An Efficient Algorithm for Dense Regions Discovery from Large-Scale Data Streams

Andy M. Yip[‡], Edmond H. Wu[†], Michael K. Ng[†], 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,
405 Hilgard Avenue, Los Angeles, CA 90095-1555, USA.
[‡]mhyip@math.ucla.edu,chan@math.ucla.edu

Abstract. We demonstrate that dense regions discovery is an important process for finding distinct and meaningful patterns from given data. Some fundamental theories for dense regions are developed based on which novel efficient algorithms for identifying such regions are derived. Next, we illustrate how to adapt our algorithms for handling high-dimensional dense regions and data streams. Finally, experiments on large-scale data and data streams such as clickstreams are given which confirm that our algorithms are highly efficient and effective in dense regions discovery.

1 Introduction

A goal of data mining is to discover useful patterns from a given data set or data stream. Examples of common data mining tasks include clustering, classification and association rules mining, see [3] and the references therein for techniques to accomplish these tasks.

Besides the patterns identified by the aforementioned data mining tasks, we realize that *dense regions*[1], which are two-dimensional regions defined by subsets of entities and attributes whose corresponding values are mostly constant, are another type of patterns which are of practical use and are significant. For example, we find that in some microarray data, a number of genes may have about the same expression level over a period of time [1]. By identifying several of such subsets of genes and experimental conditions, one might infer, for example, the underlying regulatory mechanisms. Another example is that in some cases dense regions can be used to evaluate discriminability of a subset of attributes, thus identifying such regions enhances the feature selection process in classification applications. Furthermore, we also observe such patterns in stream data such as clickstreams. These patterns can then be used to analyze users behavior and to improve web page or website topology design.

[1] Note that a dense region usually refers to a subset of the space where data points are highly populated whereas dense regions discussed in this paper lie in the entity-by-attribute space. Our precise definition is given in Section 2.

To the best of the authors' knowledge, dense region patterns are not previously explicitly studied to a rigorous extent. A similar but not identical notion is error-tolerant frequent itemsets introduced by Yang et al in [7] which focuses on mining association rules.

Our goal in this paper is to provide some solid theories for dense regions from which efficient and effective algorithms are derived. We characterize dense regions into three categories — non-covered ($\mathcal{NC}$), recursively covered ($\mathcal{C}_{\mathcal{RC}}$), and non-recursively covered ($\mathcal{C}_{\mathcal{NRC}}$). Due to the lack of space, proof of the theorems are omitted and will appear in a later paper.

A fundamental algorithm, called DRIFT (Dense Region IdentiFicaTion), is proposed for finding two-dimensional (entity-by-attribute) dense regions in large data sets. Such an algorithm is also served as a building block for more sophisticated algorithms which can be applied to high-dimensional dense regions and data streams. As an illustration to the usefulness of dense regions and the algorithms, we apply our algorithms to clickstream analysis.

2 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 (maximality).*

In the sequel, we assume that all dense regions are with respect to a common target value v unless specified otherwise.

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.

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

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 the number of dense regions may grow exponentially with the matrix size. For example, an n-by-n identity matrix has $2^n - 2$ dense regions with value 0. However, by discarding dense regions having size smaller than a specified threshold, which is what we do in our experiments, the number of legitimate dense regions is greatly reduced to a reasonable value. Moreover, dense regions having larger size are of more significance (against dense regions found in random matrices).

3 The DRIFT Algorithm

3.1 The BasicDRIFT Algorithm

The BasicDRIFT algorithm starts from a given point (s, t) (which contains the target value v) and returns two dense regions containing (s, t) where one of them is obtained by a vertical-first-search; the other is by a horizontal-first-search. The algorithm is outlined in Fig. 1.

```
**Algorithm:** BasicDRIFT($X, s, t$)

/* Vertical-first-search */
$R_v \leftarrow \{1 \le i \le n | X_{it} = X_{st}\}$
$C_v \leftarrow \{1 \le j \le p | X_{ij} = X_{it} \forall i \in R_v\}$
/* Horizontal-first-search */
$C_h \leftarrow \{1 \le j \le p | X_{sj} = X_{st}\}$
$R_h \leftarrow \{1 \le i \le n | X_{ij} = X_{sj} \forall j \in C_h\}$
**Return** $\{R_v \times C_v, R_h \times C_h\}$
```

Fig. 1. The BasicDRIFT algorithm.

