Report of Summer Internship

Decision Tree Distance Calculation

Author:
Fengli LIN

Supervisor:
Rémi Dehouck

September 4, 2019

1 Abstract

During my internship, the main project is to find a way to measure the distance between two Decision Trees(DTs). After full investigation, I found that there are mainly two directions to achieve this goal. The one is structural method [1] [4] [5], which mainly focus on the structural difference between two DTs. The other one is semantic method [3] [2], which on the contrary regardless of the structure details, only focuses on the output values. I have implemented three detailed algorithms: two structural algorithms(Edit distance based method, Substructure based method) and one semantic algorithm(Predicted value based method). The experiments results show that the Substructure method has the best performance in both robustness and time efficiency.

2 Introduction

First, the definition of Distance between Decision Trees is a number to quantify the degree of difference between DTs. To be notice that, the absolute value of this number has no great significance. Because we don't have the golden labels for training, we cannot say a value of 1 means little difference or a value of 9 means large difference. We can only say that if the distance between Tree A and Tree B is 1, while the distance between Tree A and Tree C is 1.5, in this case Tree C is more different to Tree A than Tree B to Tree A.

Next, why we need to calculate the Distance between Decision Trees? After discussing with the Customer Success Team, we conclude the potential usages as following:

- Track the evolving process of agents (measuring model/data stability, debugging)
- Detect anomaly(in the case of sudden change of model/data)
- Prevent model failure(forecast model behaviour) with the help of similar model records(Concept drift detection)
- Clustering agents(regroupe users)

3 Methodology

During my internship, I have implemented three concrete algorithm in Python. Those three algorithms are Edit distance based method, Substructure based method and Predict value based method.

3.1 Edit distance based method

3.1.1 Idea

The main idea of this method comes from the comparison of general trees. A popular method to compare the difference of two trees is to calculate edit distance between them. The formal definition of edit distance is as follows:

The tree edit distance between ordered labeled trees is the minimal-cost sequence of node edit operations that transforms one tree into another. We consider following three edit operations on labeled ordered trees:

- delete a node and connect its children to its parent maintaining the order.
- insert a node between an existing node and a subsequence of consecutive children of this node.
- rename the label of a node.

There are many different sequences that transform one tree into another. We assign a cost to each edit operation. Then, the cost of an edit sequence is the sum of the costs of its edit operations. Tree edit distance is the sequence with the minimal cost.

3.1.2 Implementation details

In the code Edit_distance.ipynb, I implemented this algorithm and two detailed use case: Clustering different agents and Measure evolving trend.

The process of the algorithm is as follows:

1. Change Craft_AI_DecisionTree type into so called bracket form. In this step, I use Depth First Search algorithm for conversion.
2. After getting the bracket form, I use the Python Library APTED to calculate the Edit distance between given trees.
3. Finally, I normalized the result through dividing the edit distance by average number of nodes of these two trees.

3.1.3 Drawbacks

The main drawback of edit distance method is its unstability. For example, if we encounter two mirror trees1, the edit distance will be large while they may not have big difference. Therefore, to deal with this problem, we will introduce the substructure based method.

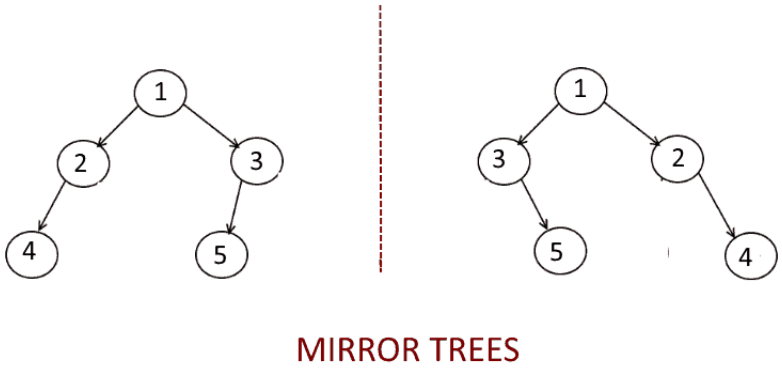


Figure 1: **Two Mirror Trees**

3.2 Substructure based method

3.2.1 Idea

Instead of calculating the minimum operations to change one tree to another, the substructure based method instead focuses on counting how many parts(substructures) of these two trees are identical. With the help of this method, we can focus more on the local parts of trees so that we can get more representative structural distance.

