Discretization of Multidimensional Web Data for Informative Dense Regions Discovery

Edmond H. Wu[†], Michael K. Ng[†], Andy M. Yip[‡], and Tony F. Chan[‡]

[†]Department of Mathematics, The University of Hong Kong
Pokfulam Road, Hong Kong.
[†]hcwu@hkusua.hku.hk,mng@maths.hku.hk
[‡]Department of Mathematics, University of California,
Los Angeles, CA 90095-1555, USA.
[‡]mhyip@math.ucla.edu,chan@math.ucla.edu

Abstract. Dense regions discovery is an important knowledge discovery process for finding distinct and meaningful patterns from given data. The challenge in dense regions discovery is how to find informative patterns from various types of data stored in structured or unstructured databases, such as mining user patterns from Web data. Therefore, novel approaches are needed to integrate and manage these multi-type data repositories to support new generation information management systems. In this paper, we first discuss and purpose several discretization methods which are suitable for multidimensional Web data. Based on it, we demonstrate some dense regions discovery applications by using Web usage data from a real Website. The experiments show that the discretization methods are quite effective and efficient, especially for high-dimensional data. It also suggests that the discretization methods can be used in other practical Web applications, such as user patterns discovery.

Keywords
Discretization, Dense regions discovery, Web mining, Web information system

1 Introduction

With the fast growing of Internet in recent years, developing Web-based information systems become an active research area of finding useful information from Web data depending on state-of-art information technologies nowadays. The rapid progress of our capabilities in data acquisition and storage technology has led to the fast growing of tremendous amount of data generated and stored in databases, data warehouses, or other kinds of data repositories such as the World-Wide Web. As results, there are many types of usable Web data in reality, such as Web content and structure data, Web log data and hypertext data. Much of the Web data is relevant in nature, involving multiple objects. Hence, novel approaches are needed to integrate and manage these multi-type Web data repositories to support new generation information management systems.

As a pre-processing step in knowledge discovery, discretization is defined as a process that divides continuous numeric values into a set of intervals that can be regarded as discrete categorical values [4]. In [2], Dougherty suggested three different axis to classify discretization methods: supervised vs. unsupervised, global vs. local and static vs. dynamic. Supervised methods use the information of class labels while unsupervised methods do not. Global methods are applied before the learning process while local methods produce partitions that are applied to localized regions of the instance space. The difference between static and dynamic methods is that in static methods, attributes are discretized independently of each other, while dynamic methods take into account the interdependencies among the attributes.

Many researchers have purposed various algorithms for data discretization from different points of view. The well known and simplest discretization method is Equal Interval Width, which divides the range of n ranking observations for a variable into k equal sized bins. The main disvantage is that the performance of this method may be affected by outliers. Another method, Equal Frequency Intervals, divides n observations of a continuous variable into k bins where each bin contains n/k adjacent values. An improved method called Maximal marginal entropy was also suggested. It also starts with equal frequency intervals, but adjusts the boundaries to decrease the entropy of each interval instead of keeping equal frequency.

Statistical methodology is also adopted in some discretization methods. The ChiMerge system [6] provided a statistical heuristic method for discretization. It begins by placing each observed real value into its own interval and proceeds by using the χ^2 test to determine when adjacent intervals should be merged. The StatDisc [7] method also uses statistical tests as a means of determining discretization intervals. This method follows a bottom-up manner and creates a hierarchy of discretization intervals using the ϕ measure as a criterion for merging intervals. The merging process continues until some ϕ threshold is achieved. StatDisc can merge N adjacent intervals while ChiMerge can only merge two adjacent intervals at a time. After that, a suitable final discretization is automatically selected.

The 1R system, purposed by Holte [5], attempts to divide the domain of every possible continuous variable into pure bins, each containing a strong majority of one particular class. This method works well, particularly when used in conjunction with the 1R induction algorithm.

In [1], Catlett suggested the use of entropy based discretization in decision tree algorithms. The experiments demonstrated an impressive increase in induction speed on very large datasets. Basd on this, Fayyad and Irani [3] employed a recursive entropy minimization heuristic for discretization. To control the number of intervals produced over the continuous space, they used a Minimum Description Length criterion. This method was applied locally at each node during tree generation. Other authors also used the method in global discretization and obtained satisfying results [2].

Another discretization method ReliefF, suggested by Robnik-Sikonja and Kononenko [8], applied the ReliefF algorithm in discretization. This method estimates the probability that two near examples of the same class have the same value for an attribute and that two near examples from different classes have the same value for the attribute. The ReliefF is a heuristic measure to decide which of two discretizations is better.

Although discretization is well studied in literature, effective and efficient discretization on high-dimensional and fast-growing Web data is a new problem because much of the data are distributed on different Web sources.

The remaining of this paper is organized as follows. In section 2, we introduce concept and method in dense region discovery and then address the discretization problems encountered in dense region discovery. Section 3 presents our suggested solutions to discretization of high-dimensional complex Web data in detail. In section 4, we perform an empirical evaluation of the method using different real Web datasets. Finally, we give some conclusions and our future work.

