Temporal Neighborhood Discovery using Markov Models

Submitted for Blind Review

Abstract

Temporal data, which is a sequence of data tuples measured at successive time instances, is typically very large. Hence instead of mining the entire data,we are interested in dividing the huge data into several smaller intervals of interest which we call temporal neighborhoods. In this paper we propose an approach to generate temporal neighborhoods through unequal depth discretization.

We describe two novel algorithms (a) **S**imilarity based **Merg**ing ($SMerg$) and, (b) **St**ationary distribution based **Merg**ing ($StMerg$). These algorithms are based on the robust framework of Markov models and the Markov Stationary distribution respectively. We identify temporal neighborhoods with distinct demarcations based on unequal depth discretization of the data. We discuss detailed experimental results in both synthetic and real world data. Specifically we show (i) the efficacy of our approach through precision and recall of labeled bins, (ii) the ground truth validation in real world datasets and, (iii) Knowledge discovery in the temporal neighborhoods such as global anomalies. Our results indicate that we are able to identify valuable knowledge based on our ground truth validation from real world traffic data.

Keywords Temporal neighborhoods, Discretization, Markov Model, Stationary Distribution

1 Introduction

Phenomenon in space and time are captured in spatio temporal datasets such as (a) sensors monitoring environmental phenomenon, (b) disease spread in a region over a period of time, and (c) vegetation changes captured in satellite imagery over time.

If we consider the spatial and temporal aspects of data in combination, many interesting patterns may not be discovered due to the multi dimensional nature of the data. Previous [13, 16, 21] approaches have successfully shown that considering the cross sections in space and time leads to discovery of interesting and meaningful patterns which would otherwise not be possible. To perform knowledge discovery in such complex datasets, the data has to be broken up into relevant sub groups. In spatial data this can be done by generating meaningful neighborhoods comprising of similarly behaving spatial objects [3, 24]. In temporal data this can be done by discretizing the data into relevant intervals capturing the similar behavior in various intervals of time [8, 14, 15]. The intervals can be considered as the temporal neighborhoods which are relevant for knowledge discovery such as discovery of anomalous intervals or anomalous points within intervals.

In this paper we focus on discovering temporal neighborhoods by discretization of temporal data by dividing it into unequal depth bins. This is useful in various application domains. For instance, if we want to discover peak periods in traffic monitoring data or discover unusual peak periods on some days, we can do so by discovering the intervals and quantifying the behavior of the objects in the interval using approximations. Secondly, if we want to discover specific local anomalies in the temporal data then we can mine the intervals and compare the distance of the data points in the interval to discover the specific points which are most unusual as compared to the rest. Thus it can be seen that the quality of mining results is highly dependent on the quality of the discretization. In general time series data is measured at successively uniform intervals of time. However our focus is not limited to uniform intervals therefore we use the term temporal data as our approach can be generalizable to any temporal datasets. Particularly for the temporal analysis data needs to be discretized or divided into bins for the analysis. In general equal width and equal frequency discretization are the most commonly used techniques. However these methods impose unnatural bounds on the data and in several cases are vulnerable to outliers in the data affecting the boundaries and widths of the discrete intervals.

Let us consider a specific application in the domain of Transportation dealing with traffic congestion to motivate this problem. Here we consider the data being gathered by sensors that monitor the traffic at various locations along highways.

Example 1 *Traffic Congestion is defined [7] as the excess of vehicles on a portion of a roadway at a particular time resulting in slower speeds than normal. Our focus here is the Recurring congestion that occurs at regular times at a site (for example: morning or evening peak hour congestion, or congestion due to regular events such as a street market on a particular day each week). Traffic congestion has become a major problem for many American cities with a measurable impact as outlined in the Texas Transportation*

Institute's 2007 Urban Mobility Report [23]. Specifically Congestion leads to $78 billion annual drain on the U.S. economy comprising of 4.2 billion lost hours and 2.9 billion gallons of wasted fuel. The average peak period traveler ends up spending an extra 38 hours of travel time and consumes an additional 26 gallons of fuel, amounting to a cost of $710 per traveler. [23]
If we analyze the traffic of any one specific spatial location we can see that the behavior, captured in traffic data, is consistent and repetitive (example at peak periods) yet highly variable and unpredictable (example: change in traffic pattern due to frequent crashes). This embodies the behavior of a temporal process. However a problem like congestion has redefined peak periods (historically and across cities, small vs large cities, peak periods vary from peak hour to peak intervals [7]). Thus definition of peak periods is critical in comparing various traffic patterns.
We can see in this dynamic problem area several challenges emerge:(a) Addressing recurring problems is important since this will help in alleviating non-recurring problems, (b) The correct identification of peak periods is important for any traffic pattern analysis , (c) It is important to identify the right non-recurrent problems especially in peak periods so that the congestion does not worsen, (d) We need to consider the consistent vs. variable nature of the traffic data in such a way that small fluctuations do not impact the discovery of such peak periods at the same time these fluctuations for instance frequent crashes [25] can be captured in the intervals.

