On Mining Satellite and Other Remotely Sensed Images [1, 2]

William Perrizo, Qin Ding, Qiang Ding, Amalendu Roy
Department of Computer Science,
North Dakota State University
Fargo, ND 58105-5164
{William_Perrizo, Qin_Ding, Qiang_Ding, Amalendu_Roy}@ndsu.nodak.edu

Abstract

Advanced data mining technologies and the large quantities of Remotely Sensed Imagery provide a data mining opportunity with high potential for useful results. Extracting interesting patterns and rules from data sets composed of images and associated ground data, can be of importance in precision agriculture, community planning, resource discovery and other areas. However, in most cases the image data sizes are too large to be mined in a reasonable amount of time using standard methods. A new spatial data organization, bit Sequential organization (bSQ) and a new “data-mining ready” data structure, the Peano Count Tree (Ptree) provide a lossless and compressed representation of image data which facilitate association rule mining, classification and other data mining techniques markedly. In this paper we propose a new model for association rule mining and classification on spatial datasets using Ptrees. Experimental results show the new model applies well to data mining on Remotely Sensed Imagery data.

Keywords: Data mining, Remotely Sensed Imagery (RSI), Association Rule Mining, Classification

1. Introduction

Data Mining is becoming more and more important as large quantities of data are generated and collected rapidly. Association rule mining and classification are two of the basic approaches in data mining.

The task of association rule mining (ARM) [8,9,10,11,12] is to find interesting relationships from the data in the form of rules. The initial application of association rule mining was on market basket data.

In the spatial data domain, Association Rule Mining is useful in identifying forest fires, insect and weed infestations, high and low crop yields, flooding and other phenomena as rule consequences. The antecedents of such rules are typically taken from the Remotely Sensed Imagery (RSI) data bands.

An association rule is a relationship of the form X=>Y, where X and Y are sets of items. X is called antecedent and Y is called the consequence. There are two primary measures, support and confidence, used in assessing the quality of the rules. The goal of association rule mining is to find all the rules with support and confidence exceeding user specified thresholds. The first step in a basic ARM algorithm (e.g., Apriori[8] and DHP[11]) is to find all frequent itemsets whose supports are above the minimal threshold. The second step is to derive high confidence rules supported by those frequent itemsets. The first step is the key issue in terms of efficiency.

Classification is another useful approach to mining information from spatial data. In classification, a training (learning) set is identified for the construction of a classifier. Each record in the training set has several attributes. There is one attribute, called goal or class label attribute, which indicates the class to which each record belongs. A test set is used to test the accuracy of the classifier once it has been developed from the learning dataset. The classifier, once certified, is used to predict the class label of unclassified data. Different models have been proposed for classification, such as decision tree induction, neural network, Bayesian, fuzzy set, nearest neighbor and so on. Among these models, decision tree induction is widely used for classification, such as ID3, C4.5[1,2], CART[4], Interval Classifier[3], SPRINT[3,5] and BOAT[6].

Both association rule mining and classification have been applied in many fields. Remotely Sensed image data is one of the promising application areas since there are huge amount of image data. However, due to the large size of image data, the existing methods are not very suitable. In this paper, we propose a new and efficient model to perform both association rule mining and classification on remotely sensed images data.

[1] Partially funded by GSA Grant ACT#K96130308
[2] Ptree technology is patented to North Dakota State University.

An image can be viewed as a 2-dimensional array of pixels. Associated with each pixel are various attributes or bands, such as visible reflectance intensities (Blue, Green and Red), infrared reflectance intensities (e.g., NIR, MIR1, MIR2 and TIR) and possibly other value bands (e.g., yield quantities, quality measures, soil attributes and radar reflectance intensities). The pixel coordinates in raster order constitute the key attribute. In this paper, our task is to derive rules among spectral bands and yield. We use two different approaches, association rule mining and classification, to perform the same task. In association rule mining, we try to derive rules with yield as consequent. A rule like "Green[192,255] ^ NIR[0,63] => Yield [128, 255]" is expected, where Green[192,255] indicates an interval with value ranged from 192 to 255 in Green band. In classification, we specify the yield as the goal attribute. These kinds of rules are particularly useful for future yield prediction both for experts and farmers. For example, if low yield is predicted in the current growing year, additional inputs can be applied, such as water and nitrogen, with high likelihood of increasing the yield.

