SHANGHAI JIAO TONG UNIVERSITY

Decision Trees for Classification

Group Project for ICE2601

Zhu Pengxiang, Zhou Shengyang

May 22, 2023

Fundamentals of Information Theory

Entropy, proposed by Shannon[1] , is a measure of the uncertainty of a random variable. Consider a discrete random variable $X \sim p(x)$ with alphabet $\mathcal{X}$, the entropy $H(X)$ is defined by

$$H(X) = -\sum_{p(x)} p(x) \log p(x) \tag{1}$$

Suppose two random variables $X, Y \sim p(x, y)$ with alphabets $\mathcal{X}, \mathcal{Y}$, we can define $H(X|Y)$ and $I(X;Y)$ as

$$H(Y|X) = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \log p(y|x) \tag{2}$$

$$I(X;Y) = H(X) - H(X|Y) \tag{3}$$

[1] Claude E Shannon. "A mathematical theory of communication". In: *The Bell system technical journal* 27.3 (1948), pp. 379–423.

General Formation of Classification Problems

Data classification problems consists of two parts, the learning step and classification step[2] . During the learning step, a training data set D of length N needs to be prepared. Here we suppose that there are n attributes for each sample $\mathbf{X}_i \in D$, $i = 1, 2, \dots, N$. For each potential $\mathbf{X}$ in alphabet $\mathcal{X}$, we assume that there are k possible labels, denoted by $C = \{c_1, \dots, c_k\}$. A predefined label $c(\mathbf{x}) \in C$ is associated with each sample $\mathbf{X}_i$. The training process is intended to formulate a function f such that

$$f := \mathcal{X} \mapsto C \tag{4}$$

[2]What Is Data Mining. "Data mining: Concepts and techniques". In: *Morgan Kaufinann* 10 (2006), pp. 559–569.

General Setting of a Decision Tree

Decision tree induction is the learning of decision trees of class-labeled training tuples. A typical decision tree is shown in Figure. It consists of a tree structure where each branch represents an outcome of the decision process and each node has a label that denotes the splitting criterion.

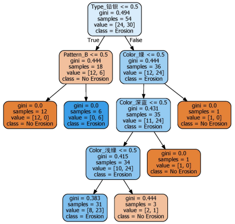


Figure: A typical example of decision tree

General Setting of a Decision Tree

ID3 algorithm[3] is an algorithm to construct such a tree from training samples.

- The inputs are a data partition D consisting of training tuples and class labels.
- $attribute\_list$ refers to the list of attributes (n of them in total) used for classification.
- $attribute\_selection\_method$ is a method to determine the splitting criterion.

Here we consider the case of ID3 algorithm.

[3] J. Ross Quinlan. "Induction of decision trees". In: *Machine learning* 1 (1986), pp. 81–106.

ID3 Algorithm

SHANGHAI JIAO TONG UNIVERSITY

Algorithm 1: Generate_Decision_Tree

```
Input: $D$, attribute_list, attribute_selection_method
Output: A decision tree $T$
create a node $N$;
if for all $\mathbf{x} \in D$, $c(\mathbf{x})$ are the same then
    return $N$ as a leaf node labeled with $c(\mathbf{x})$
if attribute_list is empty then
    return $N$ as a leaf node labeled with the majority class in $D$;
apply attribute_selection_method($D$, attribute_list) to find the best splitting
  criterion;
label node $N$ with splitting criterion;
attribute_list $=$ attribute_list - splitting_attribute;
for each outcome $j$ of splitting_criterion do
    Denote $D_j \subseteq D$ such that $D_j$ satisfies outcome $j$;
    if $D_j = \emptyset$ then
        attach a leaf labeled with the majority $D$ to node $N$;
    else
        attach the node returned by Generate_Decision_Tree($D_j$,
          attribute_list) to node $N$;
return $N$;
```

Information Gain as Selection Measure

- From the algorithm, we can see that finding the right splitting criterion is crucial for the structure of the tree, which further influences the test accuracy as well as robustness of the classifier. We will further our discussion in the following part.
- The ID3 algorithm utilizes information gain, which is essentially mutual information, as attribute selection measure. We formalize the problem as follows.
- Suppose we want to partition tuples in D on some attribute A having v distinct values $\{a_1, \ldots, a_v\}$. Since we are considering the discrete case, we can split D into v partitions such that $D_j \subseteq D$ satisfies $\mathbf{X} \in D_j$ and $\mathbf{X}_A = a_j$.Ideally, we intend to make each partition pure, i.e all elements in D_j belonging to the same class.