Suppose the two returned dense regions have dimensions n_v-by-p_v and n_h-by-p_h respectively. The number of computations required by the algorithm is $n + n_v p + p + p_h n$. Thus, the complexity of the algorithm is linear in size of the data matrix. Moreover, in practice, n_v and n_h are much smaller than n, and, p_v and p_h are much smaller than p. In this case, the complexity is of $O(n + p)$ essentially.

A property of the BasicDRIFT is that the two returned regions are in fact dense regions — though they may be identical. This fact is stated in the following theorem which is a direct consequence of the construction of the algorithm.

Theorem 1. *The submatrices $R_v \times C_v$ and $R_h \times C_h$ obtained from the BasicDRIFT algorithm are dense regions containing the starting point.*

3.2 Isolated Points and Covered Dense Regions

To further understand the properties of the BasicDRIFT algorithm, we introduce the notions of *isolated point* and *covered dense region*.

Definition 2 (Isolated points). *A point (i, j) in a dense region D is isolated if it is not contained in any other dense region.*

Definition 3 (Covered dense regions). *A dense region D with respect to v is said to be a covered dense region, or simply covered, if there exists a set of dense regions $\{D_i\}$ with respect to v where $D_i \neq D$ for all i such that $D \subset \cup_i D_i$. A dense region that is not covered is called non-covered. For a fixed X, the class of all covered dense regions in X is denoted by $\mathcal{C}$ while the class of all non-covered dense regions is denoted by $\mathcal{NC}$.*

Note that $D \neq \cup_i D_i$, for otherwise some D_i's are not maximal. The following theorem says that whether a dense region is in the class $\mathcal{C}$ or in the class $\mathcal{NC}$ can be exactly characterized by its number of isolated points.

Theorem 2. *A dense region is in $\mathcal{C}$ if and only if it has no isolated point.*

Example 2. The dense region $\{1,2,3,4\} \times \{1\}$ in Example 1 is covered by the union of the dense regions $\{1,2,3\} \times \{1,4\}$ and $\{2,3,4\} \times \{1,2\}$. The point $(1,4)$ is an isolated point as it only belongs to the dense region $\{1,2,3\} \times \{1,4\}$.

3.3 Discovery of the Class $\mathcal{NC}$ by the BasicDRIFT Algorithm

In this subsection, we prove an important property that all dense regions in $\mathcal{NC}$ can be found by the BasicDRIFT algorithm.

Theorem 3. *The BasicDRIFT algorithm starting at (s,t) returns two identical dense regions if and only if (s,t) is an isolated point.*

The above theorem has two implications. First, it characterizes whether a starting point is isolated or not. Second, it guarantees that all non-covered dense regions can be found by starting the BasicDRIFT at every point in X having the target value v. More precisely, a non-covered dense region is discovered when one of its isolated points is encountered.

We remark that, although there is no guarantee, a covered dense region may be discovered by the BasicDRIFT algorithm.

Example 3. The covered dense region $\{1,2,3,4\} \times \{1\}$ in Example 1 can be identified by starting the BasicDRIFT algorithm at $(1,1)$.

3.4 The Classes $\mathcal{C}_{\mathcal{NRC}}$ and $\mathcal{C}_{\mathcal{RC}}$

First, an example where a dense region is unidentifiable by the BasicDRIFT algorithm is given below.

Example 4. The dense region in bold face cannot be identified by the BasicDRIFT

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & \mathbf{1} & \mathbf{1} & 1 \\ 1 & \mathbf{1} & \mathbf{1} & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

We observe that in Example 4, the covered dense region is surrounded by four isolated points (these points are identifiable in view of Theorem 3). If we remove these isolated points (by resetting their values to be any value other than v), then the covered dense region in bold face becomes non-covered (with respect to the modified data matrix) and is identifiable by another sweep of the BasicDRIFT — thus removal of isolated points enlarges the set of identifiable dense regions by the BasicDRIFT. Moreover, such a procedure can be done repeatedly which allows us to find more dense regions.

Next, we show an example that some covered dense regions are still unidentifiable by the BasicDRIFT even after removal of isolated points.