A new lossless data structure, Peano Count Tree (Ptree), is used in the model. Ptrees represent image data bit by bit in a recursive quadrant-by-quadrant arrangement. With the information in Ptrees, we can design fast mining algorithms.

The rest of the paper is organized as follows. In section 2, we introduce the data formats of Remotely Sensed Images data, including a new format called bit Sequential Format (bSQ). We also describe the Ptree data structure and its algebra. In Section 3, we detail our algorithms for association rule mining and classification on RSI data using Ptree. Performance analysis is given in Section 4. Related work is given in Section 5, while conclusion and future work is given in Section 6.

2. Remotely Sensed Imagery Data and Ptree Data Structure

2.1 Remotely Sensed Images

There are different types of RSI images, such as TM, SPOT, AVHRR, TIFF, etc. For example, TM image (Thematic Mapper) contains 7 bands, which are B (Blue), G (Green), G (Red), RIR (Reflective-Infrared), MIR (Mid-Infrared), TIR (Thermal-Infrared), and MIR2 (Mid_Infrared2). The reflectance value in each band ranges from 0 to 255. Typically, a TM scene contains 40M pixels.

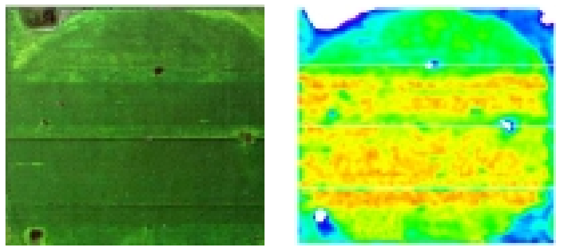

Figure 1 TIFF image and its yield map

In precision agriculture, some ground data are also very useful. For example, yield (production) is one of the most important ground data. A Yield map is a map of yield levels, either expressed using a color legend or using gray-scale levels. Figure 1 gives a TIFF image and its yield map.

Several formats used for RSI data are Band Sequential (BSQ), Band Interleaved by Line (BIL) and Band Interleaved by Pixel (BIP) format. In BSQ format, each band is stored as a separate file. Raster order is used within each band. BIL format stores data in line-major order while BIP format stores data in pixel-major order. A TM scene uses BSQ format, SPOT image uses BIL format while TIFF image uses BIP format.

In this paper, we propose a new format, called bit Sequential (bSQ) Organization to represent image data. A reflectance value in a band is a number in the range 0-255 and is represented as an 8-bit byte. We split each band into eight separate files, one for each bit position. Thus for each band there are eight separate bSQ files. There are several reasons to use the bSQ format. First, different bits contribute to the value differently. In some applications, the high-order bits alone provide the necessary information. Second, the bSQ format facilitates the representation of a precision hierarchy (from one bit precision up to 8 bit precision). Third, and most importantly, the bSQ format facilitates the creation of an efficient, rich data structure, the Ptree, and accommodates fast mining algorithms.

2.2 Basic Ptree Data Structure

We organize each bSQ bit file, B_{ij}, into a tree structure, called a Peano Count Tree (Ptree). A Ptree is a quadrant-based tree. The root of a Ptree contains the 1-bit count of the entire bit-band. The next level of the tree contains the 1-bit counts of the four quadrants in raster order. This construction is continued recursively down each tree path until the sub-quadrant is pure (entirely 1-bits or entirely 0-bits), which may or may not be at the leaf level (1-by-1 sub-quadrant). An 8-row-8-column bit-band example is shown next.

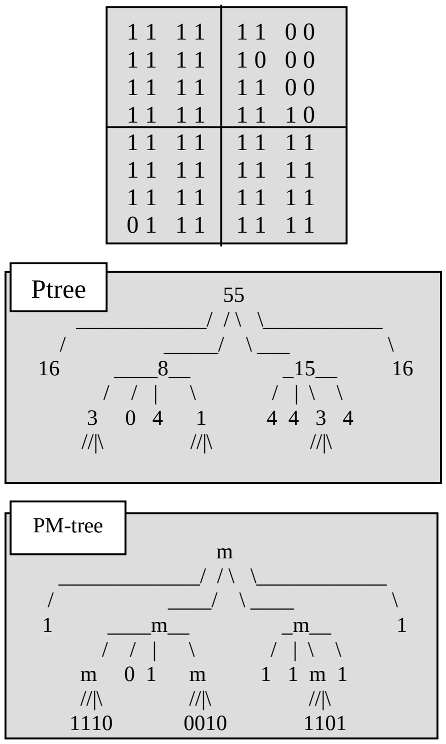

Figure 2. 8×8 bit image and its Ptree
(its basic Ptree and its PM-tree variation)