2 Dense Regions Discovery

2.1 Definition of Dense Regions

We now fix some notations and give the definition of dense regions. Given an n-by-p data matrix X where n is the number of entities and p is the number of attributes of each entity. Let R and C be an index set of a subset of rows and columns of X respectively. Since we do not distinguish between a matrix and its permuted versions, we assume that R and C are sorted in the ascending manner. A submatrix of X formed by its rows R and columns C is denoted by $X(R, C)$. We also identify $X(R, C)$ by the index set $D = R \times C$. For example, let $R = \{3\}$ and $C = \{1, 2\}$, then $R \times C = X(R, C) = (x_{31} x_{32})$.

Definition 1 (Dense regions). *A submatrix $X(R, C)$ is called a maximal dense region with respect to v, or simply a dense region with respect to v, if*

- *$X(R, C)$ is a constant matrix whose entries are v (density), and,*
- *Any proper superset of $X(R, C)$ is a non-constant matrix is non-constant (maximality).*

Example 1. Let X be a data matrix given by the first matrix below. Then, the dense regions of X with respect to 1 are given by the four matrices in the brace.

$$\begin{pmatrix} 1&0&0&1 \\ 1&1&0&1 \\ 1&1&0&1 \\ 1&1&2&0 \end{pmatrix}; \quad \left\{ \begin{pmatrix} 1&*&*&* \\ 1&*&*&* \\ 1&*&*&* \\ 1&*&*&* \end{pmatrix}, \quad \begin{pmatrix} 1&*&*&1 \\ 1&*&*&1 \\ 1&*&*&1 \\ *&*&*&* \end{pmatrix}, \quad \begin{pmatrix} *&*&*&* \\ 1&1&*&1 \\ 1&1&*&1 \\ *&*&*&* \end{pmatrix}, \quad \begin{pmatrix} *&*&*&* \\ 1&1&*&* \\ 1&1&*&* \\ 1&1&*&* \end{pmatrix} \right\}.$$

Alternatively, we may denote the above dense regions by $\{1, 2, 3, 4\} \times \{1\}$, $\{1, 2, 3\} \times \{1, 4\}$, $\{2, 3\} \times \{1, 2, 4\}$ and $\{2, 3, 4\} \times \{1, 2\}$ respectively.

We note that there exist some data matrices which possess exponentially many dense regions with respect to a certain value. In the next sections, we discuss how to identify only a subset of them which are of practical interest.

2.2 Algorithm for Mining Dense Regions

Yang et al [13] suggest an algorithm for finding error-tolerant frequent itemsets from high-dimensional data. The investigating problem of the paper is similar but not identical to dense regions discovery in which focuses on mining association rules. These frequent itemsets found can be regarded as dense regions from a sparse and large matrix(e.g., customer transaction databases).

In [15], we present the algorithm for mining dense regions in large data matrices. The essence of the algorithm is that it searches the data matrix for dense regions in both vertically and horizontal directions beginning from a given starting point (s, t). It returns two dense regions containing (s, t), one is obtained by searching in vertical direction first; the other is by searching in horizontal direction. The advantage of this algorithm is that a lot of computations can be saved by searching only the entries with possible qualifying dense regions and thus reducing the searching space. Therefore, it is an effective and efficient algorithm for dense regions discovery from high-dimensional data matrices. Due to the limited length of the paper, we just employ the dense regions discovery algorithm in experiment part and omit the detailed introduction of it.

2.3 Problem Statement and Motivation

In this section, we will discuss problems addressing on discretization. There are four main concerns why discretization is important to dense regions discovery. The first one is related to the dense regions discovery algorithm. From section 2, we know that in the algorithm of finding dense regions, the heuristic is to find sub-matrices with respect to value v. So, the original design of DRIFT algorithm is for integer or categorical values. Hence, we need to perform discretization on attributes of continuous values first before the dense regions discovery process. This is the reason why we need to do discretization.

The other concern is about the effects of discretization to dense regions discovery. Because the informative dense regions are not only subsets of entities and attributes whose corresponding values are almost constant but also patterns which are of practical use and are significant, so discretization is not a trivial for dense regions discovery since the discretization can greatly determine the dense regions found.

The third is that we may think about the computational complexity of performing attribute discretization. What's more, the quality of discretization will also greatly effect the interpretation of dense regions found. In the design of our discretization methods, we can consider the interdependence among attributes and distribution of data for effective discretization for discretization.

The fourth consideration is how many intervals is proper and how to distribute an observed value to an interval. From the introduction part, we know

that there are several approaches: Some methods fix the number of intervals in advance and some methods permit automatically setting of intervals k. For instance, k can be computed based on the number of distinct values observed on the training set, such as $k = max(1, 2 * logl))$[2]. The advantage of this method is that given a dataset with several continuous features, the number of intervals of each feature depends on the number of different values observed on the training set. Therefore, different features can be discretized with different number of intervals.

3 Discretization Model for Dense Regions Discovery

In this section, we present several methods which are particullysuitable to discretize data matrices.

3.1 Equal Width Interval

Equal width interval method is to discretize data into k equally sized bins, where k is a parameter provided by users. Given a variable v, the upper bound and lower bound of v are v_{max} and v_{min}, respectively. Then, this method use formula below to compute the size of bin:

$$\lambda = \frac{v_{max} - v_{min}}{k}$$

Hence, for i_{th} bin, the bin boundary is $[b_{i-1}, b_i]$, where $b_i = b_0 + i\lambda$, $b_0 = v_{min}, i = 1, ..., k-1$.

Equal width interval method can be applied to each feature indendently. Since this method doesn't class information, thus it is an unsupervised discretization method.