Example 5. The following data matrix contains no isolated point and the dense region in bold face cannot be identified by the BasicDRIFT:

$$\begin{pmatrix} \mathbf{1} & 1 & 0 & \mathbf{1} \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ \mathbf{1} & 0 & 1 & \mathbf{1} \end{pmatrix}.$$

In view of Examples 4 and 5, we may further divide the class $\mathcal{C}$ into two subclasses $\mathcal{C}_{\mathcal{NRC}}$ and $\mathcal{C}_{\mathcal{RC}}$ according to their identifiability by the BasicDRIFT.

Definition 4 (The class $\mathcal{C}_{\mathcal{NRC}}$). *A covered dense region is in $\mathcal{C}_{\mathcal{NRC}}$ if it becomes non-covered after a finite number of rounds of removal of isolated points. We also say that such a region is non-recursively covered.*

Definition 5 (The class $\mathcal{C}_{\mathcal{RC}}$). *A covered dense region is in $\mathcal{C}_{\mathcal{RC}}$ if it remains covered after any number of rounds of removal of isolated points. We also say that such a region is recursively covered.*

3.5 Discovery of the Class $\mathcal{C}_{\mathcal{RC}}$ by the ExtendedBasicDRIFT

Since dense regions in the class $\mathcal{C}_{\mathcal{RC}}$ are not guaranteed to be found by the BasicDRIFT algorithm, we now introduce an extended version called ExtendedBasicDRIFT (depicted in Fig. 2) to discover dense regions in $\mathcal{C}_{\mathcal{RC}}$. The idea is that after finding the set R_v (respectively C_h) in the original BasicDRIFT algorithm starting at (s, t), we perform a horizontal (respectively vertical) search over all possible subsets of $R_v \setminus \{s\}$ (respectively $C_h \setminus \{t\}$). Theoretically, the ExtendedBasicDRIFT algorithm is capable of finding all dense regions. However, since it requires more computations than the BasicDRIFT does, we only use this algorithm to find dense regions in the class $\mathcal{C}_{\mathcal{RC}}$. The question now becomes how to combine the two algorithms in an effective way.

3.6 The DRIFT Algorithm

In this subsection, we present the DRIFT algorithm which utilizes the BasicDRIFT and the ExtendedDRIFT algorithms to find dense regions efficiently. The algorithm is depicted in Fig. 3.

```
Algorithm: ExtendedBasicDRIFT($X, s, t$)

$D \leftarrow \{\}$
/* Vertical-first-search */
$R_v \leftarrow \{1 \leq i \leq n | X_{it} = X_{st}\}$
For each subset $R_{v_k}$ of $R_v \setminus \{s\}$ do
    $C_{v_k} \leftarrow \{1 \leq j \leq p | X_{ij} = X_{it} \forall i \in R_{v_k}\}$
    If $R_{v_k} \times C_{v_k}$ is not a subset of any dense region in $D$
        Insert $R_{v_k} \times C_{v_k}$ into $D$
    EndIf
EndFor
/* Horizontal-first-search */
$C_h \leftarrow \{1 \leq j \leq p | X_{sj} = X_{st}\}$
For each subset $C_{h_k}$ of $C_h \setminus \{t\}$ do
    $R_{h_k} \leftarrow \{1 \leq i \leq n | X_{ij} = X_{sj} \forall j \in C_{h_k}\}$
    If $R_{h_k} \times C_{h_k}$ is not a subset of any dense region in $D$
        Insert $R_{h_k} \times C_{h_k}$ into $D$
    EndIf
EndFor
Return $D$
```

Fig. 2. The ExtendedBasicDRIFT algorithm for discovery of dense regions in $\mathcal{C}_{\mathcal{RC}}$.

The DRIFT algorithm works as follows. The first pass of the while-loop invokes the BasicDRIFT to spot out all dense regions in the class $\mathcal{NC}$. After the each pass, all isolated points found are removed and thus allows discovery of dense regions in the class $\mathcal{C}_{\mathcal{NRC}}$. When no further isolated point is found, it is guaranteed that all dense regions in $\mathcal{NC}$ and $\mathcal{C}_{\mathcal{NRC}}$ are discovered. We remark that beginning from the second pass of the while-loop, where the original matrix X is modified, the regions returned by the BasicDRIFT may be proper subsets of a previously found dense region. In this case, these subsets are regarded as "illegitimate" and are not inserted into D.

After identifying all dense regions in $\mathcal{NC}$ and $\mathcal{C}_{\mathcal{NRC}}$, we start the Extended-DRIFT at every point in S_{reduced}, which contains no more isolated point, to find the dense regions in $\mathcal{C}_{\mathcal{RC}}$.