Information Gain as Selection Measure

The expected information needed to classify a tuple in D is its entropy

$$H(D) = -\sum_{i=1}^{m} p_i \log_2(p_i) \tag{5}$$

where p_i is the probability that $\mathbf{X} \in D$ belongs to class c_i, so it can be estimated by $\frac{|c_i|}{|D|}$. The conditional entropy $H(D|A)$ is defined as

$$H(D|A) = \sum_{j=1}^{v} p(A = v_j) H(D|A = v_j) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} H(D_j) \tag{6}$$

Thus, we have derived the formula for information gain, which is an effective measure on how much would be gained by branching on A.

$$I(D; A) = H(D) - H(D|A) \tag{7}$$

In ID3 algorithm, we chose the $splitting\_attribute$ as $\arg\max_A I(D; A)$.

C4.5 Algorithm

- The information gain measure is biased toward tests with many outcomes. That is, it prefers to select attributes having a large number of values.
- For example, consider an attribute that acts as a unique identifier such as product ID. A split on product ID would result in a large number of partitions (as many as there are so many kinds of values), each one containing just one tuple. Because each partition is pure, the information required to classify data set D based on this partitioning would be $\text{Info}_{\text{product\_ID}}(D) = 0$.
- Therefore, the information gained by partitioning on this attribute is maximal. However, such a partitioning is useless for classification.

C4.5 Algorithm

In order to solve this problem, C4.5 algorithm[4] proposes a new partition standard called Gain Ratio. In addition to using classical information entropy, the algorithm define an interesting "**split information**" to describe the attribute.

In detail, first of all we still calculate $I(A;D)$. Instead of using $I(A;D)$ as the only indicator of reference, we introduce $\text{SplitInfo}_A(D)$ to describe the instability of the attribute. $\text{SplitInfo}_A(D)$ is defined as:

$$\text{SplitInfo}_A(D) = -\sum_{i=1}^{v} \frac{|D_j|}{D} \times \log_2 \left(\frac{|D_j|}{D} \right) \tag{8}$$

It is not difficult to see the similarity between split information and entropy : split information just replace p_i in entropy with $\frac{D_j}{D}$.

[4] J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

C4.5 Algorithm

In C4.5 algorithm, split information is used to measure the uncertainty of A. This value represents the potential information generated by splitting the training data set, D, into v partitions, corresponding to the v outcomes of a test on attribute A. It considers the number of tuples having that outcome with respect to the total number of tuples in D. Finally, the Gain Ratio is defined as:

$$\text{GainRatio}(A) = \frac{I(A;D)}{\text{SplitInfo}_A(D)} \tag{9}$$

The attribute with the maximum gain ratio is selected as the splitting attribute. In other words, we'll tend to choose the attribute with the least uncertainty.

CART Algorithm

ID3 and CART[5] were invented independently of one another at around the same time, yet follow a similar approach for learning decision trees from training tuples.

However, unlike the two algorithms mentioned above, this algorithm does not use entropy as a measure, but instead uses the **Gini index** as a substitute. Using the notation previously described, the Gini index is defined as

$$\text{Gini}_A(D) = 1 - \sum_{i=1}^{m} p_i^2 \tag{10}$$

where p_i is the probability that a tuple in D belongs to class C_i and is estimated by $|C_{i,D}|/|D|$. The sum is computed over m classes. In fact, CART uses a first-order approximation to simplify logarithmic calculations : $\log x \sim 1 - x$. Due to the existence of approximation, we can consider the Gini index as a special kind of entropy.

[5] Leo Breiman. *Classification and regression trees*. Routledge, 2017.

CART Algorithm

When considering a binary split, we compute a weighted sum of the impurity of each resulting partition. For example, if a binary split on A partitions D into D_1 and D_2, the Gini index of D given that partitioning is

$$\text{Gini}(D) = \frac{|D_1|}{|D|}\text{Gini}(D_1) + \frac{|D_2|}{|D|}\text{Gini}(D_2) \tag{11}$$

For each attribute, each of the possible binary splits is considered. For a discrete-valued attribute, the subset that gives the minimum Gini index for that attribute is selected as its splitting subset.

CART Algorithm

Like ID3, when we need to process the attribute with continuous-valued, we'll determine the best **split-point** for it. For continuous-valued attributes, each possible split-point must be considered. The strategy is similar to that described earlier for information gain, where the midpoint between each pair of (sorted) adjacent values is taken as a possible split-point. The point giving the minimum Gini index for a given (continuous-valued) attribute is taken as the split-point of that attribute.

The reduction in impurity that would be incurred by a binary split on a discrete- or continuous-valued attribute A is

$$\Delta \text{Gini}(D) = \text{Gini}(D) - \text{Gini}_A(D) \tag{12}$$

The attribute that maximizes the reduction in impurity (or, equivalently, has the minimum Gini index) will be selected as the splitting attribute.

