CS575: Final Project Report

Project Title: Decision trees and Travelling problem in real life.

Team Member: Sahil Mirchandani

I. Problem

There are Trees and Graph's that shows the relation between each attribute but where these relations came from and how their relations can benefit. So, I have used concept of trees and developed an algorithm of decision trees. In decision trees we pass a set of data and they help you to make classification decisions based on the data.

We have learnt about travelling salesman problem and how it can be solved by using various approaches such as Dynamic programming, greedy methods and backtracking but all of these algorithms require a distance matrix and number of nodes and distances between each of them specified which is not true in real life problems. I have solved this problem by implementing my own algorithm that would overcome some flaws in travelling salesman problem. My algorithm works on a real-world set of real time locations and we can add or remove any location and it would provide the shortest path to reach all the points with the non-polynomial complexity.

II. Algorithms

A. Decision Trees

In Decision Tree the major challenge is to identification of the attribute for the root node in each level. This process is known as attribute selection. We choose the method of finding Information Gain of every attribute at each level.

Information Gain is the entropy of the data for each attribute. Basically, the information that the attribute gives about the prediction label. The formula for finding Information gain is:

Suppose S is a set of instances, A is an attribute, S_v is the subset of S with A = v, and Values (A) is the set of all possible values of A, then

$$Gain(S,A) = Entropy(S) - \sum_{v \epsilon Values(A)} \frac{|S_v|}{|S|}.Entropy(S_v)$$

Where Entropy is,

$$E(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$

When we calculate Information Gain of each attribute, at each level we choose the attribute of highest Information Gain as It would be best for splitting data.

After we train the model on our data and calculate every split then we test the model on our test data set which provides the accuracy of the model.

Decision trees are famous for making Binary decisions such as weather would rain or sunny, person would buy or not buy a car etc. but decision trees over train on the model because of selecting numerous attributes that doesn't take part in making better decisions.

```
Algorithm:
ID3 (Examples, Target_Attribute, Attributes)
    1. Create a root node for the tree
    2. If all examples are positive, Return the single-node
       tree Root, with label = +.
    3. If all examples are negative, Return the single-node
       tree Root, with label = -.
    4. If number of predicting attributes is empty, then
       Return the single node tree Root, with label = most
       common value of the target attribute in the
       examples.
    5. Otherwise Begin
        a) A ← The Attribute that best classifies examples.
           It is decided by finding Information Gain of
           every attribute and selecting the one with highest
           Gain
        b) Decision Tree attribute for Root = A.
       For each possible value, (class)$v_i$, of A,
            a) Add a new tree branch below Root,
               corresponding to the test A = $v_i$.
            b) Let Examples($v_i$) be the subset of examples
               that have the value $v_i$ for A
            If Examples($v_i$) is empty
                • Then below this new branch add a leaf
                  node with label = most common target
                  value in the examples
            Else
                • below this new branch add the subtree
                  ID3 (Examples($v_i$), Target_Attribute,
                  Attributes – {A})
    End
    Return Root
```

Hence, this is the Algorithm for Creating decision trees from the data. This decision trees helps to make better prediction to the data available.

B. Travelling problem

In travelling salesman problem, the main issue was the polynomial time complexity. The complexity of the n nodes is n! combinations to find the optimal route of n nodes. If we apply this algorithm on the real-world example with multiple address to find the shortest route between all the nodes. As there would be multiple address n! would take a large amount of time to compute the optimal distance. Hence algorithm would take a significant amount of time to compute the minimal distance between the points. It won't provide the optimal solution, but the time taken by algorithm is significantly less than available solutions for travelling salesman problem.

The algorithm takes input of a text file containing all the address that a travelling man has to visit. The first address is the starting point for the algorithm. Then it takes all the address as the input and forms clusters with respect to the distance between them. We can manually set the cluster size to compute clusters. Cluster size is the accepted distance between the address. Algorithm takes O(kn) time to form clusters. Where k is the number of clusters. The worst case would be every point is farthest from each other or our cluster size is inappropriate according to the data input then it would take O(n^2) complexity to take input and form k clusters.

After cluster formation we sort the cluster according to their midpoints to visit the. For Example, we have 5 clusters then the order of travel could be [1,3,4,5,2] or etc. The algorithm would take O(k^2) time to sort the clusters as we don't have distances between midpoints of the clusters hence, We need to compute distance between them and then make an order of the clusters.