In this example, 55 is the count of 1's in the entire image, called root count. The numbers at the next level, 16, 8, 15 and 16, are the 1-bit counts for the four major quadrants. Since the first and last quadrant is made up of entirely 1-bits (called pure-1 quadrant), we do not need sub-trees for these two quadrants. Similarly, quadrant with entirely 0 bits is called pure-0 quadrant. This pattern is continued recursively. Piano or Z-ordering is used in partition – therefore, the name Peano Count trees. The process terminates at the "leaf" level where each quadrant is a 1-row-1-column quadrant. If we were to expand all sub-trees, including pure quadrants, the leaf sequence is just the Peano space-filling curve for the original raster image.

We note that, the fan-out of a P-tree need not necessarily be 4. Also, the fan-out at any one level need not coincide with the fan-out at another level. The fan-out pattern can be chosen to produce good compression for each bSQ band. Finally, we note that the same general construction can be used for spatial data of more than 2-dimensions. For 3-dimensional data, for instance, at each level, we partition into octants, and so forth. This structure applies well when the data have clustered properties, for example, data represented by sparse matrix.

For each band (assuming 8-bit data values), we get 8 basic Ptrees, one for each bit positions. For band B_i, we will label the basic Ptrees, $P_{i,1}$, $P_{i,2}$, …, $P_{i,8}$, thus, $P_{i,j}$ is a lossless representation of the j^{th} bits of the values from the i^{th} band. However, P_{ij} provides much more information and are structured to facilitate many important data mining processes.

A variation of Ptree, called Peano Mask Tree (PM-tree), can be used for efficient implementation of Ptree operations. In the PM-tree structure, we use a 3-value logic, in which 11 represents a pure-1 quadrant, 00 represents a pure-0 quadrant and 01 represents a mixed quadrant. To simplify, we use 1 instead of 11 for pure-1, 0 for pure-0, and m for mixed. The PM-tree of the previous example is also given in Figure 2.

2.3 Ptree Algebra

Different operations can be applied on Ptrees. Ptree algebra contains operators COMPLEMENT, AND, OR, and XOR, which are the pixel-by-pixel logical operations on Ptrees. The COMPLEMENT operation is a straightforward translation of each count to its quadrant-complement (e.g., a 5 count for a quadrant of 16 pixels has complement of 11). AND operation is the most frequently used operation for mining on images. Figure 3 gives an example of PM-tree ANDing. OR and XOR operations are performed in a similar way.

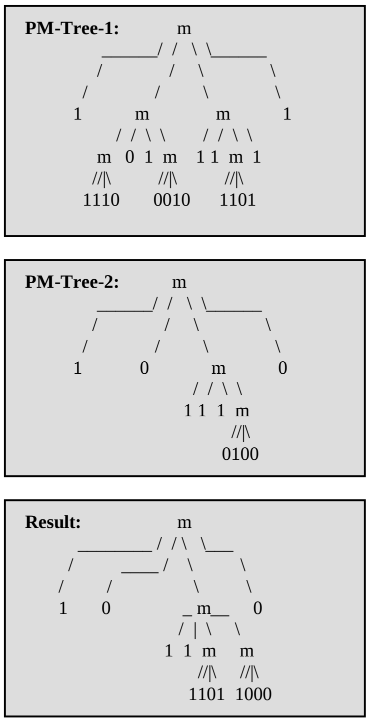

Figure 3. PM-tree ANDing

2.4 Value Ptree and Tuple Ptree

The basic Ptrees can be combined using simple logical operations (AND, OR, COMPLEMENT) to produce Ptrees for any number of bit values (any level of precision, 1-bit precision, 2-bit precision, etc.). We let $P_{b,v}$ denote the Ptree for band, b, and value, v, where v can be expressed in 1-bit, 2-bit,.., or 8-bit precision. $P_{b,v}$ is called a value-Ptree. Through the same kinds of AND/Complement operations, value-Ptrees can be combined to construct tuple Ptree, which will contain the hit counts of an entire tuple for all quadrants. Figure 4 shows the relationships among Basic, Value and Tuple Ptrees, where ' indicates the COMPLEMENT operation.