In the problem of dense regions discovery, given a data matrix X, v_{ij} represents the value of entry x_{ij}, so we can also apply Equal width interval method to discretize X. In such cases, we define k=$max\{1, \lceil In(mn) \rceil\}$ as the number of intervals of $m \times n$ matrix X. If $m = n$, k=$max\{1, \lceil 2In(n) \rceil\}$.

3.2 Supervised Adaptive Width Interval

As we have mentioned before, there are two main drawbacks of equal width interval method. One is that it may be influenced by outliers, the second one is that it does not use the class information. These two problems are correlated in nature. Suppose the attributes for discretization are numerical variables, if there exists several classes in the dataset, then it is very likely that the mean and variance of these classes are differnt. Therefore, using this information, we may get more informative discretization results for interesting patterns discovery. This is our motivation of using supervised equal width interval in our discretization tasks.

Suppose there are r classes labled $C_1, ..., C_r$ in the dataset. As to variable v, the means of v in each class of the dataset are $\overline{v}^1, ..., \overline{v}^r$, respectively. The

maxmial value and minimal value in class C_i are v^i_{max} and v^i_{min}. Then, our idea Supervised adaptive width interval method is to partition $v^i_{max} - \overline{v}^i$ and $\overline{v}^i - v^i_{min}$ into k parts, individually. In the end, we can divide all the data into $2k$ bins.

$$\lambda^i_u = \frac{v^i_{max} - \overline{v}^i}{k}$$

$$\lambda^i_l = \frac{\overline{v}^i - v^i_{min}}{k}$$

$$b^i_j \begin{cases} 0 \le j < k, & \overline{v}^i - \lambda^i_{lj} j \\ j = k, & \overline{v}^i \\ k < j \le 2k, & \overline{v}^i + \lambda^i_{u(j-k)}(j-k) \end{cases}$$

This method is different with Equal Width Interval method. For example, given a dataset with continusous attribute v while $v_{max} = 100$, $v_{min} = 0$ and $\overline{v} = 60$, if we want to partition the data into 4 parts. Then, for Equal Width Interval method, the partition is $\{0, 25, 50, 75, 100\}$, while the partition of Adaptive Width Interval is $\{0, 30, 60, 80, 100\}$.

A more general form of Supervised Adaptive Width Interval discretization is as follow:

$$\lambda^i_{uj} = \rho_{uj}(\hat{v^i} - v^*)$$

$$\lambda^i_{lj} = \rho_{lj}(v^* - \check{v^i})$$

where $0 < \rho_j < \rho_{j+1} \le 1$, $\check{v^i} < v^i < \hat{v^i}$, $j = 1, ..., k-1$.

In the formula, v^* is self-selecting value in the data value range between v_{max} and v_{min}. For instance, besides $\overline{v}$, user can also select median as v^*. As to ρ_{lj}, it provides the flexibility for user to select suitable length of interval for discretization. The setting of these parameters depends on the datasets and mining tasks.

Generally, we can use this formula to do discretization of data in certain value. We can also adopt this approach for iterated discretization. For example, we can first partition the data into two parts and perform supervised adaptive width interval discretizationby by a certain value v^*. Then, using v^* as v_{min} in the right part and v^* as v_{max} as left part, we can also select other values as v^* to do discretization on any subset of the data. This process proceed until certain criteria satisfies, e.g., number of bins.

The merits of Supervised Adaptive Width Interval method is to overcome the drawbacks of Equal Width Interval method. In some cases, the Supervised Adaptive Width Interval is more useful to inditify potential patterns, we will show this through our Web application and example.

3.3 Mixed Equal Frequency Interval and K-means Discretization

Compare with Equal Width Interval, the main advantage of Equal Frequency Interval is that the equal frequency discretization produces even amount of instance into n/k bins based on the ranking values, so it can effectively avoid the problems of outlier and the influence of data means or variance. In many analytical tasks, analysts prefer to user equal frequency because it seems easier to use and interpret, e.g., analyse the user patterns of top 5% Web pages being frequently visited. However, this method ignore the correlation in the data. As results, using Equal Frequency Interval discretization may lose some useful data information.

Recall a well known approach in clustering is k-means, the essence of the method is to minimizes the intra-interval distance and maximizes the inter-interval distance. It begins with an equal width distribution of the observed values over the k intervals, followed by an iterative process where the values near the boundaries change between intervals while this process improves certain criterion until no more imprvement can be achieved. Motivated by k-means, we combine it with equal frequency interval for discretization.

The idea is that we first use equal frequency interval to discretize the data into k bins, then we use k-means to adjust the values in each bin. We refine the objective function to minimizes intra-interval data variance and maxmize the inter-interval data variance with some constrain on the difference of frequency on each bin. Combining this two methods, the discretization results can better represet the underlying data patterns. The algorithm of the purposed method is summarized as follows:

Begin
1. Import dataset D with n instance in numerical variable v
2. Sort D into $D' = d_1, ..., d_n$ by v and set number of interval k
3. For each d_i, i=1...,n. Put d_i into bin $b_{\lfloor i/k \rfloor}$
4. Put all n instance into k bins until each bin contains n/k instance
5. Calculate the mean and variance of each bin and the variance among bins
6. Set the number of movable values and maximal difference of instances in bins
7. For each value around the interval boundaries, move it to neighbour intervals
8. Recompute the mean and variance of each bin and the variance among bins
9. If the variance of means among bins is larger and the variance in bins is smaller
10. Save the new partition of intervals, otherwise, go to 7
11. Output the discretization with k adjusted bin-size intervals for D