We remark that, as mentioned at the end of Section 2, one might want to discard dense regions with small size. To do so, one may define a dense region to be "illegitimate" if its size is below a user-specified threshold and thus it is not inserted into the output sequence D.

4 Adapting the DRIFT for Different Data Types

4.1 Generalization to μ Dense Regions

So far we considered regions with constant value. In practice, regions with only majority of constant value may also be interested. In this subsection, we discuss the notion of μ dense regions where the density property of dense regions is relaxed. A region $R \times C$ is called μ dense regions with value v if at least a percentage μ of the entries of $R \times C$ take the value v. For example, a vector with entries (0,1,1,1) is a 75% dense region (in fact it is a μ dense region for any $0 \leq \mu \leq 75\%$). We now present a modified version of DRIFT algorithm, called

```
Algorithm: DRIFT($X, v$)

$D \leftarrow \{\}$, $Flag = TRUE$
/* Finding dense regions in $\mathcal{NC}$ and in $\mathcal{C}_{\mathcal{NRC}}$ */
While $Flag = TRUE$
    $Flag = FALSE$, $S \leftarrow \{(i,j) | X_{ij} = v\}$
    If $S = \{\}$, Break, EndIf
    For each $(s,t) \in S$
        $\{R_v \times C_v, R_h \times C_h\} \leftarrow$ BasicDRIFT($X, s, t$)
        If $R_v \times C_v$ is a legitimate dense region
            Insert $R_v \times C_v$ into $D$
        EndIf
        If $R_h \times C_h$ is a legitimate dense region
            Insert $R_h \times C_h$ into $D$
        EndIf
        If $R_v \times C_v = R_h \times C_h$
            $X_{st} \leftarrow \infty$, $Flag = TRUE$
        EndIf
    EndFor
EndWhile
/* Finding dense regions in $\mathcal{C}_{\mathcal{RC}}$ */
$S_{\text{reduced}} \leftarrow \{(i,j) | X_{ij} = v\}$
For each $(s,t) \in S_{\text{reduced}}$
    $D' \leftarrow$ ExtendedBasicDRIFT($X, s, t$)
    For each $R \times C \in D'$
        If $R \times C$ is a legitimate dense region
            Insert $R \times C$ into $D$
        EndIf
    EndFor
EndFor
Return $D$
```

Fig. 3. The DRIFT Algorithm.

μDRIFT, to deal with μ dense regions (see Fig. 4). Starting with a dense region $R \times C$ found by the DRIFT, the μDRIFT algorithm repeatedly enlarges the region in a greedy way where the density of the enlarged region is guaranteed to be at least μ.

4.2 Generalization to Continuous Data

In the previous algorithms, we assume that each attribute is categorical. Also, for a target value (category) v, an entry (i,j) is selected if $X_{ij} = v$. However, such an assumption can be relaxed by allowing other logical and relational operators, thus the algorithms can be easily adapted to other data types.

For example, if v takes a continuum of value, we can set $X(i,j) \leq v$ (or $\geq$) as the entry selecting condition. An alternative way is to apply quantization techniques to deal with continuous data, see [2].

By extending the data types and conditions that the DRIFT algorithms can handle, we can perform diverse mining tasks with more flexibility.

4.3 Generalization to High-Dimensional Data

In the previous sections, we consider dense regions of the form $R \times C$ which are two-dimensional. Another kind of extension is to discover high-dimensional dense regions, i.e., dense regions of the form $D_1 \times \ldots \times D_k$ for $k \geq 2$. In this paper,

```
Algorithm: $\mu$DRIFT($X, v, R, C, \mu$)

/* $R$ and $C$ define a previously identified dense region or a $\mu$ dense region */
/* found by the DRIFT or $\mu$DRIFT respectively */

$C_\mu \leftarrow C$, $R_\mu \leftarrow R$
For each column $j$ not in $C$ /* Search horizontally*/
    $Count \leftarrow$ number of $v$'s in $X(R, \{j\})$
    If $Count \geq \mu|R|$
        $C_\mu \leftarrow \{j\} \cup C_\mu$
    EndIf
EndFor

For each row $i$ not in $R$ /* Search vertically*/
    $Count \leftarrow$ number of $v$'s in $X(\{i\}, C_\mu)$
    If $Count \geq \mu|C_\mu|$
        $R_\mu \leftarrow \{i\} \cup R_\mu$
    EndIf
EndFor

/*Repeat the process to enlarge the region until no further improvement*/
If $C_\mu \neq C$ or $R_\mu \neq R$
    $\mu$DRIFT($X, v, R_\mu, C_\mu, \mu$)
Else
    Return $R_\mu$, $C_\mu$
EndIf
```

Fig. 4. The μDRIFT Algorithm.

we suggest two types of high-dimensional dense regions, namely, single-measure dense regions (SMDR) and pairwise-measure dense regions (PMDR).