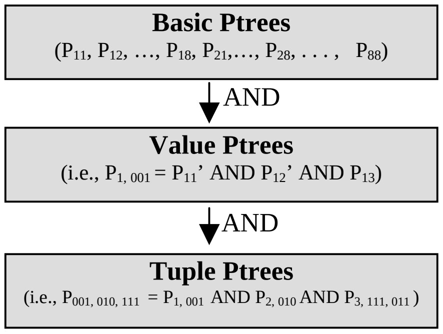
Basic Ptrees
(P_{11}, P_{12}, …, P_{18}, P_{21},…, P_{28}, . . . , P_{88})
AND
Value Ptrees
(i.e., $P_{1,001} = P_{11}$' AND P_{12}' AND P_{13})
AND
Tuple Ptrees
(i.e., $P_{001,010,111} = P_{1,001}$ AND $P_{2,010}$ AND $P_{3,111,011}$)

Figure 4. Basic, Value and Tuple Ptrees

3. Data Mining on RSI data using Ptrees

3.1 P-ARM Algorithm – Association Rule Mining on RSI data using Ptrees

The formal definition of association rules is introduced in [8]. Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of literals, called items. Let D be a set of transactions, where each transaction T is a set of items (called "itemset") such that $T \subseteq I$. We say that a transaction T contains X, a set of items in I, if $X \subseteq T$. An association rule is an implication of the form X => Y, where $X \subset I$, $Y \subset I$, and $X \cap Y = \emptyset$. The rule X => Y holds in the transaction set D with confidence c if c% of transactions in D that contain X also contain Y. The rule X => Y has support s in the transaction set D if s% of transactions in D contain $X \cup Y$.

Given a set of transactions D, the problem of mining association rules is to generate all association rules that have certain user-specified minimum support (called minsup) and confidence (called minconf).

The discovery of association rules is usually performed in two steps. The first step is to find all itemsets whose support is greater than the user-specified minimum support. Itemsets with minimum support are called frequent itemsets. The second step is to generate the desired rules using the frequent itemsets generated in the first step. The overall performance is mainly determined by the first step. Once the frequent itemsets have been generated, it's straightforward to derive the rules.

Basic association rule mining algorithms are proposed for dealing with Boolean attributes, such as Market Basket data. To perform association rule mining on RSI data, data partition is required since RSI data are quantitative data [10]. There are various kinds of partition approaches [7], including Equi-length partition, Equi-depth partition and user-customized partition.

As we mentioned, frequent itemsets generation is the key step in association rule mining. Usually a step-wise procedure is used to generate frequent itemsets [8,9]. To determine if a candidate itemset is frequent, the support is calculated then compared to the threshold. In Apriori and most other ARM algorithms, the entire transaction database needs to be scanned to calculate the support for each candidate itemset. When the transaction set is large, (e.g., a large image with 40,000,000 pixels), this cost will be extremely high.

We propose a new algorithm, P-ARM, to solve this problem. The main idea is that support of each candidate itemset can be obtained directly from ANDing Ptrees (i.e., the support count is just the root count of the result). There is no need to scan the transaction database, which is the main cost for standard ARM methods.

In P-ARM, each pixel is a transaction. After performing data partition, items are all the intervals in all the bands. An itemset is in the form of $Int_1 \times Int_2 \times \ldots \times Int_n = \Pi_{i=1..n} Int_i$, where Int_i is an interval of values in $Band_i$ (some of which may be the full value range 0~255). A 1-itemset is an itemset with (n-1) full value range intervals, while a 2-itemset is an itemset with (n-2) full value range intervals, where n is the total number of bands.

```
Procedure P-ARM
{
    Data Partition;
    $F_1$ = {frequent 1-Itemsets};
    For (k=2; $F_{k-1} \neq \emptyset$) do begin
        $C_k$ = p-gen($F_{k-1}$);
        Forall candidate Itemsets $c \in C_k$ do
            c.count = AND_rootcount(c);
        $F_k = \{c \in C_k \mid$ c.count >= minsup$\}$
        end
    Answer = $\cup_k F_k$
}
```

Figure 5. P-ARM Algorithm

The P-ARM algorithm is given in Figure 5. The *p-gen* function in P-ARM differs from the apriori-gen function in Apriori ([9]) in the way pruning is done. Since any itemsets consisting of two or more intervals from the same band will have zero support (no value can be in both intervals simultaneously), the kind of joining done in [9] is unnecessary. The *AND_rootcount* function is used to calculate itemset counts directly by ANDing the appropriate basic-Ptrees. For example, in the itemset {B1[0,64), B2[64,127)}, where B1 and B2 are two bands, the support count is the root count of $P_{1,00}$ AND $P_{2,01}$.