3.2.2 Implementation

I have implemented tree versions of Substructure based method: nonvalue_Substructure.ipynb, noncontinuous_Substructure.ipynb and Label_Substructure.ipynb. The difference of these versions are as follows:

- nonvalue_Substructure.ipynb: Only encode the output label and the attribute name of internal nodes.
- noncontinuous_Substructure.ipynb: Encode the output label and both the attribute name and value(except continuous values) of internal nodes.
- label_Substructure.ipynb: Encode all the output label and both the attribute name and value of internal nodes.

The general process of the algorithm is as follows:

1. Linearize the paths of DTs into sequences of branches(BSs). I use Depth First Search algorithm here and for the encode of continuous values, I apply the SpectralClustering algorithm to transfer the continous values into k(to be choose)-class labels.

2. Apply a sequential pattern mining algorithm(prefix-span algo) to find all frequent sequences(FSs). Prefix-span is an efficient algorithm to calculate the common sequences of input sets and their frequency.

3. Finally calculate the size of intersection between these two FSs and normalize it through dividing by total size of these FSs.

3.3 Predict value based method

The idea of semantic method comes from the phenomenon that some DTs, though structurally different, may describe the same concept, i.e., they may be semantically similar or even identical to each other. Semantic similarity in the presence of structural differences might arise for a variety of reasons, such as superficially different tests on attributes which are in fact equivalent, different attributes that convey the same information due to attribute redundancy, or simply because the same concept can be described in different ways which are nevertheless semantically equivalent.

3.3.1 Implementation details

In semantic_baseline.ipynb, I implemented a baseline method to capture the degree of semantic similarity between DTs: predict value based method. The detailed process of this method is as following:

1. Given two DTs and a dataset, we calculate the predict values of these two DTs.

2. We calculate the difference between these two predict values and obtain the difference by the formula

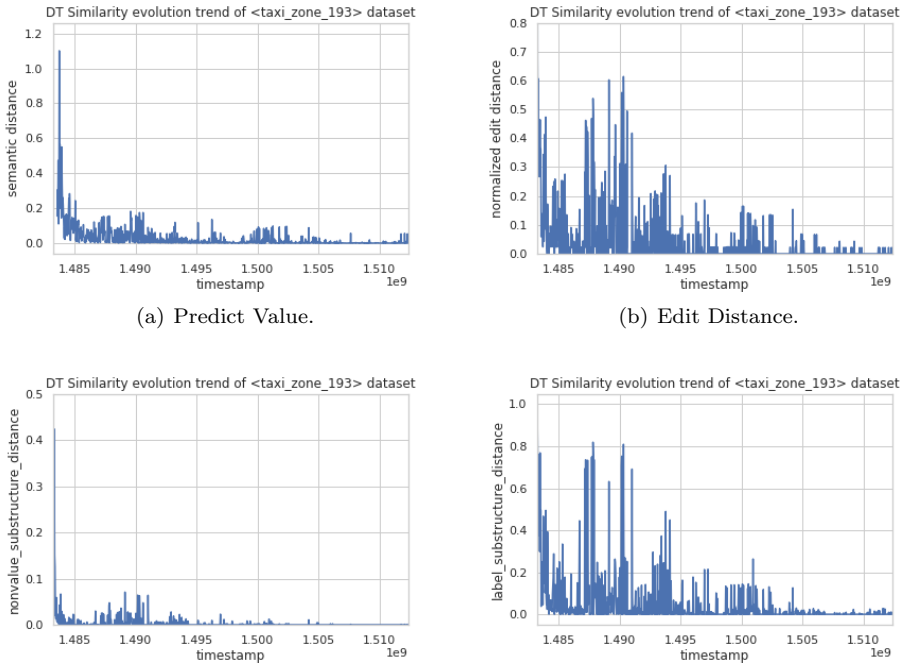
$$Distance = mean(diff) + std(diff)$$

3.3.2 Drawbacks

One main problem of this method is that we need to choose a test dataset and calculate the predict value. The potential risk is that the data we choosen doesn't have the same generation distribution of the target DTs. This may result in the unreliable shift of the resulted distance. Therefore, to use this method, it's better to know well about the generation distribution of dataset.

4 Experiments

I have done a set of experiments on both fake datasets and real datasets. The target of these experiments is to measure the performance on Evolving process, Detection of anomalies and Clustering.



(a) Predict Value.

(b) Edit Distance.

(c) Nonvalue Substructure.

(d) Label Substructure.

Figure 2: Evolving Process.

4.1 Evolving Process

For the experiment of Evolving Process, I calculate the distance between each nearby timestamps, such as t and t+1. Through these calculations, we can capture the sudden change and evolving process of a set of generated DTs as time go on.

Here, I test these algorithms on NYC Taxi and Limousine Commission (LTC) dataset.

We can see from Fig.2 that these methods can all successfully reflect the converge process of the training process. While generally speaking, these methods show different stability: the predict value method and nonvalue substructure method are more robust while the edit distance method

Methods	Edit distance	Nonvalue_Substructure	Nonconti_Substructure	Label_Substructure	Predicted values
Time used	52.785390948	9.6203202	31.281042096	41.140227564999996	578.519689588

Figure 3: Time Cost

and label substructure method are more sensitive. But by this dataset, we cannot conclude which method has better performance. Therefore I created some fake datasets for test.

4.2 Anomaly Detection

In the fake datasets, I change the generation formula of the dataset at certain timestamp and to see if our algorithms can detect this change.

From the results in Fig.4 and 5, we can see that the Predict Value method has the worst performance. This is because I use the last 30% data as the test data, which has different distribution to the previous DTs. Therefore, the calculated distance cannot obviously reflect the change at certain timestamps. For the structural methods, we can see that because of the sensibility of Edit Distance and Label Substructure, they could not even converge when the change happens(Fake Dataset_2). Therefore, compared to Nonvalue Substructure method, they are not stable enough.

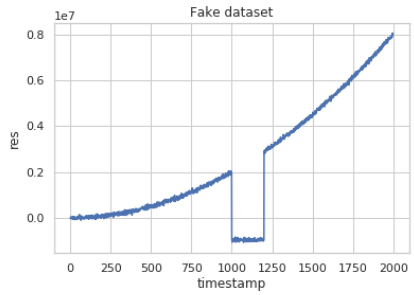
4.3 Clustering

For the experiment of clustering, I also first create 10 datasets. Each half of them has the similar distribution except that I add some shifts and noises. We can easily see that the Nonvalue Substructure method has the best performance on Fake Dataset. This is because it not consider the nodes' value, which means it focuses more on the general structure such that it can successfully cluster the similar agents. To be notice that, this method assumes that the nodes' values don't matter a lot for us.

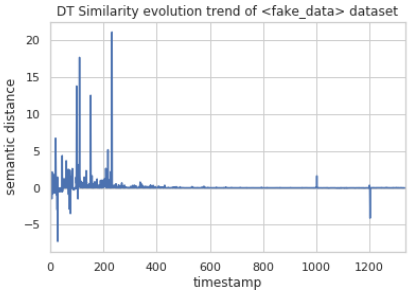
5 Discussion

In summary, the above experiments prove that if there is no particular request for concrete values, the Nonvalue Substructure method shows both stability and time efficiency. The Edit Distance method and Label Substructure method have the risk of overfitting, which means they consider too much the details and are sensitive to noises. For the semantic method, I found that the choice of test dataset influence the result a lot, which means this could not be a good option in real usage.