End

3.4 Two-Way Discretization

In dense regions discovery, the most important issue is to find subsets from data matrix in which have potentially useful patterns. However, matrices with continuous values are hard to apply dense regions discovery algorithms since most dense regions discovery algorithms assume the values of entries are discrete. Therefore, we need to discretize matrices with continuous values into discrete or categorical entries first. For an entry x_{ij} in matrix X, it belongs to i_{th} row and j_{th} column. If X is a $m \times n$ matrix, then X is a two-dimensional matrix with m and n attributes on each dimension. Specially, if each dimension represents

one variable, then we can regard the value v_{ij} of x_{ij} as a data point in i_{th} row and j_{th} column, individually. For example, in Table 4, v_{ij} denotes the frequency of visiting from page i to page j. The characteristic of such matrix is that the rows and columns represeting the same attribute,e.g., frequency. Thus, how to discretize such matrix become an interesting research problem.

For efficient dense regions discovery, our idea is to discretize the continuous value of x_{ij} into two discrete value by vertical and horizonal searching. It is so called Two-Way discretization. The major steps for discretization of X as follows:

Begin
1. Import $m \times n$ matrix X with continuous variable v
2. Sort each row R_i and column C_j by v respectively
3. Set the criteria to discretize every row R_i and column C_j, e.g., equal frequency interval
4. Determine number of intervals k and interval boundaries for R_i and C_j
5. For each x_{ij}, $i = 1..., m, j = 1, ...n$. Discretize x_{ij} by R_i and C_j, store in vr_{ij} and vc_{ij}
6. Output the two-way discretization matrix

End

3.5 Examples of Web Data Discretization

In this section, we first introduce several indexes for Web usage mining and then adopt our purposed methods for Web Data discretization.

Table 1 is a dataset of user Access sessions. It contains browsing behavior of three users(User 1, User and User 3) during time period T1, T2 and T3, individually. The Web pages visited by users are A, B, C, D, E, F, G and H. The left Website topology of Fig4 donotes the connection of these Web pages. The corresponding connection matrix of the Website topology lists in Table 3.

	User1	User2	User3
T1	EHCABF	ACHECABFBG	ABGBACH
T2	ADACHE	GBACH	ACHEHCABF
T3	DACHEH	ADACHEHCABFBG	ABGBACHC

Table 1. User Access Sessions

	User1	User2	User3	SAE
T1	EF	EF	GH	34%
T2	DE	GH	EF	38%
T3	DE	DEHFG	GH	48%
UAE	33%	32%	25%	

Table 2. Session Access Efficiency

Example 1. Table 4 is the access matrix (refer to Appendix 1) obtained from Table 1. Based on the visiting frequency from one page to another page, we can classify the data into several categories. In this case, we divided the accesses into three classes by frequency: High(H), Median(M), and Low(L), excluding no access entries which are labeled as None(N). The highest frequency is 9 and the lowest is 1, using Equal Frequency Interval, we obtain three bins: L={1, 2, 3},M={4, 5, 6},H={7, 8, 9}. In this case, values are discrete, however, this method is suitable for continuous values. Table 5 shows the result using Equal Width Interval(EWI) for discretization.

Example 2. Considering the difference in the connection of pages, we may get more potential information of page access. In Table 3, for entry x_{ij}, if $x_{ij} = 1$, it means that there is a hyperlink from page i to page j. Otherwise, $x_{ij} = 0$. So,

we can classify the page relationship into two kinds: one is direct access and the other is indirect access. Using this class information, we can employ supervised discretization method in this case. In this case, we divided all x_{ij} into three classes by frequency: High(H), Median(M), and Low(L) by using supervised adaptive width interval (SAWI) method. Here, $k = 2$, we merge two bins near the mean, so we get 3 classes of page frequency by different page access. The result shows in Table 6.

Page	A	B	C	D	E	F	G	H
A	0	1	1	1	0	0	0	0
B	1	0	0	0	0	1	1	0
C	1	0	0	0	0	0	0	1
D	1	0	0	0	0	0	0	0
E	0	0	0	0	0	0	0	1
F	0	1	0	0	0	0	0	0
G	0	1	0	0	0	0	0	0
H	0	0	1	0	1	0	0	0

Table 3. Original Connection

Page	A	B	C	D	E	F	G	H
A	0	7	9	2	5	4	4	8
B	3	0	3	0	0	4	5	3
C	4	4	0	0	5	4	2	8
D	3	1	3	0	3	1	1	3
E	4	5	4	0	0	4	2	4
F	0	2	0	0	0	0	2	0
G	3	3	3	0	0	0	0	3
H	4	4	6	0	5	4	1	0

Table 4. Access Matrix

Page	A	B	C	D	E	F	G	H
A	N	H	H	L	M	M	M	H
B	L	N	L	N	N	M	M	L
C	M	M	N	N	M	L	L	H
D	L	N	L	N	L	L	L	L
E	M	M	M	N	N	M	L	M
F	N	L	N	N	N	N	L	N
G	L	L	L	N	N	N	N	L
H	M	M	M	N	M	M	L	N