P-ARM is applicable equally to any kind of data partition. Since whether partitioning is equi-length, equi-depth or user-customized, it can be characterized as follows. For each band, choose partition-points, $v_0 = 0$, v_1, ..., $v_{n+1} = 256$, then the partitions are, $\{ [v_i, v_{i+1}) : i = 0..n\}$ and are identified as values, $\{ v_i : i = 0..n\}$. The items to be used in the data mining algorithms are then pairs, (b_i, v_j), where b_i is band i and v_j is partition point value.

3.2 P-Classifier – Classification on RSI data Using Ptrees

Classification is another basic data mining task. The aim of the classification task is to discover some kind of relationship between the goal attribute and other attributes, so that the discovered knowledge can be used to predict the class label of a new, unknown-class tuple.

A classification task typically involves three phases, a learning phase, a testing phase and an application phase. In the learning phase, training data are analyzed by a classification algorithm. Each tuple in a training dataset is a training sample, randomly selected from the sample population. A class label attribute is identified, whose values are used to label the classes. The learning model or classifier resulting from this learning phase, may be in the form of classification rules or a decision tree or a mathematical formulae. Since the class label of each training sample is provided, this approach is known as supervised learning. In unsupervised learning (clustering), the class labels are not known in advance. In the testing phase test data are used to assess the accuracy of classifier. If the classifier is accurate enough, it can be used to predict the class label for a new data tuple, which is the application phase.

In this paper, we consider the classification of RSI data in which the resulting classifier is a decision tree, a commonly used format for discovered classification knowledge.

Classification can be performed within one single image or a series of images for the same location. The mining process is quite similar except the partition between training data set and testing data set is different. For the single-image classification, some pixels in the image form the training set and the rest pixels form the testing set. For the multi-image classification, we can choose the images in the consecutive two years as training set and testing set. In both cases, the new data samples are remotely sensed image values taken during a "current" growing season prior to harvest. The goal is to classify the previous year's data using yield as the class label attribute and then to use the resulting classifier to predict yield levels for the current year (e.g., to determine where additional nitrogen should be applied to raise yield levels).

Our explanation is based on basic ID3 algorithm, however, Ptrees can be applied to other classification algorithms in a very similar way. The basic algorithm for inducing a decision tree from the learning or training sample set is as follows ([2] [7]):

- Initially the decision tree is a single node representing the entire training set.
- If all samples are in the same class, this node becomes a leaf and is labeled with that class label.
- Otherwise, an entropy-based measure, "information gain", is used as a heuristic for selecting the attribute (the “decision attribute”), which best separates the samples into individual classes.
- A branch is created for each value of the test attribute and samples are partitioned accordingly.
- The algorithm advances recursively to form the decision tree for the sub-sample set at each partition. Once an attribute has been used, it is not considered in descendent nodes.
- The algorithm stops when all samples are of the same class or there are no remaining attributes.

The attribute selected at each decision tree level is the one with the highest information gain. The information gain of an attribute is computed as follows. Let S be a set of data samples in the learning dataset and let s by its cardinality. Let the class label attribute have m values or classes, C_i, i=1..m. Let s_i be number of samples from S in class C_i. The expected information needed to classify a given sample is computed as follows: $I(s_1..s_m) = -\sum_{i=1..m} p_i*\log_2 p_i$, where $p_i = s_i/s$ (the probability that a sample belongs to C_i).

Let attribute, A, have v distinct values, $\{a_1..a_v\}$. Attribute A could be used to classify S into $\{S_1..S_v\}$, where S_j is the set of samples having value, a_j. Let s_{ij} be the number of samples of class, C_i, in a subset, S_j.