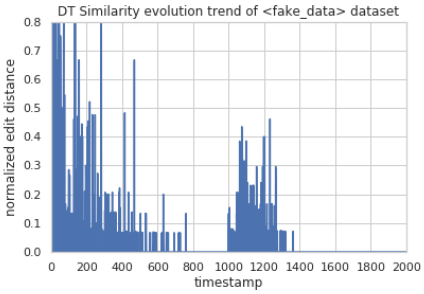
Through my survey, I find there is little work related to the calculation of Decision Tree Regressor Distance. Current works mainly focus on the Decision Tree Classifier and they don't provide a good method to treat continuous values. Therefore, the problem of continuous values can be a promising research direction. In my implementation, I simply try to convert the continuous values into discrete values using Spectral Clustering algorithm and the results are not very satisfying. It seems that there are still some space for improvement.



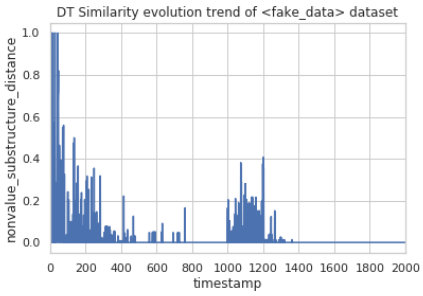
(a) Fake Dataset.



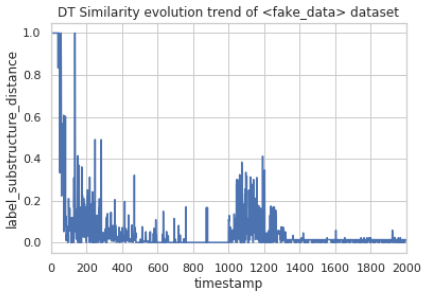
(b) Predict Value.



(c) Edit Distance.

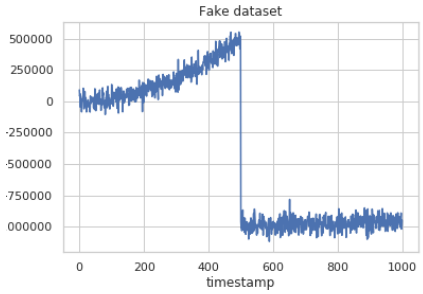


(d) Nonvalue Substructure.

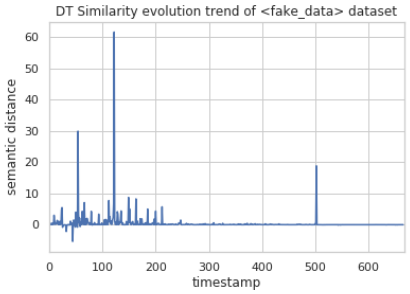


(e) Label Substructure.

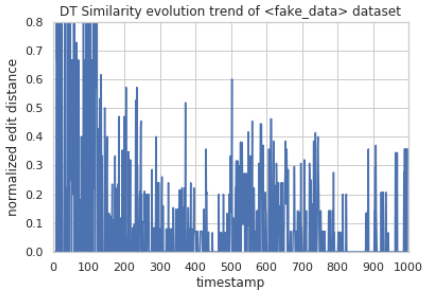
Figure 4: Anomaly Detection: Fake Dataset_1. (Change between [1000,1200])



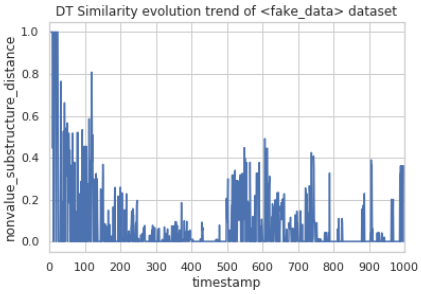
(a) Fake Dataset.