Table 5. EIW Discretization

Page	A	B	C	D	E	F	G	H
A	L	H	H	L	H	H	H	H
B	L	L	M	L	L	M	M	M
C	M	H	L	L	H	H	M	H
D	L	L	M	L	M	L	L	M
E	H	H	H	L	L	H	M	M
F	L	L	L	L	L	L	M	L
G	M	L	M	L	L	L	L	M
H	H	H	M	L	M	H	L	L

Table 6. SAWI Discretization

Page	A	B	C	D	E	F	G	H
A	0	1	1	1	1	0	1	0
B	1	0	0	0	0	1	0	0
C	1	0	0	0	0	1	0	0
D	1	0	0	0	0	0	0	0
E	1	0	0	0	0	1	1	0
F	0	0	0	0	1	0	0	0
G	1	0	0	0	0	0	0	1
H	0	0	0	0	0	1	1	0

Table 7. Optimal Connection

Page	T1	T2	T3	Total(m)
A	6	8	5	19
B	3	5	3	11
C	7	5	5	17
D	0	12	11	23
E	27	18	15	60
F	171	123	56	350
G	35	9	46	90
H	55	120	145	320

Table 8. Staying Time

Fig 1 and Fig 2 show the comparison of discretization results by using EWI method and SAWI method. We can see that the data distribution after discretization is quite different. Hence, we can conclude that using different discretization methods on the same data matrix, we can still get different results. As to the problem how to choose proper discretization method for mining tasks, we will discuss it in later chapter.

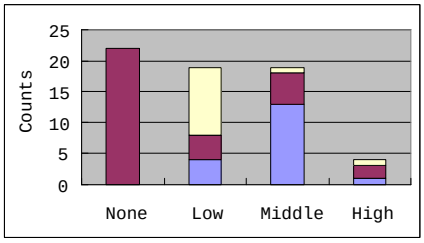

Fig. 1. EWI Discretization

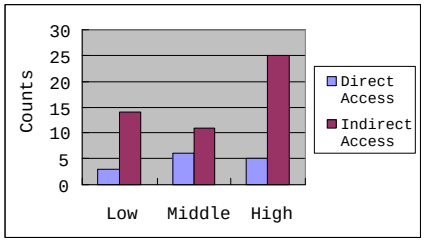

Fig. 2. SAWI Discretization

Example 3. Access Distinctness (AD) (refer to Appendix 2) is a useful index of finding potential patterns with access efficiency (refer to Appendix 2) problem from access matrix. Table 9 is the AD matrix. We can see that entries in the matrix are continuous values. We attempt to use Mixed Equal Frequency Interval and K-means method (MEK) to discretize the matrix. Here, k=$max\{1, \lceil k2In(8) \rceil\}$=5. So, we discretize the AD value into 5 classes, ranging from level 1 to level 5.

Discretization of AD matrix can help us to analyze user access patterns under the Website structure. Website analyst should pay more attention to the access patterns with higher level of AD value because these patterns are important for users, however, they can be efficiently accessed in current Website topology.

Page	A	B	C	D	E	F	G	H
A	0	3.4	3.6	2.2	5.2	4.2	4.2	4.6
B	2.6	0.0	3.6	0	0	2.8	3.0	5.0
C	2.7	3.9	0	0	3.8	5.3	4.6	3.2
D	2.6	2.8	3.6	0	5.4	3.5	3.5	4.3
E	5.0	6.6	3.6	0	0	7.1	6.4	2.2
F	0	2.2	0	0	0	0	2.8	0
G	3.9	2.6	5.0	0	0	0	0	5.7
H	3.9	5.3	2.9	0	2.4	6.0	4.6	0

Table 9. AD Matrix

Page	A	B	C	D	E	F	G	H
A	0	2	3	1	5	4	4	4
B	1	0	3	0	0	2	2	4
C	1	3	0	0	3	5	4	2
D	1	2	3	0	5	2	2	4
E	4	5	3	0	0	5	5	1
F	0	1	0	0	0	0	2	0
G	3	1	4	0	0	0	0	5
H	3	5	2	0	1	5	4	0

Table 10. MEK Discretization

Class	Range	Frequency
5	5.2-7.1	9
4	4.2-5.0	9
3	3.6-3.9	8
2	2.8-3.5	9
1	2.2-2.7	8
0	0	21
Total	0-7.1	64

Table 11. Index Statistics

Example 4. Access Interest(AI) (refer to Appendix 2) is also a very useful index of finding potential patterns with access efficiency (refer to Appendix 2) problem. The advantage of AI is that it considers the time factor. Table 12 is the AI matrix, which is calculated by AD matrix (see Table 9) and page staying time (see Table 8). In this example, we use two-way discretization method, setting $k = 2$. S denotes that this entry is significantly larger than other entries in the row or column while N denotes it is not significant. Here, we define an entry is significant if its AI value is larger 80% entries in the row or the column. Table 13 demonstrates the discretization. For example, SN of entry x_{AD}, S denotes the AI value of x_{AD} is significant in column D but not significant in row A. Two-way discretization is especially effective in our purposed dense regions discovery algorithms. What's more, it can meet the needs of different mining tasks.