The entropy or expected information based on the partition by A is $E(A) = \sum_{j=1..v} \sum_{i=1..m} (s_{ij} / s) * I(s_{1j}..s_{mj})$. The information gained by using Attribute A as a decision attribute is $gain(A) = I(s_1..s_m) - E(A)$.

We propose a new classifier, called P-classifier, to construct decision tree quickly by using the count information recorded in Ptrees. To calculate the information gain, s_{ij} is calculated for each i, j. Using Ptree structure, we can provide a fast way of calculating s_{ij}.

Suppose B1 (Band 1) is the class attribute. Start with A = B2 (Band 2) to classify S into $\{A_1..A_v\}$, where $A_j = \{t| t(B2)=a_j\}$ and a_j ranges over those B2-values, v', such that the root count of $P_{2,v'}$ is non-zero. The symbol, s_{ij}, counts the number of samples of class, C_i, in subset, A_j, that is, the root count of $P_{1,v}$ AND $P_{2,v'}$, where v ranges over those B1-values such that the root count of $P_{1,v}$ is non-zero.

It is unnecessary to rescan the learning set to form these sub-sample sets, since the Ptrees for those samples have been computed. So, P-Classifier saves the cost of sub-sample set creation.

Any attribute, which has one single value in each candidate decision attribute over the entire sample, need not be considered, since the information gain will be zero. If all candidate decision attributes are of this type, the algorithm stops for this subsample.

3.3 Other applications of Ptrees to RSI data mining

Ptrees can be successfully applied to clustering as well as other classification techniques, including Bayesian Classification and k-Nearest Neighbor classification. We won't be able to detail these algorithms due to length limitations on this paper.

4. Performance Analysis

4.1 Performance Analysis on P-ARM Algorithm

For association rule mining, we compare our work with the classical frequent itemsets generation algorithm, Apriori [9], and a recently proposed efficient algorithm, FP-growth [12], in which no candidate generation step is needed. The experiments are performed on a 900-MHz PC with 256 megabytes main memory, running Windows 2000. We generalized our algorithm to find all the frequent itemsets, not limited to those of-interest (e.g., containing Yield) for the fairness, so that the association rules we got using Apriori, FP-growth and P-ARM algorithms are completely identical. The images we used are actual aerial TIFF images with a synchronized yield band and can be found at [14]. So, each dataset has 4 bands {Blue, Green, Red, Yield}. We use different image sizes up to 1320×1320 pixels (the total number of transactions will be ~ 1,700,000). We only store the basic Ptrees for each dataset. In this paper, we gave the performance results based on one typical TIFF-Yield dataset, and the performance on other TIFF-Yield datasets are quite similar as we tested.

4.1.1 Comparison of the P-ARM Algorithm with Apriori algorithm

We implemented the Apriori algorithm [9] for the TIFF-Yield datasets using Equi-length partitioning. P-ARM is more scalable than Apriori in two ways. First, P-ARM is more scalable for lower support thresholds. The reason is, for low support thresholds, the number of candidate itemsets will be extremely large. Thus candidate itemset generation performance degrades markedly. Figure 6 gives the results of the comparison of the P-ARM algorithm using Ptree and Apriori for different support thresholds.

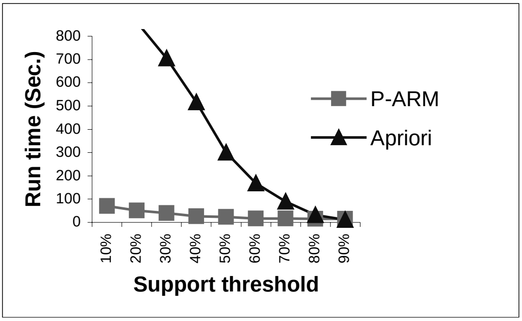

Figure 6. Scalablity with support threshold

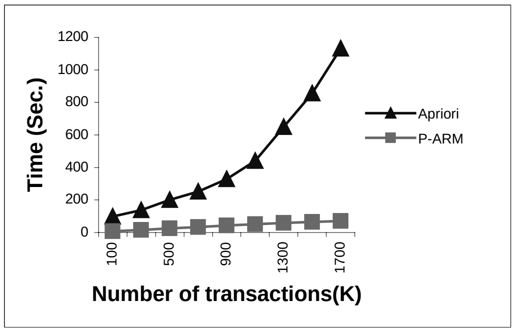

Figure 7. Scalability with number of transaction

Secondly, P-ARM algorithm is more scalable to large image datasets. The reason is, in the Apriori algorithm we need to scan the entire database each time a support is to be calculated. This is a very high cost for large databases. However, in P-ARM, since we calculate the count directly from the root count of a basic-Ptree AND program, when we double the dataset size, only one more level is added to each basic-Ptree. The cost is relatively small compared to the Apriori algorithm as shown in figure 7.

