuracy of the modified decision tree is much lower than the accuracy of the classical decision tree, although the maximal depth of modified decision tree is less than the classical decision tree.

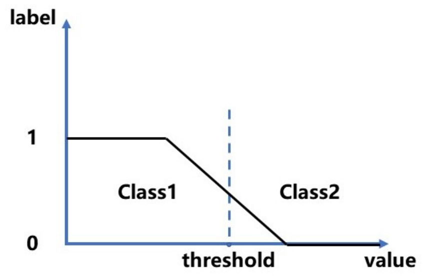
Therefore, it is not accurate to say that there is a tradeoff between the size of the tree and the model accuracy. Because we can choose appropriate smallest acceptable difference of entropy to get both higher accuracy and less depth of the decision tree. And this technic can help the decision tree avoid over fitting problem.

3.3 Soft Approach

The boundary of the classical decision tree is parallel lines. This makes the solution more interpretable, but also makes the decision tree much more complex. The schematic diagram of this method is as follows:



But we can consider the soft boundary to decide which branch to follow. We can consider the membership function in fuzzy mathematics[1]. If the value of a specific feature of a test data is around the threshold, then it may belong to either the first or the second class. The schematic diagram of this method is as follows:



The fuzzy membership value $\mu(x)$ is calculated as given below.

$$\mu(x) = \begin{cases} 1, & x \leq \text{threshold} - \omega \\ \dfrac{(\text{threshold} + \omega) - x}{2\omega}, & \text{threshold} - \omega < x < \text{threshold} + \omega \\ 0, & x \geq \text{threshold} + \omega \end{cases}$$

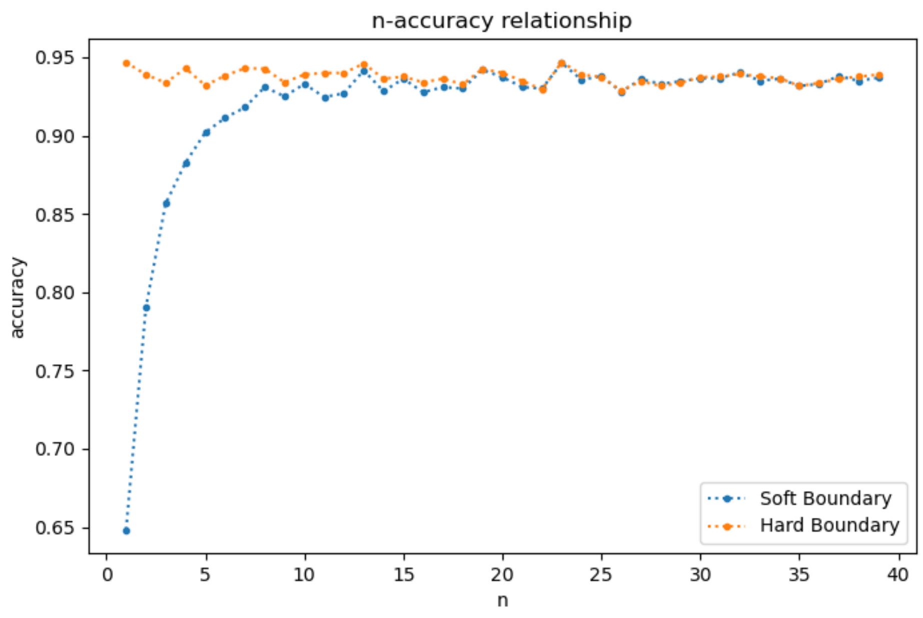
Hence, we can consider probability model to achieve this model. we can sample a number from a uniform distribution and if this number $p(x) < \mu(x)$, then this data will be labeled Class 1, otherwise it will be labeled Class 2.

Code is completed in `SoftDecisionTree.py` function `classify.py`.

And we turn the hard boundary into soft boundary using a method in fuzzy mathematics.

And a question is, how do we choose a appropriate ω. A experiment is conducted to find this answer. let $\omega = \frac{\text{threshold}}{2n}$. And we traverse n from 1 to 40. And we can get the relationship between n and accuracy.

Here is the result:



And it can be discovered that the accuracy of soft decision tree is lower than the classical decision tree. When n increases, the difference of accuracy decreases. Only the n is about 20, the accuracy of soft decision tree is a little higher than the classical decision tree. Therefore, the soft boundary does not help improve the specifications of the decision tree (One reason is that the accuracy of classical deciosion tree is already very high in Iris dataset). So we also need to design the parameters of the soft decision tree instead of using the parameters of the classical decision tree.

And how to determine threshold and ω of a soft decision tree is a question, because going through all the data increases time complexity sharply. It's not worth it in my opinion. And I still can not find a good way to solve it.

3.4 Comparison

Naïve Bayes supposes that the distribution of all random variables are independent. This is a "strong" hypothesis, and it may fail to predict in some occasions.

Decision Tree does not require that all random variables are independent, but it is easy for decision tree to be over-fitting. And that requires us to do prune to balance compactness and accuracy. To ensure the validity of the experiment, each n value experiment will be conducted 100 times repeatedly.

4 Conclusion

By using Python dictionary, this paper successfully implements the algorithm of decision tree, and uses bi-partition strategy to transform the continuous values in Iris data into discrete values.

It is found that in the process of classification, if the difference between the two information entropy is too small, the decision tree will be over fitted. If the nodes whose entropy difference is less than 0.15

are no longer divided, the accuracy of the decision tree on the testing dataset will be improved by about 2%, and the number of layers of the decision tree will be reduced by 1 to 2.

Soft decision tree uses fuzzy function in the decision-making process, but the experiment shows that the performance of soft decision tree is not obviously better if the parameters and fuzzy function of the decision tree are not designed carefully. But how to choose the appropriate parameters and fuzzy function needs further study.

5 Summary of Experiences

Through this experiment, I basically understand the basic principle of decision tree and its code implementation, and have a deeper understanding of information entropy.

At the same time, whether the data feature is continuous or discrete, I have a deep understanding of its solution. More importantly, through this experiment, I feel that the classical decision tree model generally has over fitting phenomenon. If we do not divide the nodes if there is only little change in information entropy (usually smaller than 0.15), the accuracy of the decision tree will be improved, the depth will be reduced, and the over fitting problem will be solved.

And Soft approach can be used to improve the specifications of the decision tree. But how to determine the parameter of the decision tree is still a problem for me.

最后想说的是，之前看作业题目是英文描述，所以下意识就用英文写报告了，写了一半才意识到中文报告也是可以的。但想着：写都写了，就算了吧，于是诞生了这样一篇报告。英语不才，诸多语言纰漏之处请老师谅解。

Github Code Address: https://github.com/Akaikai/Homework_DecisionTree

References

[1] Kumar G K , Viswanath P , Rao A A . Ensemble of randomized soft decision trees for robust classification[J]. Sādhanā, 2016, 41(3):273-282.