Page	A	B	C	D	E	F	G	H
A	0	6.7	7.2	5.8	9.6	10.1	8.9	10.4
B	5.9	0	6.9	0	0	8.7	7.6	10.8
C	6.4	7.2	0	0	8.2	11.2	9.2	9.0
D	6.2	6.3	7.3	0	9.8	9.4	8.2	10.1
E	9.3	10.8	7.9	0	0	13.1	11.4	8.0
F	0	8.0	0	0	0	0	8.9	0
G	8.6	7.1	9.6	0	0	0	0	11.7
H	9.7	11.1	8.7	0	8.2	12.5	10.6	0

Table 12. AI Matrix

Page	A	B	C	D	E	F	G	H
A	0	NN	NN	SN	SN	NS	NN	NS
B	NN	0	NN	0	0	NS	NN	SS
C	NN	NN	0	0	NN	NS	NS	NN
D	NN	NN	NN	0	SS	NN	NN	NS
E	SN	SN	NN	0	0	SS	SS	NN
F	0	NS	0	0	0	0	NS	0
G	NN	NN	SS	0	0	0	0	SS
H	SN	SS	SS	0	NN	SS	SN	0

Table 13. Two-way Discretization

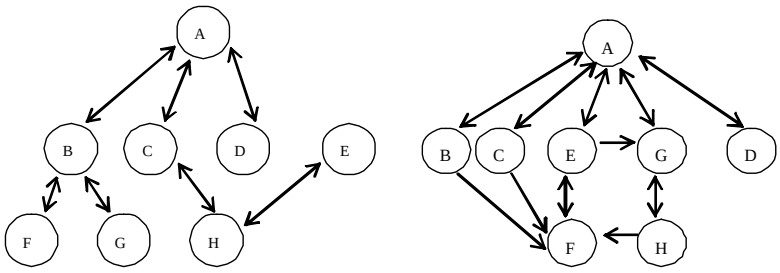

Fig. 1. Comparison of Original and Optimized Website Topology

4 Experiments

4.1 Web Datasets

ESPNSTAR Web Datasets
In this section, we use the Web-log data from ESPNSTAR.com.cn, a sports website in China, to test and validate the performance and effectiveness of our algorithm. Each dataset contains a set of access sessions during some period of time. We use two months' Web-log data to do the experiments. Table 1 lists the datasets for experiments. ES1, ES2 and ES3 are the log datasets during December, 2002 and ES4 and ES5 are the log datasets from April, 2003.

Dataset	No.Accesses	No.Sessions	No.Visitors	No.Pages
ES1	583,386	54,300	2,000	790
ES2	2,534,282	198,230	42,473	1,320
ES3	6,260,840	517,360	50,374	1,450
ES4	78,236	5,000	120	236
ES5	7,691,105	669,110	51,158	1,609

Table 1. Characteristics of real datasets

4.2 Perofrmance Analysis

In section 3, we introduce four discretization methods for Web data. They are Equal Width Interval (EWI), Supervised Adaptive Width Interval(SAWI), Mixed Equal Frequency and K-Means(MEK), and Two-way discretization. In this section, we adopt these methods with real Web usage datasets to evaluate the performance of these algorithms.

Experiment 1. In this experiment, we increase the size of dataset for discretization. The variable for discretization is the staying time on each page. After preprocessing of ES1, we got a dataset with continuous values $T = \{t_1, ..., t_n\}$ with 39,500 instances. We classify Web pages into two categories: index pages and content pages. The number of interval k is determine by $In(n)$, where n is number of values. We adopt EWI, SAWI and MEK for testing.

In Fig 3, we can see that the running time for three different discretization methods is satisfying. Since EWI is the simplest method, so it needs less time to finish the discretization task.

Experiment 2. In [12], we suggest a multidimensional model to store information from Web data, which focuses on Web page, user and time profiles. In this experiment, we used ES4 to do the experiment. The dataset contains the access information of 120 members during April, 2003. We have also recorded the member information for user profile. We increase the number of dimensions(attributes) from the multidimensional model to test the scalability of different discretization methods. Some attributes are generated by other attributes,e.g., AD value. We also adopt EWI, SAWI and MEK for testing. In this experiment, we set the number of instances as 500. Fig 4 shows the experiment results. We can see that the running time of SAWI obviously longer than the others. It can be explained that each attribute may have its own class label, a lot of computation spent on discretizing data from different classes, individually.

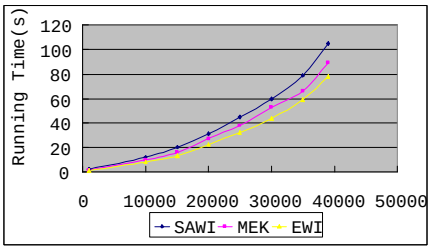

Fig. 3. Increasing Number of Continuous Values

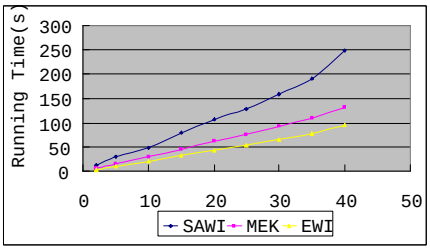

Fig. 4. Increasing Number of Dimensions

Experiment 3. In this experiment, we want to evaluate the affects of the number of intervals k to the performane of different discretization methods. We used ES2 dataset to generate AD matrix for testing the four discretization methods suggested. From the experiment result in Fig 5, we notice that MEK method spent longer time when increasing k. The reason is that when the number of bins increase, the cost for performing k-means process will also increase a lot. However, usually k is not a large number. Hence, the number of intervals won't affect the performance of discretization much.