4.1.2 Comparison of the P-ARM algorithm and the FP-growth algorithm

FP-growth is a very efficient algorithm for association rule mining, using a data structure called frequent pattern tree (FP-tree) to store compressed information about frequent patterns. We use the FP-growth object code and convert the image to the required file format. For a dataset of 100K bytes, FP-growth runs very fast. But when we run the FP-growth algorithm on the TIFF image of size 1320×1320 pixels, the performance falls off. For large sized datasets and low support thresholds, it takes longer for FP-growth to run than P-ARM. Figure 8 shows the experimental result of running the P-ARM and the FP-growth algorithms on a 1320×1320 pixel TIFF dataset. In these experiments we have used 2-bits precision.

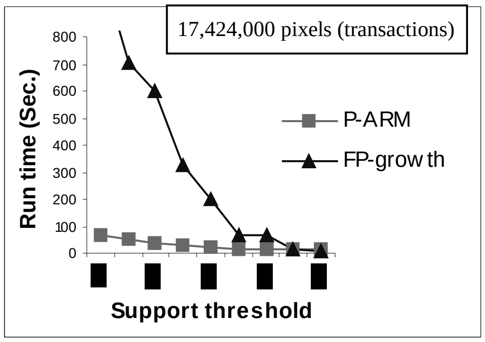

Figure 8. Scalability with support threshold

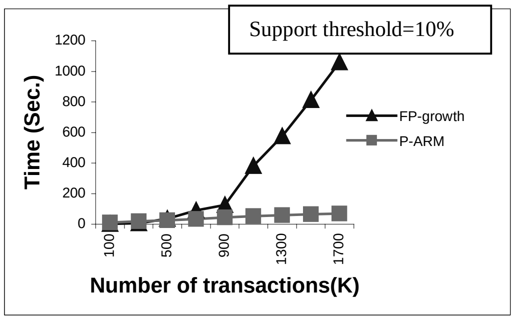

Figure 9. Scalability with the number of transactions

Both P-ARM and FP-growth run faster than Apriori algorithm. For large image datasets, the P-ARM algorithm runs faster than FP-growth algorithm when the support threshold is low. Also, the relative performance of P-ARM (relative to FP-growth) increases as the size of the data set increases (Figure 9).

4.2 Performance Analysis on P-Classifier

Usually prediction accuracy is used as a basis of comparison for the different classification methods. However in P-Classifier, we use ID3 with new data structures, which are intended to improve the speed of the algorithm, not the predictive accuracy. Therefore the performance issue here is classification time. We got identical decision trees by using ID3 and P-Classifier.

In P-Classifier, we only build and store basic Ptrees. All the ANDings are performed on the fly when the corresponding root count is needed. Our experimental results show that in P-Classifier the classification time decreases significantly than basic ID3, especially for large data size (in Figure 10).

The reason lies in several aspects. First, in ID3, to test if all the samples are in the same class, one scan on the entire sample set is needed. While in P-Classifier, we only need to check if the corresponding quadrant is pure-1. Second, in P-Classifier, the creation of sub-sample sets is not necessary when all the samples are not in the same class. For example, if the Ptree of the current sample set is $P_{2,0100} \wedge P_{3,0001}$, and the current attribute is B1 (with, say, 2 bit values), then $P_{2,0100} \wedge P_{3,0001} \wedge P_{1,00}$, $P_{2,0100} \wedge P_{3,0001} \wedge P_{1,01}$, $P_{2,0100} \wedge P_{3,0001} \wedge P_{1,10}$ and $P_{2,0100} \wedge P_{3,0001} \wedge P_{1,11}$ identifies the partition of the current sample set. To generate S_{ij}, only Ptree ANDings are required.