To resolve these challenges a robust framework is required which identifies these clearly demarcated distributions within the temporal data. Such a demarcation of the data should provide an optimal solution and should not be adhoc or heuristic based such as equal width or equal frequency discretization. [18] discusses the symbolic representation of temporal data by using SAX technique and also describes how the clustering works on the symbolic representation with a very high degree of accuracy. However it does not take into account the similarity in between successive intervals while creating this approximation. Since the PAA technique used by SAX relies on the mean for approximation, it is highly susceptible to be affected by noise or outliers, thereby losing some interesting subtle patterns. Extended SAX technique [19] was proposed to improve this drawback of PAA which is highly affected by extremely large or small values, by keeping min and max values for each interval but still it does not consider similarities in between adjacent intervals while obtaining a temporal approximation. In addition [10] and [12] discuss clustering temporal data. The major issue is that these techniques do not consider the similarity of data or distribution while creating bins. Further certain techniques such as equal frequency binning only consider the equal size of data in each of the bins. As a result, data tuples that are very similar to each other in terms of distribution may fall in different bins, even though they are adjacent. Hence local data mining applied to the bins may not be able to extract the expected pattern. Lastly the discretization should go beyond an approximation and be robust in the presence of noise. In this paper we address these issues to propose an approach for temporal neighborhood discovery using a Markov model based discretization technique.

Specifically we make the following contributions:

- We present a novel approach to generate temporal neighborhoods based on an unequal depth discretization for a more refined knowledge discovery in these neighborhoods.
- We model our approach using the robust framework of Markov Model to define the temporal dependencies in the data such that the intra neighborhood similarity and inter neighborhood dissimilarity are high.
- We also present a stationary distribution of Markov Model to approximate the original data keeping the properties of the temporal data intact.
- Both the algorithms take care of the transitive closure in terms of similarity in between the adjacent bins during the merging process.
- We present detailed experimental results for (i) the efficacy of our approach through precision and recall of labeled bins, (ii) the ground truth validation in real world datasets and, (iii) knowledge discovery in temporal neighborhoods such as global anomalies. Our results indicate that we are able to identify valuable knowledge based on our ground truth validation from real world traffic data.

The rest of the paper is organized as follows: In section 2 we outline our approach. In section 6 we discuss the experimental results. Finally we conclude in section 7.

2 Approach

In this paper we propose an approach to generate temporal neighborhoods by discretizing temporal data into unequal depth bins. We describe this process in the following distinct steps:
(a) **Markov Modeling:** We first begin with an equal frequency binning to divide data into initial equal depth bins. We consider these bins as the states of a Markov model. We then compute the similarity between the bins using a distance measure d. We use various distance measures as described in section 2.1.1. We then generate a transition matrix based on these similarities for this Markov model and subsequently normalize to obtain a row-stochastic matrix.
(b) **Unequal depth discretization:** In order to form an unequal depth discretization we propose two solutions. We use the intuition that the adjacent bins in the previous step having high degree of probability of transition should be

merged, in order to obtain unequal depth bins. Specifically we propose a *S*imilarity based *merg*ing (SMerg) and a Markov *St*ationary distribution based *merg*ing (StMerg). We next describe our approach in details. Figure 1 presents our approach using both the algorithms.

Table 2 summarizes the terminology used in this paper.

Symbols	Definitions
$X_{temporal}$	Temporal Dataset $\{X_1, X_2, \ldots, X_N\}$
T	Set of time instants corresponding to X $\{t_1, t_2, \ldots, t_N\}$
N	Size of Temporal Dataset
X_i	$X(t_i)$, the i^{th} Temporal Datapoint corresponding to time t_i
$B_{temporal}$	Set of Temporal Bins $\{B_1, B_2, \ldots, B_n\}$
n	Number of (initial) Temporal bins
$B_i(\mu_i, \sigma_i)$	i^{th} Temporal Bin with Mean μ_i and Standard deviation σ_i
$\|B_i\|$	Size of the i^{th} Bin
$\Delta_{temporal}$	Temporal Summarization Set $\{\Delta_1, \Delta_2, \ldots, \Delta_n\}$
Γ	Set of time instants corresponding to Δ $\{\tau_1, \tau_2, \ldots, \tau_n\}$
Δ_i	2-tuple (μ_i, σ_i), the statistics of B_i
τ_i	$t_{mid}(B_i)$
S	Set of Markov States $\{S_1, S_2, \ldots, S_n\}$
S_i	i^{th} Markov State corresponding to B_i
$d(i,j)$	Distance between bins B_i and Bj, i.e., distance between Δ_i and Δ_j
$P(i \rightarrow j)$	Transition Probability from S_i to S_j
T	$n \times n$ Markov Transition Matrix
π	Markov Stationary Distribution Vector
S_{spl}	Set of split points in the temporal data
n_{final}	Number of final Temporal bins
I_N	Index Set $\{1, 2, \ldots, N\}$