After we have made an order of the cluster we need to sort the internal address of the cluster so as to visit the internal clusters in the minimal order possible.

```
Algorithm:
    1. While read each line from the text file;
        a. If no clusters formed
            i. Form cluster
        b. Else:
            i. Compute distance between the address and clusters midpoints
            ii. If its within cluster size then add address to the cluster
            iii. Else: make a new cluster with the address
    2. Add the first cluster into sorted order list
    3. For each midpoint of cluster:
        a. Compute the cluster midpoint with the shortest distance between the current_ midpoint and last element in the sorted order list
        b. Return the sorted order list
    4. For each cluster in sorted order list:
        a. Sort the internal elements of the cluster
        b. Return sorted cluster
    5. Make a result_list according to the sorted order list with sorted internal clusters.
    6. Return result_list
```

III. Software Design and Implementation

A. Software Design

Decision Trees

I have used the python as the programming tool and Jupyter notebook as the software tool to develop this algorithm. Along with it I have used libraries like pandas, NumPy and math to handle and manipulate data. I have used ID3 (Iterative Dichotomiser 3) algorithm that uses the concept of Information gain to split data at every level of building a tree. Number of children is the number of unique classes the node has. ID3 algorithm is biased on the data which has maximum samples. Hence, we use methods like post and pre purring to make the tree unbiased and increase the accuracy of the model.

Travelling Problem

I have used the python as the programming tool and Jupyter notebook as the software tool to develop this algorithm. Along with it I have used libraries such as google-maps and geodesic and Api of google maps to find the distance between address as I am working on real world data. I have used arbitrary set of address to test the algorithm. Algorithm computes the overall minimal distance between the address or data points. But it does not provide an optimal solution to the problem.

B. Implementation and Tools Used

Decision Trees

I have used the python as a developing tool and developed many functions to do my work seamlessly. At first, I have read the data into a pandas Data-frame. Then I have pass the Data-frame int a function Build_nodes.

Build_nodes: It takes the parameter as data frame and then it separates the last column that is known as the decision-making column(Y). Then, we find the total Information Gain of the model which is given by finding information gain of parameter (Y).

If the total Information Gain is 0 then it indicates we don't have to make a prediction as no attribute is contributing to the prediction and there is only one class in the output Y.

Else, we start building the tree. First find the Information Gain of the entire dataset excluding the output parameter Y. and we select the attribute which gives the highest Information Gain and make that attribute as the root node and after doing that we try to find gain and choosing the param with max gain again and again until we find the Information gain to be 0. That indicates we don't have to split the data more.

FindGain:

This function is required to find gain of each attribute to return the value to build nodes to select the attribute with max gain. The formula for the gain is

$$Gain(S, A) = Entropy(S) - \sum_{v\epsilon Values(A)} \frac{|S_v|}{|S|}.Entropy(S_v)$$

The information gain is calculated by finding entropy of each attribute hence, to find entropy of each attribute we call the function FindEntropy().

FindEntropy:

The function find entropy return the value of the entropy of the attribute hence, It takes the value of an attribute(column). The formula for calculating entropy is,

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

Hence, Entropy is an import feature while calculating the gain. Of an attribute.

Travelling Problem

I have used the python as the programming tool develop this algorithm. Along with it I have used libraries such as google-maps and geodesic and Api of google maps to find the distance between address. At first, we read the address line by line then we form clusters as mentioned in the Algorithm. After we make clusters, we pass the clusters in the functions below,

makeOrder:

This function makes an order of the clusters according to its midpoint and then returns the sorted order clusters that we use for passing it into different functions. It takes a list of midpoints as the input parameter and returns a list of sorted elements as the return value.

sortClusters():

When we get the sorted clusters then we pass each midpoint into the sorted cluster which sorts the internal elements in the cluster. As to optimize the internal path of each cluster. It takes parameters a cluster and returns a sorted cluster.

calculateMiles():

This is a function developed for testing purposes It takes a parameter a file and it iterates through each address and returns the total miles we have to travel to reach all the nodes.

After we have formed sorted order of clusters and the sorted internal clusters, we append the path into a result file. To test our implementation, we can pass the original and the result file into the calculate miles function.

According to the evaluation of our algorithm we have took a random set of local address and test on our algorithm. It has a significant decrease in the number of miles to travel from 246 miles to 123 miles. It is not the optimal path, but the decrease is significant. Also, the time complexity has a significant improvement from n! to n^2 which is great as implementation.

/* If you have compared the performance of several algorithms, describe 1) performance metrics (e.g., latency or throughput) and 2) results using the defined metrics. To illustrate your results, plot graphs or tables and clearly explain them. Note that performance evaluation is not required but optional, since correctly implementing N+1 non-trivial algorithms is the first priority. If you haven't done any performance evaluation just leave this section empty. */

ATTACHMENTS

- GitHub: https://github.com/sahil-mirchandani29/School-projects/tree/master/Data%20Analysis%20and%20Algorithms%20Project
- Video: <https://youtu.be/-J3Z-TbVi8A>
- Slides: <https://docs.google.com/presentation/d/1GB15zLBxzul8qQKoj4yuOnnUailvDTnmOfHL35DLcJ8/edit?usp=sharing >

REFERENCES

[1] En.wikipedia.org. 2020. *ID3 Algorithm*. [online] Available at: <https://en.wikipedia.org/wiki/ID3_algorithm >.

[2] En.wikipedia.org. 2020. *Travelling salesman problem*. [online] Available at: <https://en.wikipedia.org/wiki/Travelling_salesman_problem>.