Fast Attribute-based Unsupervised and Supervised Table Clustering using P-Trees


Department of Computer Science
North Dakota State University
Fargo, ND, 58102, USA
{arjun.roy, matthew.piehl, arijit.chatterjee, mohammad.hossain, bryan.mesich, tingda.lu, g.w, william.perrizo}@nds.edu, shehan@uom.lk

Abstract

Since the advent of digital image technology and remote sensing imagery (RSI), massive amount of image data has been collected worldwide. For example, since 1972, NASA and U.S. Geological Survey through the Landsat Data Continuity Mission, has been capturing images of Earth down to 15 meter resolution. Since image clustering is time-consuming, much of this data is archived even before analysis. In this paper, we propose a novel and extremely fast algorithm called FAUST_P or Fast Attribute-based Unsupervised and Supervised Table Clustering for images. Our algorithm is based on Predicate-Trees which are compressed, lossless and data-mining-ready data structures. Without compromising much on the accuracy, our algorithm is fast and can be effectively used in high-speed image data analysis.

1 INTRODUCTION

Digital Image Technology developed in 1970s has caused an exponential growth in the amount of image data throughout the world. This includes, personal photographs, scientific data such as remote sensing imagery, etc. For e.g. the Landsat Data Continuity Mission by NASA and U.S. Geological Survey has been providing image captures of Earth with moderate resolution from 15 meters to 100 meters for the last 38 years [1]. Demand for a fast clustering algorithm has increased due to the current scenario of usage of Unmanned Air Vehicles for security purpose where there is massive data collection and classifying objects of interest is of utmost importance. Due to slowness of existing algorithms, much of these data is archived without proper analysis.

In our paper, we propose an extremely fast supervised clustering algorithm for images. We use P-Trees to classify images as explained in the following sections. These data structures basically convert the attributes comprising the image into vertical strips of binary data. We then perform P-Tree specific operations to achieve extremely fast results. The overall structure of this paper is as follows: In the next section, we provide background information about P-Trees. In section 3, we propose our novel FAUST_P algorithm and explain it using the IRIS dataset available from [4]. In section 4, we give the implementation details followed by the algorithm analysis in section 5. Finally, in section 6, we state our conclusion and provide a list of future work based on our existing algorithm.

2 P-TREES

In this section, we give the background information about the P-Trees which form the basis of our algorithm. As mentioned above, P-Trees are data-mining-ready, compressed and lossless data structures. Simplest forms are the Peano-Trees which are bitwise trees and can be 1, 2 or n-dimensional depending on their application. For e.g. a spatial image can be efficiently represented by 2-dimensional Peano-Trees. For more information on P-Tree structure, construction and operations, please refer to [3]. In this paper, we only provide FAUST_P algorithm specific details about P-Trees. More specifically, we discuss the inequality Predicate-Trees which are used in evaluating the range predictions.

2.1 Inequality P-Trees

The Inequality P-Trees are used in evaluating range predictions. They represent data points within a dataset which satisfy the inequality predicates like \( x \geq y, x \leq v \), etc. A P-Tree representing \( P_{x\leq v} \) is computed as follows: Consider \( x \) to be a data point inside a dataset \( X \). Let \( x \) be an \( m \)-bit data and let \( P_m, P_{m-1}, …, P_0 \) be the P-Trees representing the dataset \( X \). Let \( v \) be the value represented by \( b_m, b_{m-1}, …, b_0 \) for which we need to find the equality. Then a Predicate Tree representing the predicate \( x \geq y \) is given by \( P_{x\geq y} = P_m \odot P_{m-1} \ldots P_1 \odot P_0 \), where \( i \) varies from 1 to \( m \). Also, \( \odot \) is a binary and operation if \( b_i \) is 1 and a binary or operation if \( b_i \) is 0. The operators are right binding. For e.g. \( P_{x \geq 14} \) or \( P_{x \geq 1110} \) can be computed as \((P_3 \land (P_2 \land (P_1 \lor P_0)))\) Values less than 14
in the given example can be computed by taking the complement of the Predicate tree.