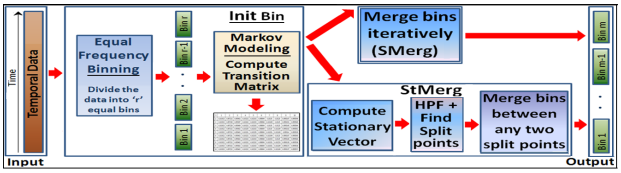

Figure 1. Temporal Neighborhood discovery

2.1 Markov Modeling

Our data is a series of temporal observations. We first formally define temporal data:

Definition 1 (Temporal data) *Sequence of data tuples $X_{temporal}$ measured at successive time instances T. A size-N dataset $X_{temporal}$ can be defined as $X_{temporal} = \{X_i | i \in I_N\}$, where $T = \{t_i | i \in I_N\}$ with $t_i < t_j$ whenever $i < j$ and $I_N = \{1, 2, .., N\}$, such that $X(t_i) = X_i$, with $X = \{X_i | t_i \in T\ , i \in I_N\}$.*

Here the temporal data is associated with a set of attributes. For instance in the case of the traffic example attribute values can be speed or count of traffic. Next we divide the data using equal frequency binning defined as follows:

Definition 2 (Equal frequency binning) *An n-parition of the temporal dataset $X_{temporal}$ into a set of equal depth temporal bins $B_{temporal} = \{B_1, B_2, \ldots B_n\}$, each bin with size $m = \lfloor \frac{N}{n} \rfloor$, such that $B_i \subset X_{temporal}$, $\cup_{i=1}^{n} B_i = X_{temporal}$ and $B_i \cap B_j = \Phi$ whenever $i \neq j$, with $|B_i| = m$, $\forall i = 1 \ldots n$.*

Each temporal bin $B_i = \{X_j | t_j \in T,\ i.m + 1 \leq j \leq (i+1)m\},\ \forall i = 1 \ldots n$. (when $N \mod n \neq 0$, last bin size $= N - (n-1)\lfloor \frac{N}{n} \rfloor$).

Given a set of temporal bins $B_{temporal} = \{B_i | i \in I_n\}$, the mean μ_i and variance σ_i^2 for all B_i are computed in order to extract a temporal summarization set, which is a size-n set of 2-tuples $\Delta_i = (\mu_i, \sigma_i^2)$, defined by, $\Delta_{temporal} = \{\Delta_i | i \in I_n\}$, where the set of time instants $\Gamma = \{\tau_i | i \in I_n\}$, with $\tau_i = t_{mid}(B_i) = \frac{1}{2}\left((i.m + 1) + (i+1)m\right)$. By definition of temporal data this summarization set also forms a temporal dataset. This temporal summarization now represents our equal frequency bins.

Now, we model the temporal summarization dataset as a Markov process. Since there is a temporal dependency between the successive bins, we consider the summarized bins as the states of a Markov model such that $S_\tau = \Delta_\tau = B_\tau(\mu_\tau, \sigma_\tau^2), \forall \tau = \tau_1 \ldots \tau_n$.

We next define a first order Markov model in the context of our approach which also explains why we can use a Markov model.

Definition 3 (Markov model) *Given temporal summarization set $\Delta_{temporal} = \{\Delta_i | i \in I_n\}$ and corresponding times $\Gamma = \{\tau_i | i \in I_n\}$, we can model a stochastic process $Y(\tau)$, $\tau = \tau_1..\tau_n$, where $Y(\tau)$ is a random variable with values $Y(\tau_i) = Y_i = \Delta_i = (\mu_i, \sigma_i^2)$, $\forall i,\ i = 1 \ldots n$ and assume the Markov property $P(Y_i = y_i | Y_{i-1} = y_{i-1}, Y_{i-2} = y_{i-2} \ldots, Y_1 = y_1) = P(Y_i = y_i | Y_{i-1} = y_{i-1})$ holds. Hence, the stochastic process $Y(\tau)$ becomes a Markov process.*

We define a first order Markov chain in which the conditional probability of any state S_i given all the previous states $S_{i-1}, \ldots, S_1$ is only dependent on the previous state S_{i-1}, i.e., $P(S_i | S_{i-1} S_{i-2} \ldots S_1) = P(S_i | S_{i-1})$ [2].

A Markov chain can be completely defined by the initial distribution and transition probability matrix. However, first we introduce the concept of similarity measured in terms of distance, which will be later used for these definitions.

We compute the similarity between the intervals using any given distance measure d. We use different distance

measures to test the efficacy of our approach for various distance measures. Specifically we use (a) Bhattacharya [4],(b)Kullback-Leibler or differential entropy [17], (c) Mahalanobis [20], (d) Hellinger [22].