Definition 6 (SMDR). *Given a multi-dimensional dataset* $\vartheta = \{D_1, D_2, \ldots, D_r\}$ *with* r *attributes, a single-measure region (SMR) is a tuple* $T = \{\vartheta', V\}$ *where* $\vartheta' \subset \vartheta$ *and* V *is a common measure of all the attributes in* ϑ'*. A SMR is a high-dimensional single-measure dense region (SMDR) if and only if any two-dimensional region in SMR is a dense region with respect to* V*.*

Definition 7 (PMDR). *Given a multi-dimensional dataset* $\vartheta = \{D_1, D_2, ..., D_r\}$ *with* r *attributes and a set of measures* $P = \{P_1, ..., P_{r(r-1)/2}\}$*, a pairwise-measure region (PMR) is a multi-dimensional subset* $T = \{\vartheta', P'\}$ *where* $\vartheta' \subset \vartheta$ *and* $P' \subset P$ *is the set of measures for all pairwise combinations of the attributes in* ϑ'*. A PMR is a high-dimensional pairwise-measure dense region (PMDR) if and only if every two-dimensional regions in PMR is a dense region with respect to their corresponding measure* P'_i*, where* $1 \leq i \leq |P'|$*.*

Example 6. In web usage analysis, given a three-dimensional dataset with attributes $\{User, Page, Time\}$. If we want to find a group of users who visit the web pages $\{A, B, C\}$ at least one time on Sundays, then the query is equivalent to find the SMDRs with respect to the common measure, the visiting frequency.

Using the same data, if we want to find the users, pages, and time slots which satisfy: (a) the $\{User, Page\}$ measure — the probability of the users visiting the pages is more than 50%; (b) the $\{Page, Time\}$ measure — the pages should be accessed at least 1000 times totally at each time slot; (c) the $\{User, Time\}$

measure — the users should have already stayed in the website for than half an hour at each time slot. This complicated query is to find the PMDRs with respect to three different measures.

From the definitions of SMDR and PMDR, we can easily extend the DRIFT algorithm for high-dimensional dense regions since both SMDR and PMDR discovery problems can be reduced into the sub-problems of two-dimensional dense region discovery. First, find two-dimensional dense regions using the DRIFT, then search another dimension to find higher dimensional dense regions based on the submatrices found. In practice, applying the DRIFT in high-dimensional dense region discovery is also a desirable solution due to the efficiency of the DRIFT. Moreover, together with the data cube model proposed in [4], dense regions discovery from multi-dimensional data streams can be made very efficient.

4.4 Generalization to Stream Data

In many practical situations, data are generated in the form of continuous, rapid data streams, such as clickstreams. Therefore, the efficiency of a data stream algorithm is one of the most important considerations, especially when performing some online data mining tasks. In this subsection, we purpose two feasible strategies to improve the efficiency of dense regions discovery from data streams. These strategies focus on how to update the changing dense regions in a data stream environment.

Snap-Shot Update (SSU)
One strategy is to give snap-shots to the changing data streams. Given a stream window W_i and an incremental window ΔW_i, we have $W_{i+1} = W_i + \Delta W_i$ for $i = 0, 1, \ldots$, starting from an initial window ΔW_0. In each stream window W_i, the corresponding snap-shot of the data matrix is denoted by X_i. We regard X_i as a static matrix in this stream window, then we can use the DRIFT algorithm to discover dense regions in the partitions of original data streams.

The benefit of this strategy is that it is easy to implement. However, the drawback is that this method may not be so efficient, especially for large matrices. It is because for each new stream window W_i, we need to search all the dense regions in the matrices again where many of them may not be changed at all. Another drawback is that it is not easy to determine the size of the incremental window ΔW_i. If we set ΔW_i to be too small (e.g., 10 seconds), then the computational cost will significantly increase due to the massive updates. Conversely, if we set ΔW_i to be too large, then the dense regions found may not be useful, especially for those dense regions that change frequently.