Experiment 4. The last experiment is to test the scalability of these discretization methods for large matrices. Using ES5, we want to discretize AI matrix for dense regions discovery. The experiment result demonstrate that due the time complexity of high dimensional data, the running time will increase significantly, especially the MEK and Two-way discretization. However, the size of matrix is less than 1,000, the running time is still acceptable.

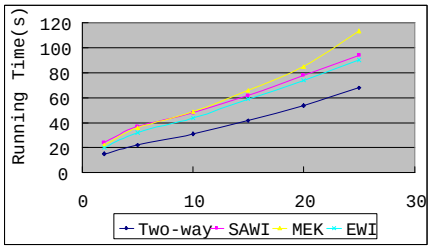

Fig. 5. Increasing Number of Intervals

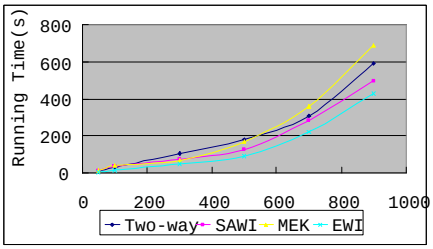

Fig. 6. Increasing Matrix Size

5 Conclusion

In this paper, we suggest several methods for Web data discretization. These methods can meet the needs of different data mining tasks, especially for mining dense regions from large matrices. We also the effects of the discretization results to patterns discovery. The discretization methods for Web data can be used in real Web information management applications, such as detecting Web usage patterns and analyzing user behavior. In the future, we purpose to implement the discretization algorithms in a multidimensional Web usage mining system.

References

1. Catlett, J. *On changing continuous attributes into ordered discrete attributes*. In Proceedings of the European Working Session on Learning (1991), pp. 164–178.
2. J. Dougherty, R. Kohavi, and M. Sahami, *Supervised and Unsupervised Discretization of Continuous Features*, Proceedings of International Conference on Machine Learning, Tahoe City, CA, 1995, pp. 194-202.
3. U. M. Fayyad and K. B. Irani, *Multi-interval discretization of continuous-valued attributes for classification learning*, Proc. of the 13th Intl. Joint Conf. on Artificial Intelligence, IJCAI-93: Chambery, France, 1993.
4. Gama, J., Torgo, L., and Soares, C. *Dynamic discretization of continuous attributes*. In Proceedings of the Sixth Ibero-American Conference on AI (1998), pp. 160–169.
5. R. C. Holte, *Very Simple Classification Rules Perform Well on Most Commonly Used Datasets*, Machine Learning, vol. 11, pp. 69-91, 1993.
6. Kerber, R. *Chimerge: Discretization for numeric attributes*. In National Conference on Artificial Intelligence (1992), AAAI Press, pp. 123–128.
7. Richeldi, M., and Rossotto, M. *Class-driven statistical discretization of continuous attributes*. In European Conference on Machine Learning (335-338, 1995), Springer.
8. M. Robnik-Sikonja and I Kononenko. *Discretization of continuous attributes using relie*. In Proceedings of RK95, 1995.
9. J. Srivastava, R. Cooley, M. Deshpande, and P. N. Tan. *Web Usage Mining: Discovery and applications of usage patterns from web data*. SIGKDD Explorations, 1:12-23, 2000.
10. E. H. Wu, M. K. Ng, and J. Z. Huang, *On improving website connectivity by using web-log data streams*, Proc. of the 9th International Conference on Database Systems for Advanced Applications (DASFAA 2004), Jeju, Korea, 2004.
11. E. H. Wu, M. K. Ng, *A graph-based optimization algorithm for Website topology using interesting association rules*, Proc. of the Seventh Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2003), Seoul, Korea, 2003.
12. E. H. Wu, M. K. Ng, and J. Z. Huang, *An efficient multidimensional data model for Web usage mining*, Proceeding of the Sixth Asia Pacific Web Conference(APWEB2004), 2004.
13. C. Yang, U. Fayyad, and P. S. Bradley, *Efficient discovery of error-tolerant frequent itemsets in high dimensions*, Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining: San Francisco, California, pp. 194–203, 2001.
14. Q. Yang, J. Z. Huang and M. K. Ng,*A data cube model for prediction-based Web prefetching*, Journal of Intelligent Information Systems, 20:11-30, 2003.

15. Andy M. Yip, Edmond H.Wu, Michael K.Ng, Tony F. Chan, *An efficient algorithm for dense regions discovery from large-scale data stream*, Proc. of the 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD2004), 2004.

Appendix 1: Counting of Accesses

Since we want to investigate the users navigational behaviors under certain Website topology, the user access sessions reconstructed are the users traversal paths. A user access session is equivalen to a traversal path in this paper. As we have mentioned, if Web pages are cached in the Web browser, the Web server will not record the backtrack accesses in the Web server logs. However, we can recover backtrack accesses by checking the Website topology. Then, we present a data structure of aggregating access sessions as follows: Given a Website topology G=V,E, where V=V1, V2, ,Vn and E=E1,E2, , Em. Also, given a user acce session dataset D=S1, S2,Sk, for each access session SD, S=P1, P2,, Pr, where PiV, i=1,, r. We set Cij= Count(Pi, Pj) to indicate the number of accesses from Pi to Pj in the acces session dataset D. We notice that the access session is the traversal path that a user follows in a Website topology. Therefore, any two adjacent pages in the session must be accessible by one click. Therefore, there are two cases of such counting, one is that Pi and Pj are adjacent in the access session S which also means the link of Pi Pj $\in$ E. The other case is that Pi and Pj are adjacent which means there exist at least a page between Pi and Pj. For the first case, we will count it whenever it appears. As to the second case, we will only count once in the access session.