2.1.1 Distance Measures

The distance measures compute the distance between two probability distributions (of two bins), using the mean and variance of the distributions. Since for normalized temporal data it is ok to assume Gaussian distribution [18], we define the distance measures formally (under Gaussian assumption) as follows:

Given two normal (Gaussian) distributions $\aleph_{0_k}(\mu_0, \Sigma_0)$ and $\aleph_{1_k}(\mu_1, \Sigma_1)$, with covariance matrices Σ_0 and Σ_1 respectively, we formally define the various distance measures in between these two distributions (from $\aleph_{0_k}(\mu_0, \Sigma_0)$ to $\aleph_{1_k}(\mu_1, \Sigma_1)$) is defined as:

Definition 4 (KL Divergence Measure) *The Kullback-Leibler divergence (differential entropy) =* $D_{KL}(\aleph_0 \| \aleph_1) = \frac{1}{2}\left(\log_e\left(\frac{\det \Sigma_1}{\det \Sigma_0}\right) + \mathrm{tr}\left(\Sigma_1^{-1}\Sigma_0\right)\right) + \frac{1}{2}\left((\mu_1-\mu_0)^\top \Sigma_1^{-1}(\mu_1-\mu_0) - k\right)$ *For* $k = 1$*, i.e., for one dimension, for any two bins* B_i *and* B_j *(such that* $B_i \sim \aleph(\mu_i, \sigma_i^2)$*,* $B_j \sim \aleph(\mu_j, \sigma_j^2)$*), the KL divergence reduces to the following,* $d(B_i, B_j) = \frac{1}{2}\left(2\log_e\left(\frac{\sigma_j}{\sigma_i}\right) + \frac{\sigma_i^2}{\sigma_j^2} + \frac{(\mu_i-\mu_j)^2}{\sigma_j^2} - 1\right)$

Definition 5 (Mahalanobis Distance Measure) *The Mahalanobis Distance Measure =* $D_{Mahalanobis}(\aleph_0 \| \aleph_1) = \left((\mu_1-\mu_0)^\top \left(\frac{\Sigma_0+\Sigma_1}{2}\right)^{-1}(\mu_1-\mu_0)\right)$*. For* $k = 1$*, this reduces to,* $d(B_i, B_j) = (\mu_i - \mu_j)^2 \frac{2}{\sigma_i^2+\sigma_j^2}$*.*

Definition 6 (Bhattacharyya Distance Measure) *The Bhattacharyya Distance Measure =* $D_{Bhattacharya}(\aleph_0 \| \aleph_1) = \frac{1}{8}\left((\mu_1-\mu_0)^\top \left(\frac{\Sigma_0+\Sigma_1}{2}\right)^{-1}(\mu_1-\mu_0)\right) + \frac{1}{2} log_e\left(\frac{|\frac{\Sigma_0+\Sigma_1}{2}|}{\sqrt{|\Sigma_0||\Sigma_1|}}\right)$ *For* $k = 1$*, the above reduces to:* $d(B_i, B_j) = \frac{1}{4}.\frac{(\mu_i-\mu_j)^2}{\sigma_i^2+\sigma_j^2} + \frac{1}{2} log_e\left(\frac{\sigma_i^2+\sigma_j^2}{2\sigma_i\sigma_j}\right)$

Definition 7 (Hellinger Distance Measure) *Hellinger Distance in between two bins* B_i *and* B_j *(such that* $B_i \sim \aleph(\mu_i, \sigma_i^2)$*,* $B_j \sim \aleph(\mu_j, \sigma_j^2)$*),* $D_{Hellinger}(B_i, B_j) = d(B_i, B_j) = \sqrt{1/2 - (\sqrt{\sigma_i\sigma_j}/2\left(\sigma_i^2+\sigma_j^2\right)e^{-1/2(\mu_i-\mu_j)^2/\sigma_i^2+\sigma_j^2}}$

We discussed a one dimensional special case here, however, our approach generalizable to multidimensional temporal data.

2.1.2 Computing Transition Probabilities

In the previous section we have demarcated equal frequency bins and considered them as the states of the Markov model. To have the complete definition of the Markov model we consider the similarities between the states to generate a transition probability matrix. We first define the initial distribution:

Definition 8 (Initial Distribution) *Given set of states* $S = \{S_1, \ldots, S_n\}$*, we assume all the states are equally likely, hence we start with the* $1 \times n$ *vector* $[\frac{1}{n}, \ldots, \frac{1}{n}]$ *as our initial distribution.*

We will define normalized transition probability shortly, but before that let us formally define the concept of temporal window (or lag) first,

Definition 9 (Temporal window) *Given a set of bins* $B_{temporal} = \{B_1, \ldots, B_n\}$*, a temporal window of size* w *for any starting bin* B_i *is defined as a set of bins* $W(B_i) = \{B_k : |k-i| \leq \lfloor \frac{w}{2} \rfloor\}$*, when* $i - \lfloor \frac{w}{2} \rfloor \geq 1$ *and* $i + \lfloor \frac{w}{2} \rfloor \leq n$*,* $\forall i$*.*

The temporal windows, within the transition probability matrix, for the bins are shown in figure 2. In this figure the greyed out areas are non zero transition values. The extreme left and extreme right windows are defined by suitably shifting the windows towards right or left respectively, thereby making its size w. We next define the transition probability between any two states on our Markov model:

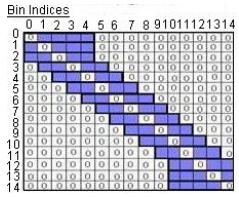

Figure 2. Transition probability matrix with Temporal window

Definition 10 (Transition Probability) *Given a set of states* $S = \{S_1, \ldots, S_n\}$*, transition probability* $P(S_j|S_i) = P(i \rightarrow j) = p_{ij} = e^{-d(S_i,S_j)}$ *iff* $|i-j| \leq \lfloor \frac{w}{2} \rfloor$*, otherwise* $p_{ij} = 0$*.*

Since the distance measures have the property that the distance between any two bins B_i, B_j, decreases with the increase in similarity between the bins, but the transition probability from one bin to another one has the property that the probability must increase with increase in similarity between the bins. A decreasing function is used to map the computed distance measures to probability space. Various decreasing functions can be used such as $\frac{1}{d(i,j)}$, $e^{-d(i,j)}$, $\frac{1}{\log(d(i,j))}$ etc. However we found that $e^{-d(i,j)}$ produces the best result, hence we use this as our decreasing function. The transition probability of each bin to every other bin is thus captured in the Markov transition probability

matrix T which is an $n \times n$ matrix. It is easy to see that $\lim_{d\to\infty} e^{-d} = 0 \leq e^{-d} \leq 1 = e^{-0}$, $\forall d \geq 0$, so that it can be thought of as a probability measure.

The Markov transition probability matrix must be at least row (or column) stochastic, that is the entries in the row or column must sum to 1. Therefore, we need to row normalize the matrix, since we want to convert the transition matrix to at least a row-stochastic one, if not a doubly stochastic matrix [26].

Definition 11 (normalized Transition probability) *Given a set of states* $S = \{S_1, \ldots, S_n\}$, *normalized transition probability* $p_{ij} = \dfrac{p_{ij}}{\sum_{|k-i|<\lfloor \frac{w}{2} \rfloor} p_{ik}}$, $\forall(i,\ j)$.

We can see from figure 2 that during transition matrix computation we need to consider only these w windows for each bin, thereby reducing the complexity of computation.

We summarize the above process in figure 3 using an example. It shows the Markov modeling for initial number of bins $= 4$ and $w = 4$, where $B_1 \ldots B_4$ are the initial (equal-frequency) bins, the dotted lines show the bin demarcations. The temporal summarization (μ, σ^2) from each bin is obtained. The Markov model is defined based on the summarized bins. Figure 3 (a) shows the state transition diagram of the Markov model, while figure 3 (b) and (c) show the initial distribution and transition probability matrix for the Markov model, which are computed from the distance measures using formula figure 3 (d).

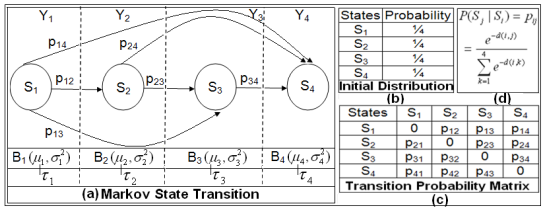

Figure 3. Example Markov Model

We outline this process of generating the Markov model in algorithm 1. In algorithm 1 on line 7 we compute the transition probability between any two states i, j which is then normalized on line 8, in order to obtain a row-stochastic matrix. It is important to note that this normalization however breaks the symmetry of the matrix. In the following merging algorithm we address this aspect.

Here d can be any one of the various distance measures discussed above and $d(i,j) = d(S_i, S_j) = d(\Delta_i, \Delta_j) = d(B_i, B_j)$. Finally on line 9 we take the average of p_{ij} and p_{ji} to compute the probability of transition in between bins B_i and B_j. Also note on line 3, n_{min} is an optional parameter to be provided by the user to avoid overmerging. The algorithm 1 has time complexity $O(N+nw)$ (lines 1-6 with complexity $O(N)$ and lines $7-8$ with complexity $O(nw)$), where N is data size and n is the number of bins and w is the window size.

Algorithm 1 InitBin: Setup Markov Model

Inputs

(1) $X_{temporal}$: Temporal data (of size N).
(2) n: Initial number of bins.
(3) w: Lag or window size.

Outputs

(1) Equal-frequency temporal bins $\{B_1, \ldots B_n\}$.
(2) Temporal summarization $\{\Delta_1, \ldots \Delta_n\}$.
(3) Markov Transition Probability matrix T (normalized).

Require: $(3 \leq w \leq n)$.