Point-Trigger Update (PTU)
Another strategy is called Point-Trigger Update. In a data stream environment, dense regions change constantly. However, a new arriving data point will only cause changes to a small portion of the existing dense regions at that moment.

Therefore, we may just update those dense regions associated with the actually changed data entries. In this way, a lot of redundant computation cost is saved.

In fact, the notion of isolated points is very useful for reducing computations. Since an isolated point belongs to only one dense region, a change in the value of an isolated point only affects the region that it belongs (before the change). We may also employ indexing to pair up an isolated point and its corresponding non-covered dense region. More precisely, if an entry (s, t) is an isolated point (determine by the BasicDRIFT algorithm), then we index the dense region as $D(s, t)$. After indexing, we implement the Point-Trigger Update in data streams. Such a process is shown in Fig. 5.

```
$Point - TriggerUpdate$
Input: A changing entry $(s, t)$ in $X$
    If $(s, t)$ is an isolated point (before the change)
        Run BasicDRIFT($X, s, t$) to update dense region $D(s, t)$
    Else
        For each $D(s', t')$ containing $(s, t)$ where $(s', t')$ is an isolated point (before the change)
            Run ExtendedBasicDRIFT($X, s', t'$) to update dense region $D(s', t')$
        EndFor
    EndIf
Output: Updated dense regions associated with the changing entry $(s, t)$
```

Fig. 5. Point-Trigger Update

The most important advantage of PTU is that the update cost is greatly reduced, especially for large matrices. Moreover, the PTU is more adaptable than SSU. In SSU, the size of the incremental window ΔW affects the performance of the algorithm to a large extent. In some cases, large volumes of data streams arrive very quickly, the preset size of the incremental windows may not be able to adapt to the fast changes in the data streams. However, PTU can accelerate the update speed based on the volumes of arriving stream data. The updates are continuous and on time, hence the dense regions found based on PTU can well represent the potential patterns in data streams.

5 Experimental Results

5.1 Web-log 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 DRIFT algorithm. In our experiments, we take the data stream as a sequence of data items $X = x_1, \ldots, x_n, \ldots$, where the sequence is scanned only once in the increasing order of the indices. Each data stream contains a set of continuous access sessions during some period of time. Using the cube model purposed [4], we can convert original web-log streams into access session data streams for dense regions discovery.

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 while ES4 and ES5 are the log datasets during 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 the datasets ES1–ES5.

5.2 Performance Evaluation

Experiment 1. We evaluate the scalability of the DRIFT with respect to increasing matrix size. We use the ES2 dataset to find dense regions (DRs) in page probability matrix (refer to [6]). The experiment results show that even for a matrix with one million entries, the running time is still acceptable (See Fig. 6). We can also compare the running time of finding dense regions in $\mathcal{C}$ and $\mathcal{NC}$ with the DRIFT.

Experiment 2. We test the efficiency of the DRIFT and the μDRIFT algorithms. We use the ES1 to discover DRs in page access frequency matrix (refer to [5]). We set the minimal size of the DRs to be 10×10. The results show that the DRIFT is efficient (see the second column of Table 2). Starting with the 100% DRs found by the DRIFT, the time for searching μDRs with the μDRIFT is also quite fast. The μDRIFT algorithm can find larger matrices which can be very helpful for finding more useful patterns.

μ Threshold	100%	95%	90%	85%	80%	75%
Number of DRs	261	383	506	596	687	758
Average Size	12×13	15×16	19×21	21×24	24×26	26×29
Maximal	28×31	32×38	42×51	56×43	69×47	84×102
Average Density	100%	96%	92%	89%	83%	78%
Running Time (sec.)	275	296	322	341	367	390

Table 2. Varying the μ threshold.