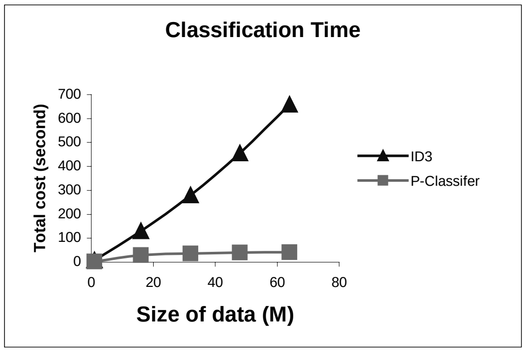

Figure 10. Classification cost with respect to the dataset size

The additional cost of building and storing the basic Ptrees are very small. For high order bit of bands in TIFF image (1320×1320), the average size of basic Ptree is 22K Bytes, while for other bits, the average size is about 200K Bytes. The time to build the basic Ptrees is about 30ms.

5. Related work

The P-tree structure is related to Quadtrees [15] and its variants (such as point quadtree [15] and region quadtree [15]), and HHcodes [16]. The similarities among P-trees, quadtrees and HHCodes are that they are quadrant based. The difference is that P-trees focus on counts. P-trees are not only beneficial for storing data, but also for fast data mining, since they contain useful needed information for data mining.

Wavelets provide a good way for compressing data, but unlike Ptree, they do not facilitate pixel by pixel mining.

6. Conclusion and Future Work

In this paper, we propose a new model of performing various data mining approaches, such as association rule mining and classification, on remotely sensed imagery data. We use a new data organization, bit Sequential organization (bSQ) and a lossless, data-mining ready data structure, the Peano Count tree (Ptree), to represent the information needed for data mining in an efficient and ready-to-use form. The rich and efficient Ptree storage structure and fast Ptree algebra facilitate fast mining algorithms, such as P-ARM and P-Classifier proposed in this paper. Experiments show that the new model improves performance obviously than existing approaches on mining large sized images.

In this paper, Ptree structure provides a fast way to calculate some measurements for mining task, such as support and confidence in association rule mining task and information gain in classification task. Similarly, Ptree can facilitate the fast calculation for other measurements, such as interest and conviction[13] in association rule mining and Gini index[4] in classification.

Two types of classification can be performed for image data. One is the classification on one individual image. The other is the classification on a series of related images, such as time-series images in the same location. Our future work includes incremental classification on time series image data using Ptrees.

Acknowledgements

We would like to express our thanks to Dr. Jiawei Han for providing us the FP-growth object code.

References

[1] J.R. Quinlan, R.L. Riverst, “Inferring decision trees using minimum description length principle”, Information and Computation, 80, 227-248, 1989.
[2] Quinlan, J. R., “C4.5: Programs for Machine Learning”, Morgan Kaufmann, 1993.
[3] R. Agrawal, S. et al, “An interval classifier for database mining applications”, VLDB 1992.
[4] L. Breiman, et al, “Classfication and Regression Trees”, Wadsworth, Belmont, 1984.
[5] J. Shafer, R. Agrawal, M. Mehta, “SPRINT: A scalable parallel classifier for data mining”, VLDB 96.
[6] J. Gehrke, et all, “BOAT: Optimistic Decision Tree Construction, SIGMOD 99.
[7] Jiawei Han, Micheline Kamber, “Data Mining: Concepts and Techniques”, Morgan Kaufmann, 2001.
[8] R. Agrawal, T. Imielinski, A. Swami, “Mining Association Rules in Large Database”, SIGMOD 93.
[9] R. Agrawal and R. Srikant, “Fast Algorithms for Mining Association Rules,” VLDB 94.
[10] R.Srikant, R.Agrawal, "Mining Quantitative Association Rules in Large Tables", SIGMOD 96.
[11] J. S. Park, M.S. Chen and P. S.Yu, “An effective Hash-Based Algorithm for Mining Association Rules,” SIGMOD 95.
[12] J. Han, J. Pei and Y. Yin, “Mining Frequent Patterns without Candidate Generation”, SIGMOD 2000.
[13] S. Brin et al, “Dynamic Itemset Counting and Implication Rules for Market Basket Data”, SIGMOD 97.
[14] SMILEY Project. Available at http://midas.cs.ndsu.nodak.edu/~smiley
[15] H. Samet, “The quadtree and related hierarchical data structure”. ACM Computing Survey, 16, 2, 1984.
[16] HH-code. Available at http://www.statkart.no/nlhdb/iveher/hhtext.htm