1: Normalize the temporal dataset $X_{temporal}$ using Max-Min normalization to have all values in $[0, 1]$ interval: $X_{i_{normalized}} = \frac{X_i - \min(X_{temporal})}{max(X_{temporal}) - min(X_{temporal})}$.
2: Divide the temporal data into n temporal bins $B_1,\ B_2, \ldots, B_n$ with equal frequency.
3: **if** $n \leq n_{min}$ **then**
4: exit.
5: **end if**
6: Compute statistics $\Delta_i = (\mu_i, \sigma_i^2)$, $\forall i$ (each bin), to obtain the temporal summarization $\{\Delta_1, \ldots, \Delta_n\}$.
7: Compute transition probability matrix $P = [p_{ij}]$, where $p_{ij} = \left\{ \begin{array}{cc} e^{-d(i,j)} & |i-j| \leq \lfloor \frac{w}{2} \rfloor \\ 0 & \text{otherwise} \end{array} \right\}$
8: Normalize P to obtain row-stochastic T, with $p_{ij} = \dfrac{p_{ij}}{\sum_{|i-k|\leq \lfloor \frac{w}{2} \rfloor} p_{ik}}$, where $|i-j| \leq \lfloor \frac{w}{2} \rfloor$.
9: Define symmetric transition probability between i and j as: $P(i,j) = P(i \leftrightarrow j) = \frac{1}{2}.(P(i \rightarrow j) + P(j \rightarrow i))$ $= \frac{1}{2}.(p_{ij} + p_{ji}), \forall(i,\ j)$.

2.2 Similarity based $merg$ing (SMerg)

Our aim is to identify unequal depth bins. In this approach we iteratively merge highly similar bins. At every iteration the transition matrix is recomputed since bin population has changed. The steps of this approach are outlined in the algorithm 2. Re-computation of transition probabilities (line 7) is necessary for the in-coming edges to any of the two states (bins) merged, $p_{k\mathbf{i}}$ and $p_{k\mathbf{j}}$ and also for the out-going edges from any of the two states merged, $p_{\mathbf{i}k}$ and $p_{\mathbf{j}k}$, $\forall k = 1 \cdots n$. The algorithm takes as an input the series of temporal observations t, n: Initial number of bins, n_{min}: Minimum number of bins to be output where m defaults to 2, w: Lag or window size where w defaults to r and k: Threshold factor, so that threshold for bin merging $= \lambda = \frac{k}{n-1}$, k defaults to 1. We iteratively do the following two steps until no (two) mergeable bins are left. On lines 3 and 10, we find a pair of bins having maximum probability of transition between them, which implies the highest degree of similarity and mergeability. The iterative merging from line 4-10 takes care of the transitive closure property. On line 5 based on this we merge these two bins and on line 6 we compute the statistics of this newly merged bin (mean and variance both being algebraic measure), combining the individual bins statistics. $|B_i|$ and $|B_j|$ represent the sizes of B_i and B_j respectively. We use threshold as terminating condition of the iterative algorithm. This threshold has to be carefully selected.

Threshold is computed for each iteration using the formula $\frac{k}{n-1}$, where k is a constant, typically 1, in which case threshold becomes equal to: the probability of transition from a bin to any other bin, if all the transitions are equally likely. In our experimental results we discuss results with different values of threshold, with k starting from 0.1 to 2.0 and the results are compared.

The worst-case complexity of algorithm 2 is $O(N + n^2 w)$, when all the bins are very similar in nature. Line 1 has complexity of $O(N + nw)$ and line 7 is $O(nw)$, line 3 and line 10 has complexity $O(n)$, all other lines are constant time operations, except the while loop on line 4-11 that is $(n - n_{final}) = O(n)$ in the worst case. The window size used for this algorithm is typically a constant (typically $w \leq 20$), hence the complexity is $O(N + n^2)$, where N is data size and n is the number of bins and w is the window size.

2.3 Markov Stationary distribution based $Merg$ing (StMerg)

Given the initial equal-depth bins (figure 4 (a)) and the transition matrix T, we compute the stationary distribution vector, using power-iteration method on lines 3-9. This essentially takes a self product of the transition matrix to generate the next level transition matrix. Our aim is to get a convergence of the matrix such that every row of the matrix is approximately the same producing a stationary dis-

Algorithm 2 SMerg: Generation of Unequal depth Bins using similarity based merging

Inputs

(1) $X_{temporal}$: Temporal data (of size N).
(2) n: Initial number of bins.
(3) w: Lag or window size.
(4) n_{min}: Minimum number of output bins (defaults to 2).
(5) k: Threshold factor, s.t. $\lambda = \frac{k}{n-1}$ (k defaults to 1).

Outputs

Unequal-depth temporal bins $\{B_1, \ldots B_{n_{final}}\}$.

Require: $(2 \leq n_{min} \leq n) \wedge (0.1 \leq k \leq 2) \wedge (3 \leq w \leq n)$.

Ensure: $n_{final} \geq n_{min}$.