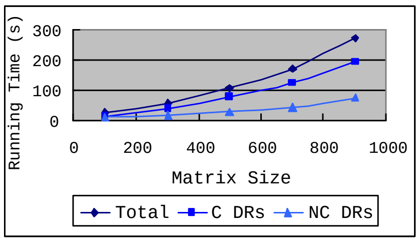

Fig. 6. Increasing matrix size

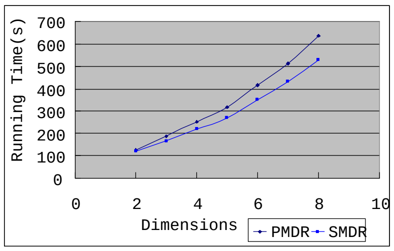

Fig. 7. Increasing dimensions

Experiment 3. Next, we test the scalability of the DRIFT in high-dimensional data using the ES3 dataset. We select up to eight attributes (dimensions) from the web usage data to perform high-dimensional DRs mining using the DRIFT.

The running time of finding SMDRs is less than that of PMDRs. It can be explained that, in SMDRs discovery, we can save the computation cost by pruning more SDRs in sparse dimensions that share the same measure. However, the running times for SMDRs and PMDRs discovery suggest that the DRIFT algorithm can be applied to high-dimensional data (see Fig. 7).

Experiment 4. This experiment is to compare the update cost of two update methods proposed in Section 4.4. We simulate the stream updates by using the ES4 dataset. The experiment results suggest that PTU is more adaptable than SSU to update the dense regions in a data stream environment (See Fig. 8). On average, the update cost by PTU is only around 16% of that by SSU. Moreover, the PTU can response to the peak period for updates (see the 18th time slot in Fig. 8) while SSU cannot. This experiment strongly demonstrate that the DRIFT algorithm is suitable for data stream mining due to its desirable properties.

Experiment 5. The last experiment is to employ the largest dataset ES5 to test the the scalability of the DRIFT with PTU to update the continuous arriving clickstream data. The experimental results show that both the searching time on $\mathcal{C}$ and on $\mathcal{NC}$ dense regions are acceptable. Moreover, even for several million of clickstreams per hour, the DRIFT is still robust for dense regions discovery (see Fig. 9).

The experiments above validated that the DRIFT algorithm is effective and efficient for large-scale data streams and can be applied in practical applications.

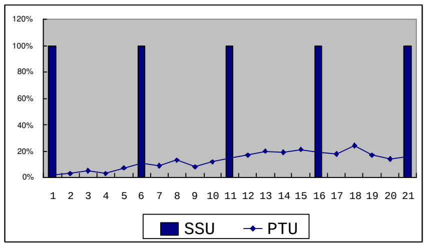

Fig. 8. Comparison between PTU and SSU.

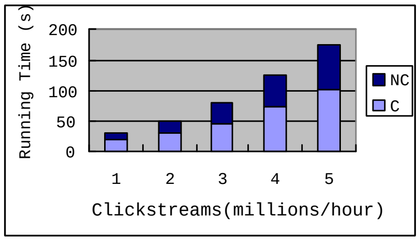

Fig. 9. Increasing data streams per hour.

6 Conclusion

We demonstrated that dense regions are significant patterns which are useful for knowledge discovery. Some theories on dense regions were also developed based on which efficient and effective algorithms are proposed. Our experiments confirmed that the DRIFT algorithm is very useful in data stream applications such as online Web usage mining. As future works, we would like to further develop some theoretical results for dense regions and explore the use of dense regions in other data mining tasks such as data clustering, classification, and association rule mining.

Indeed, we observe that the algorithms proposed by Wu et al in [5, 6] for mining usage patterns from websites and optimizing website browsing efficiency from using web-logs and website topology information share a common computational bottleneck when finding potential association patterns (e.g. a group of users who have common interests in a group of web pages) from large-scale matrices (page probability matrix [6] or access interest matrix [5]). We believe that the DRIFT algorithm can be incorporated into these algorithms to accelerate the processing time and to improve the quality of the mining results, especially for large-scale stream data.

References

1. M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein, *Cluster analysis and display of genome wide expression patterns*, Proceedings of National Academy of Sciences, **85**, pp. 14863–14868, 1998.
2. 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.
3. J. Han and M Kamber, *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, 2001.
4. J. Huang, M. Ng, W. Ching, J. Ng, and D. Cheung, *A cube model and cluster analysis for web access sessions*, Proceeding of the WEBKDD 2001 Workshop, San Francisco, USA, 2001.
5. 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.
6. E. H. Wu, M. K. Ng, *A graph-based optimization algorithm for Website topology using interesting association rules*, Proc. the Seventh Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2003), Seoul, Korea, 2003.
7. 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.