### 2.2 Computation of Mean using P-Trees

Our algorithm uses computation of mean from the vertical representation of the data. P-Trees can be efficiently used to compute the aggregate functions such as count, sum, mean, max, min, etc. For e.g. count is nothing but the RootCount of the P-Trees. Sum function can be computed using the following algorithm:

$$
total \leftarrow 0;
$$

for i from n - 1 to 0

$$
total \leftarrow total + 2^i \times \text{RootCount}(P_i)
$$

Here, \(n\) is the number of bits representing the attribute or column of the table. The mean can be computed by \(\text{total} / \text{count}\). A detailed research on max, min and other aggregate functions is given in [2].

### 3 FAUST_P ALGORITHM

In this section, we propose our algorithm called the FAUST_P or Fast Attribute based Unsupervised and Supervised Table Clustering and explain it using the IRIS dataset. The dataset is freely available from the UCI Machine Learning Laboratory [4]. The pseudo code of the algorithm is as follows:

Initially, let \(P_{\text{REMAINING}}\) be a pure-1 P-Tree. From the training set:

1. For each attribute, calculate the mean for each class, sort ascending according to the mean. Calculate all the mean gaps = difference of consecutive means. Create MeanTable(Class, Attribute, Mean, \(\text{gap}_L\), \(\text{gap}_H\), \(\text{gap}_{\text{RELATIVE}}\)) and sort descending on \(\text{gap}_{\text{RELATIVE}} = (\text{gap}_L + \text{gap}_H) / 2 \times \text{mean}\). \(\text{gap}_L\) and \(\text{gap}_H\) are gap on the low and high side of the mean.

2. Choose and remove a MeanTable record with maximum \(\text{gap}_{\text{RELATIVE}}\). Calculate \(C_L = \text{Mean} - \text{gap}_L / 2\) to produce \(P_L = P_{\text{A+CL}}\). Calculate \(C_H = \text{Mean} + \text{gap}_H / 2\) to produce \(P_H = P_{\text{A+CH}}\). The class mask is \(P_{\text{CLASS}} = P_L \& P_H \& P_{\text{REMAINING}}\). Update \(P_{\text{REMAINING}} = P_{\text{REMAINING}} \& P_{\text{CLASS}}\).

3. Repeat step 2 above until all the classes have a P-tree.

4. Repeat steps 1, 2 and 3 until means stop changing (much).

The algorithm functions on IRIS data in the following way. We first remove the decimal point from all the data points to use integer values. Consider 10 random samples from each of the class, i.e. Iris-setosa, Iris-versicolor and Iris-virginica. Because of space limitation, we do not show the 30 points (in total) but their means along with mean gaps are listed in the table below.

<table>
<thead>
<tr>
<th>SL</th>
<th>M</th>
<th>mg</th>
<th>SW</th>
<th>m</th>
<th>mg</th>
<th>PL</th>
<th>M</th>
<th>mg</th>
<th>PW</th>
<th>m</th>
<th>mg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Se</td>
<td>51</td>
<td>12</td>
<td>63</td>
<td>1</td>
<td>70</td>
<td>2</td>
<td>32</td>
<td>1</td>
<td>32</td>
<td>2</td>
<td>35</td>
</tr>
<tr>
<td>Vi</td>
<td>14</td>
<td>33</td>
<td>60</td>
<td>13</td>
<td>11</td>
<td>47</td>
<td>12</td>
<td>25</td>
<td>12</td>
<td>2</td>
<td>60</td>
</tr>
</tbody>
</table>

| Table 1: IRIS data means and mean gaps |

SL, SW, PL and PW are the four attributes while se, vi and ve are the three classes defined in IRIS dataset. m and mg represent the mean and mean gap. We now create the MeanTable as defined in the algorithm.