Tree Purning

SHANGHAI JIAO TONG UNIVERSITY

When a decision tree is built, many of the branches will reflect anomalies in the training data due to noise or outliers. Tree pruning methods address this problem of overfitting the data. There are two main approaches of tree pruning : pre-pruning approach and post-pruning approach.

In the prepruning approach, a tree is “pruned” by halting its construction early (e.g., by deciding not to further split or partition the subset of training tuples at a given node). Upon halting, the node becomes a leaf. The leaf may hold the most frequent class among the subset tuples or the probability distribution of those tuples.However,high thresholds could result in the failure of split leaves, which oversimplified the decision tree. Low thresholds could result in a few simplification.

The second and more common approach is postpruning, which removes subtrees from a “fully grown” tree. A subtree at a given node is pruned by removing its branches and replacing it with a leaf. In fact, in C4.5 and CART algorithm, they both use postpruning approach.

Tree Purning

- The cost complexity pruning algorithm used in CART is an example of the postpruning approach. It starts from the bottom of the tree. For each internal node, N, it computes the cost complexity of the subtree at N, and the cost complexity of the subtree at N if it were to be pruned. The two values are compared. If pruning the subtree at node N would result in a smaller cost complexity, then the subtree is pruned. Otherwise, it is kept.
- C4.5 uses a different method called **pessimistic pruning**, which is similar to the cost complexity method in that it also uses error rate estimates to make decisions regarding subtree pruning. Pessimistic pruning, uses the training set to estimate error rates. Recall that an estimate of accuracy or error based on the training set is overly optimistic and, therefore, strongly biased. The pessimistic pruning method therefore adjusts the error rates obtained from the training set by adding a penalty, so as to counter the bias incurred.

Results of Classic Algorithm

After adjusting the parameters and modifying the training set partitioning method, we obtained the results using the traditional three methods as shown in the table below

Table: Test results

ALG	Depth	Acc	ALG	Depth	Acc	ALG	Depth	Acc
gini	3	0.902778	entropy	3	0.902778	log_loss	3	0.902778
gini	4	0.819444	entropy	4	0.819444	log_loss	4	0.819444
gini	5	0.861111	entropy	5	0.861111	log_loss	5	0.861111
gini	6	0.861111	entropy	6	0.861111	log_loss	6	0.861111
gini	7	0.861111	entropy	7	0.861111	log_loss	7	0.861111
gini	None	0.851852	entropy	None	0.847222	log_loss	None	0.851852

Modified Pruning based on Dropout

Through the above analysis, it is not difficult to find that if the calculation method of entropy is fixed, the shape of this tree has been roughly determined.If the quality of the training set is poor, the decision tree we get will have the problem of **overfitting**. In order to solve the problem of overfitting, we propose the following method.

When considering the solution to this problem, we follow the idea of by *the dropout method* which is commonly used in the construction of neural networks in the field of machine learning. The core idea of the Dropout method can be summarized as follows: different fixed neural networks will have different level of overfitting, and multiple averaging may create some opposite effects.

Modified Pruning based on Dropout

SHANGHAI JIAO TONG UNIVERSITY

Specifically, we introduce a variable p that represents the possibility of splitting an attribute. It can be regarded as a hyper-parameter.

To deal with the problem that the average accuracy drops when p gradually increases, we consider the following.In fact, there is an inheritance relationship in the classification of decision trees. The splitting of the previous node will be directly retained in the classification criteria of the subsequent nodes. In order to eliminate the impact of parameter inheritance, we added a exponential correction factor q to adjust the probability of deletion between different levels.The equation is give by

$$\text{Dropout} = p(1+q)^l \tag{13}$$

where p is the initial value of deletion probability, and q is the correction factor for the number of layers and l is the level of the decision tree.

Modified Pruning based on Dropout

During the testing, we set p ranging from 0 to 0.25 with interval 0.05, and q ranging from 0.1 to 1 with interval 0.1. The result can be summarized in Figure 2.

We can see that the classical algorithm achieves an accuracy of 0.86 (also corresponds to the result in Table 1. The best accuracy achieved is 0.96 with $q > 0.9$. From Table 1, we can see that this is better than the best result achieved by the three classical algorithms. Multiple hyper-parameter settings also achieved decent performance. But we can also see that there are a noticeable number of parameter combinations that are quite poor. After repetitive testing, we can conclude that the accuracy 0.96 can be achieved with some $q \in [0.6, 1]$. This implies that with the help of adequate pruning, the running result of the algorithm can be improved without changing the attribute selection method.