1: Call InitBin.
2: $\lambda_{threshold} \leftarrow \frac{k}{n-1}$.
3: $(\mathbf{i}, \mathbf{j}) \leftarrow \underset{i,j}{\text{argmax}}\{P(i \leftrightarrow j) | (B_i, B_j) \text{ are adjacent}\}$.
4: **while** $(P(\mathbf{i} \leftrightarrow \mathbf{j}) > \lambda_{threshold}) \wedge (n > n_{min})$ **do**
5: Merge the bins $(B_{\mathbf{i}}, B_{\mathbf{j}})$ into a single bin and corresponding states $(S_{\mathbf{i}}, S_{\mathbf{j}})$ to a single state.
6: Compute $\Delta_{merged} \equiv (\mu_{merged}, \sigma^2_{merged})$ for the merged bin:
$\mu_{merged} \leftarrow \frac{|B_i|\mu_i + |B_j|\mu_j}{|B_i| + |B_j|}$ and
$\sigma^2_{merged} \leftarrow \frac{|B_i|(\mu_i^2 + \sigma_i^2) + |B_j|(\mu_j^2 + \sigma_j^2)}{|B_i| + |B_j|} - \mu^2_{merged}$.
7: Re-compute the transition probability matrix and renormalize.
8: $n \leftarrow n - 1$.
9: $\lambda_{threshold} \leftarrow \frac{k}{n-1}$.
10: $(\mathbf{i}, \mathbf{j}) \leftarrow \underset{i,j}{\text{argmax}}\{P(i \leftrightarrow j) | (B_i, B_j) \text{ are adjacent}\}$.
11: **end while**
12: $n_{final} \leftarrow n$.
13: Terminate and output final (merged) bins.

Algorithm 3 StMerg: Generation of Unequal depth Bins using Markov Stationary Distribution

Inputs
(1) $X_{temporal}$: Temporal data (of size N).
(2) n: Initial number of bins.
(3) i_{max}: Max^m iterations for stationary convergence.
(4) ϵ_{error}: Threshold for stationary convergence.
(5) h: Number of DFT coefficients for the high pass filter.
Outputs
(1) Unequal-depth temporal bins $\{B_1, \ldots B_{n_{final}}\}$.
(2) Stationary distribution π as approximate temporal data.
Require: $(0 \leq \epsilon_{error} \leq 1) \wedge (0 < i_{max} \leq 4)$.

1: Call InitBin.
2: $\Pi \leftarrow T$.
3: **for** $iteration = 0$ to i_{max} **do**
4: $\Pi \leftarrow \Pi \times T$.
5: To check whether Π has already been converged, compute $error(\Pi) = \sum_{j=0}^{n} |p_{i+1j} - pij|$, for any $i = 0 \ldots n-1$.
6: **if** $(error(\Pi) < \epsilon_{error})$ **then**
7: break.
8: **end if**
9: **end for**
10: **if** Π already converged **then**
11: We readily obtain the stationary distribution π from any row of Π (e.g., take 0^{th} row, $\pi \leftarrow \Pi[0]$).
12: **else**
13: take average of the rows to obtain approximate stationary distribution vector $\pi = [\pi_0 \pi_1 \ldots \pi_{n-1}]$.
14: **end if**
15: Perform high pass filter on the stationary distribution vector π using DFT, IDFT , and threshold h
16: $S_{spl} \leftarrow \Phi$.
17: **for** $i = 0$ to $n-1$ **do**
18: **if** $(\pi_i . \pi_{i+1} < 0)$ **then**
19: $S_{spl} \leftarrow S_{spl} \cup \{i\}$.
20: **end if**
21: **end for**
22: Merge all bins in between every two successive split points in S_{spl}.
23: Terminate and output final (merged) bins.

tribution (figure 4 (b)) which is formally defined as follows which is adapted from [2, 9] in the context of our approach:

Definition 12 (Stationary Distribution) *Given a finite, irreducible and aperiodic Markov chain with transition probability matrix T, then T^k converges to a rank-one matrix in which each row is called the stationary distribution π, that is, $\lim_{k\to\infty} \mathbf{T}^k = \mathbf{1}\pi$. Also there is a unique stationary distribution π. This is stated by the Perron-Frobenius theorem. [11]*

For this convergence we consider the difference in the rows of the matrix such that the error term ϵ_{error} defining this difference is minimized. However this can be very expensive and the convergence may not always be achieved in a smaller number of steps. In this case we consider another option which is to take an average of the rows of the matrix and use this as the stationary distribution vector. This is outlined on lines 10-14 in algorithm 3. Once we have this stationary distribution vector we need to detect the spikes in the vector which indicate the split points such that the data on either side of any split point is very dissimilar and the data within two split points is highly similar. This is possible since the stationary distribution provides a transitive closure such that it takes care of all possible similarities for every bin. Thus the split points indeed capture a truly similar behavior.

In order to detect these spikes in the stationary distribution vector we use Discrete Fourier Transform (DFT) and Inverse Discrete Fourier Transform (IDFT) [6] with higher h coefficients as High Pass Filter (HPF) (figure 4 (c)). For this the stationary distribution vector π is first transformed from time domain to frequency domain, using DFT. Then the spatial domain vector Π is transformed back to time domain, using IDFT, but this time use only higher h coefficients are taken (figure 4, (e), (f)). After application of HPF, on lines 17-21 the split points are found - these are the (spike) points (with sharp changes in temporal bin distribution) where there are changes of sign (from $+ive$ to $-ive$ or vice-versa). We then merge all successive equal frequency bins between any two split points to form unequal frequency bins (as shown in figure 4 (e)) resulting in the final bins (figure 4 (d)). The worst-case complexity of this algorithm is $O(N+n^3)$. This is due to the fact that matrix multiplication during power-iteration method is $O(N + n^3)$ (if Strassen's algorithm [9] is used, matrix multiplication complexity can be improved to $O(N + n^{2.807})$). Here N is data size and n is the number of bins. Here typically $w = n$.