<table>
<thead>
<tr>
<th>cl</th>
<th>Att</th>
<th>M</th>
<th>gapL</th>
<th>gapH</th>
<th>gR</th>
</tr>
</thead>
<tbody>
<tr>
<td>se</td>
<td>SL</td>
<td>51</td>
<td>12</td>
<td>12</td>
<td>0.235</td>
</tr>
<tr>
<td>se</td>
<td>SW</td>
<td>35</td>
<td>2</td>
<td>2</td>
<td>0.057</td>
</tr>
<tr>
<td>se</td>
<td>PL</td>
<td>14</td>
<td>33</td>
<td>33</td>
<td>2.357</td>
</tr>
<tr>
<td>se</td>
<td>PW</td>
<td>2</td>
<td>12</td>
<td>12</td>
<td>6</td>
</tr>
<tr>
<td>vi</td>
<td>SL</td>
<td>63</td>
<td>12</td>
<td>7</td>
<td>0.15</td>
</tr>
<tr>
<td>vi</td>
<td>SW</td>
<td>33</td>
<td>1</td>
<td>2</td>
<td>0.045</td>
</tr>
<tr>
<td>vi</td>
<td>PL</td>
<td>60</td>
<td>13</td>
<td>13</td>
<td>0.217</td>
</tr>
<tr>
<td>vi</td>
<td>PW</td>
<td>25</td>
<td>11</td>
<td>11</td>
<td>0.44</td>
</tr>
<tr>
<td>ve</td>
<td>SL</td>
<td>70</td>
<td>7</td>
<td>7</td>
<td>0.1</td>
</tr>
<tr>
<td>ve</td>
<td>SW</td>
<td>32</td>
<td>1</td>
<td>1</td>
<td>0.031</td>
</tr>
<tr>
<td>ve</td>
<td>PL</td>
<td>47</td>
<td>33</td>
<td>13</td>
<td>0.489</td>
</tr>
<tr>
<td>ve</td>
<td>PW</td>
<td>14</td>
<td>12</td>
<td>11</td>
<td>0.821</td>
</tr>
</tbody>
</table>

| Table 2: IRIS data MeanTable |

Since 6, 0.821 and 0.44 are the highest relative gaps among their class, they represent the cut point for classification. It may be noted that PW is the most relevant attribute in separating out one class from the other. Clearly, Iris-setosa is the first class to be classified since it has the maximum relative gap. We now calculate the \(C_L = 2 - 12 / 2 = -4\) and \(C_H = 2 + 12 / 2 = 8\). Since \(C_L\) is negative, it is assigned a pure-1 P-Tree. For \(C_H\), we convert 8 to binary 01000 and apply the Predicate Tree formula, i.e. \(P_{\text{A+CH}} = (P_{\text{A+CL}} \& (P_{\text{A+CL}} \& P_{\text{A+CH}})).\) Here, \(P_{\text{A+CL}}, P_{\text{A+CH}}, P_{\text{A+CL}}\) and \(P_{\text{A+CH}}\) are the P-Tree representing the PW attribute for the entire dataset. Thus \(P_{\text{setosa}} = P_{\text{A+CL}} \& P_{\text{A+CH}} \& P_{\text{REMAINING}}\) now holds only those data points which have not been classified as \(P_{\text{setosa}}\). We

![Figure 1: FAUST_P Algorithm](image)
repeat the above procedure to classify $P_{versicolor}$ and $P_{virginica}$.

4 IMPLEMENTATION

We execute our algorithm on a standard Ubuntu Linux machine with JAVA and GNU g++ compiler installed on it. We use an extractor program written in JAVA to extract the Red, Green and Blue values along with the x,y pixel coordinates of the image. The algorithm has been implemented in standard C++ language. The program is generalized to take any number of classes and attributes as input. To calculate the speed, we use the standard gettimeofday system call with a resolution of 1 microsecond. We test our algorithm on several test images. Most primitive among them is the aerial view of a car park to distinguish between cars of different color as well as road and shade. We also test on the aerial view of Red River near Fargo, North Dakota where we define 6 classes namely green farm, forest area, shade area, barren brown land, water and house.

5 ALGORITHM ANALYSIS

As can be analyzed from the algorithm defined in section 3, FAUST_P algorithm has a complexity of $O(k)$ where $k$ is the number of attributes or columns. This is extremely fast considering the fact that all the horizontal methods have at least $O(n)$ assuming no suitable indexing. The value of $k$ is generally small ranging from 2 to 7 (in case of Landsat data). Even high-attributed images with $k$ of for e.g. 200 can be rapidly classified in comparison to horizontal methods where n are of the order of 1 billion or even more. Our algorithm achieves an accuracy of 95% on IRIS dataset with only 1 epoch. Higher accuracy can be achieved at the cost of time.

6 CONCLUSION AND FUTURE WORK

In this paper, we propose a fast attribute-based unsupervised and supervised clustering algorithm for images. Our algorithm is extremely fast with a small compromise on the accuracy. This can be effectively used for Landsat data or any other kind of image data where rate of data generation is much higher than time to analyze it. In our future work, we plan to propose a divisive method which considers all data points in one cluster initially and splits depending on maximum gap.

Also, since mean is sensitive to outliers, use of variance is preferred in future algorithms.

7 REFERENCES