Modified Pruning based on Dropout

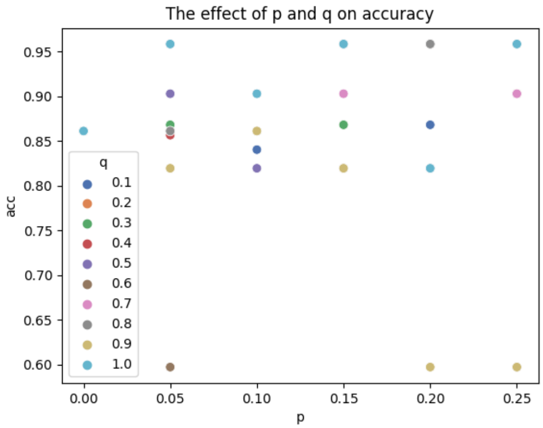


Figure: Running result of modified decision tree with the introduced parameter p, q

Strengths and Limitations

- The ID3 algorithm selects metrics based on maximizing the information gain, which is actually mutual information. In the case where training data is insufficient, the mutual information between certain attributes can be wrongly estimated.
- The method that we have proposed only keep those with the greatest information gain, and introduce randomness to deal with more trivial nodes. By selecting the right parameters, there stands a better chance of generalizing the model to more unfamiliar test cases.

To test the theory, we plot the training accuracy with respect to different p and q in Figure 3. We can see that the training accuracy of the classic ID3 algorithm is 1.0, which is a clear sign of overfitting as the test accuracy is only 0.86. On the other hand, the training accuracy of the proposed algorithm mostly floats around 0.85 to 0.95, which implies better flexibility.

Strengths and Limitations

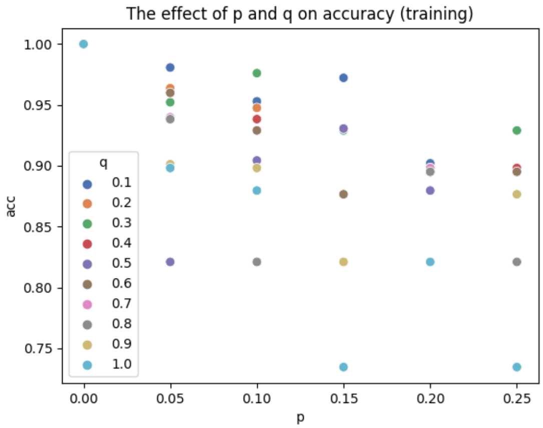
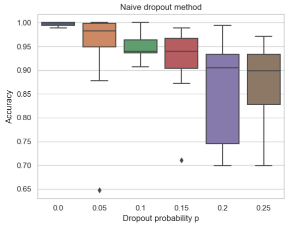
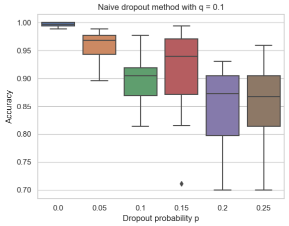


Figure: Training accuracy with respect to different p and q

Strengths and Limitations

However, if the training data is *accurate* enough, then the pruning algorithm can hurt the accuracy, as it potentially eliminates cases where it should be classified with more detail. To test our hypothesis, we run the algorithm with a shuffled dataset, which eliminates overfitting. The results are shown as follows. We can see that as p increases, the mean of the accuracy decreases, accompanied with increased variance. Nevertheless, the mean value of the accuracy remains acceptable, so setting p and q to a relatively low value is robust enough to deal with various test cases.





Strengths and Limitations

In conclusion, the introduction of hyper-parameters p and q can improve the performance of the decision tree algorithm under certain overfitting circumstances. By adjusting the values of p and q, we can find the optimal balance between underfitting and overfitting, and achieve better accuracy on new data. But it lacks stability and can perform worse if the training data is more accurate and well-defined.

References I

- Breiman, Leo. *Classification and regression trees*. Routledge, 2017.
- Mining, What Is Data. "Data mining: Concepts and techniques". In: *Morgan Kaufinann* 10 (2006), pp. 559–569.
- Quinlan, J Ross. *C4. 5: programs for machine learning*. Elsevier, 2014.
- — ."Induction of decision trees". In: *Machine learning* 1 (1986), pp. 81–106.
- Shannon, Claude E. "A mathematical theory of communication". In: *The Bell system technical journal* 27.3 (1948), pp. 379–423.

SHANGHAI JIAO TONG UNIVERSITY

Thank You

Zhu Pengxiang, Zhou Shengyang · Decision Trees for Classification