(b) Predict Value.

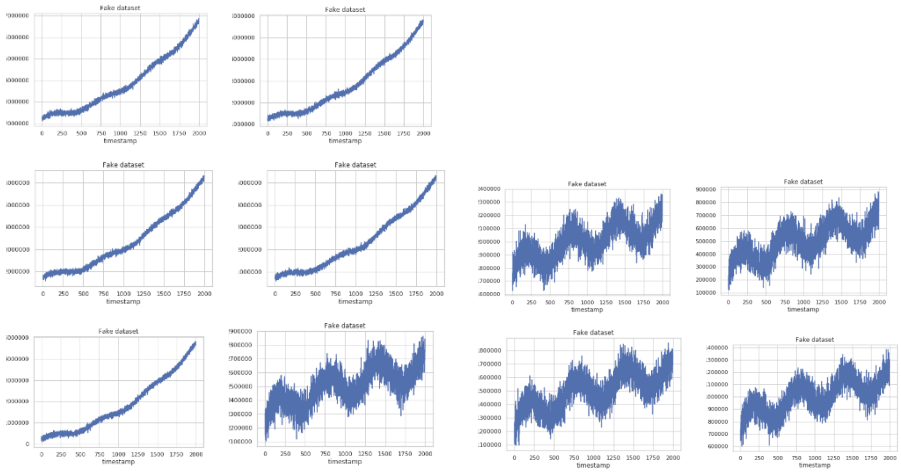


(c) Edit Distance.

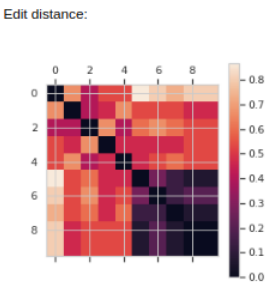


(d) Nonvalue Substructure.

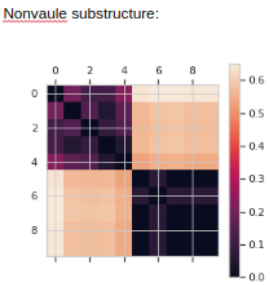
Figure 5: Anomaly Detection: Fake Dataset_2.(Change at 500)



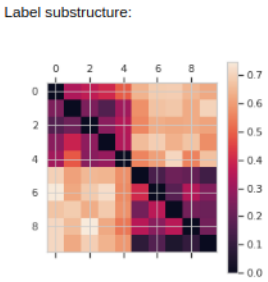
(a) Fake Dataset



(b) Edit Distance.



(c) Nonvalue Substructure.



(d) Label Substructure.

Figure 6: Clustering on Fake Dataset.

References

[1] GÖZDE BAKIRLI and Derya Birant. Dtreesim: A new approach to compute decision tree similarity using re-mining. *Turkish Journal of Electrical Engineering & Computer Sciences*, 25(1):108–125, 2017.

[2] Joao Gama, Pedro Medas, Gladys Castillo, and Pedro Rodrigues. Learning with drift detection. In *Brazilian symposium on artificial intelligence*, pages 286–295. Springer, 2004.

[3] Irene Ntoutsi, Alexandros Kalousis, and Yannis Theodoridis. A general framework for estimating similarity of datasets and decision trees: exploring semantic similarity of decision trees. In *Proceedings of the 2008 SIAM International Conference on Data Mining*, pages 810–821. SIAM, 2008.

[4] Petra Perner. How to compare and interpret two learnt decision trees from the same domain? In *2013 27th International Conference on Advanced Information Networking and Applications Workshops*, pages 318–322. IEEE, 2013.

[5] Abraham Itzhak Weinberg and Mark Last. Selecting a representative decision tree from an ensemble of decision-tree models for fast big data classification. *Journal of Big Data*, 6(1):23, 2019.