For example, given access session S=A, B, C, B, in the first case, we will count AB, BC, and CB once since they are adjacent in the session, respectively. As to the second case, we count AC once. But we wont count AB again since it has been counted once in the first case. So, Count(A, B)

Appendix 2: Access Efficiency

From the above analysis, we need to find new measurements to evaluate the Website access situation under a Website topology. For a given Website topology, visitors must follow certain traversal paths to access the Web pages that they are interested in. For instance, if a visitor wants to sequentially visit Web page A, F,E (See Fig 1), the shortest traversal path is A, B, F, B, A, C, H, E. The corresponding access sequence is S = AB, BF, FB, BA, AC, CH, HE. Thus, the visitor should click at least seven times to access the target pages A, F, E. A access P1 P2 is defined as the access from page P1 to P2. If a visitor wants to browse the same target pages in a different order A, E, F, another traversal path A, C, H, E, H, C, A, B, F with eight clicks is needed, the corresponding accesses are AC, CH,HE, EH, HC, CA, AB, BF. But if one want to access other 3 pages A, B, G, just two accesses are enough. We observe that the access efficiency of A, F, E is low due to the redundancy of accesses. In general, there are two types of access redundancy. One is the jump-track access. For example, starting from A, the target page is F, we must access B first before we access F. The other type of redundancy access is the backtrack access, e.g., in order to access C from B, a visitor must click the back button in the browser to go from B to A and then to C, even though A has been accessed previously. The difference

between the jump-track access and backtrack access is determined by looking into whether the access has been performed in a visitor access session, e.g., in accessing A,F, E, AB is a jump-track access while BA is backtrack access. In this paper, we call both of them non-target accesses, comparing with target accesses which acquire the target Web pages directly. In order to measure the access efficiency of a visitor access session for a Website topology, we define the User Access Efficiency (UAE) as follows:

Definition 2. *Given a user access session $S = s_1, s_2, , s_m$, $A = a_1, a_2, , a_i$ is the ascend accessing session, $B = b_1, b_2, , b_j$ is the backtrack access session, where $A \subset S$ and $B \subset S$, $A \cap B = \Phi$. The User Access Efficiency (UAE) is given*

$$UAE(S) = 1 - \frac{|A| + |B|}{|S|} \tag{1}$$

where $|S|$ is the length of session S.

If Web pages are cached in the Web browser, the Web server will not record the backtrack accesses in the Web-log. For example, a visitor follows a traversal path A, C, H, E, H, C, A, B, F, the Web-log will only contain A, C, H, E, B, F, which is not a complete traversal path. Therefore, we propose a new measure called Server Access Efficiency to evaluate the access efficiency from thee Web server side of view.

Definition 3. *Given a user access session $S = s_1, s_2, , s_m$, $A = a_1, a_2, , a_i$ is the ascend accessing session, $B = b_1, b_2, , b_j$ is the backtrack access session, where $A \subset S$ and $B \subset S$, $A \cap B = \Phi$. The Serve Access Efficiency (SAE) is given*

$$SAE(S) = 1 - \frac{|A|}{|S| - |B|} \tag{2}$$

where $|S|$ is the length of session S.

Appendix 3: Indexes for Access Patterns Discovery

In our precious work, we suggest a novel measure named Access Distinctness (AD) as below: **Definition 1** Given Aggregating Session Matrix C and Topology Probability Matrix P, the Access Interest Matrix is given by:

$$AD(i, j) = In(\frac{C_{ij}}{P_{ij}} + 1) \tag{3}$$

where C_{ij} is the number of accessing from Web page V_i to V_j, and P_{ij} is the probability from V_i to V_j.

If an access pattern is distinct, it means the pattern is frequent user access pattern, i.e., people would like to visit. However, the access efficiency of the pattern is relatively low, i.e., a visitor is hard to access related Web pages. A pattern may raise our concerns if its AD value is higher than most other patterns.

In practice, Website managers are eager to know which contents can attract the visitors. It is natural that visitors will spend more time on the Web pages they interested in. Hence, the temporal factor should also be taken into consideration. Since the Web log also contains the access timestamps, so we can record the time period spent on each page. Table 2 shows the staying time on each page in the sample Website. Thus, we can use these information to identify really interesting access patterns. For this purpose, we suggest another interestingness measure named Access Interest (AI) as below:

Definition 2 Given Aggregating Session Matrix C and Topology Probability Matrix P, page staying time T, the Access Interest Matrix is given by:

$$AI(i,j) = In(\frac{C_{ij}T_{ij}}{P_{ij}} + 1) \tag{4}$$

where C_{ij} is the number of accessing from Web page V_i to V_j, and P_{ij} is the probability from V_i to V_j, T_{ij} is the average stay time of visiting V_i and V_j. Given $T = T_1, T_2, ..., T_n$, T_i is the average staying time of V_i. We define $T_{ij} = T_i + T_j$.