2.4 Temporal Neighborhoods

The goal of the SMerg and StMerg algorithm is to generate temporal neighborhoods using unequal depth discretization. Essentially the merged bins discovered through the algorithms correspond to our temporal neighborhoods such

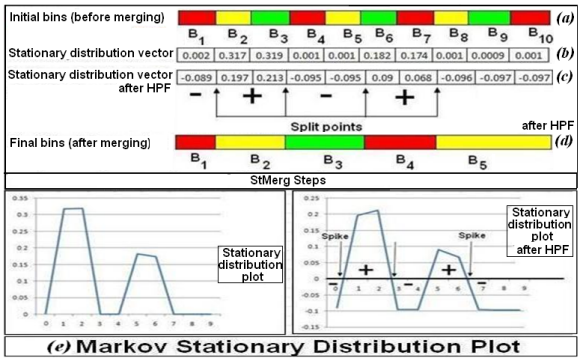

Figure 4. STMerg

that any mining task can be focused to these bins which have high intra neighborhood(or bin) similarity and high inter neighborhood (or bin) dissimilarity. We define our temporal neighborhood based on the merged bins as follows:

Definition 13 (Temporal Neighborhood) *Given a set of equal depth temporal bins* $B_{temporal} = \{B_1, B_2, \ldots B_n\}$, *Temporal Neighborhood* $NBD_{temporal} = \{NBD_1, \ldots, NBD_{n_{final}}\}$ *such that* $\cup_{i=1}^{n_{final}} NBD_i = B_{temporal}$ *and* $NBD_i \cap NBD_j = \Phi$ *whenever* $i \neq j$ *such that inter neighborhood dissimilarity is maximized and intra neighborhood similarity is maximized.*

3 Greedy Approach

We can simplify the algorithm 3 to form a greedy version without using Markov model, the one we are going to describe now.

The above algorithm is greedy in the sense that it chooses the current most similar intervals for merging and hence may stuck to some local optima instead of obtaining a globally optimal merging. Also, we must be very careful when choosing the threshold $\lambda_{threshold}$ for merging, we may use $\mu + \sigma$ limit on the similarities to choose the threshold. Note here $sim[i]$ refers to the similarity between the i^{th} and $(i+1)^{th}$ adjacent intervals, measured by $e^{-d(i,i+1)}$.
Also, from complexity perspective, we see that line 1 is $theta(N)$, where the lines $2-4$ are $\theta(n)$, finding the maximum similarity on the line 5 and line 10 can be done in $\theta(n)$, hence the iterative merging steps on the lines $6-11$ will take $\theta(n^2)$ time, the total complexity being $\theta(N+n^2)$. This approach does not necessarily ensure (global) optimality (since we do not have greedy choice property), but gives good experimental results. From next section onwards we shall focus on obtaining optimal merging.

Algorithm 4 GMerg: Merging intervals and generation of temporal neighborhoods using greedy approach

Inputs
(1) $X_{temporal}$: Temporal data (of size N).
(2) n: Initial number of bins.
(4) n_{min}: Minimum number of output bins (defaults to 2).
(5) $\lambda_{threshold}$: Threshold on similarity (defaults to 0.7).
Outputs
Unequal-depth temporal bins $\{B_1, \ldots B_{n_{final}}\}$.

1: Call InitBin. //obtain n initial equal-depth bins
2: **for** $i = 1$ to $n-1$ **do**
3: $sim[i] \leftarrow e^{-d(i,i+1)}$
4: **end for**
5: $i \leftarrow max\_element(sim)$. //find maximum similar adjacent bins
6: **while** $(n > n_{min})$ and $(sim[i] > \lambda_{threshold})$ **do**
7: Merge the bins $(B_{\mathbf{i}}, B_{\mathbf{i+1}})$
8: $n \leftarrow n-1$
9: Remove element i from the array sim.
10: $i \leftarrow max\_element(sim)$. //find maximum similar adjacent bins from the rest of the bins
11: **end while**

4 Optimal merging

4.1 Naive Approach

We can easily see that starting with n initial intervals (bins), total number of possible different ways of merging the (initial) bins = $1 + 2 + 3 + \ldots n = \frac{n(n-1)}{2} = \theta(n^2)$, as shown in the following table 4.1.

Total number of bins merged	Number of different possible merging
n	1
$n-1$	2
$\ldots$	$\ldots$
1	$n-1$

Exactly one of these $\theta(n^2)$ merging is optimal. In the naive or brute-force approach, we can compute the intra-partition-similarities and across-partition-dissimilarity for each one of these $\theta(n^2)$ partitions.

4.2 Computing the similarities

Starting with initial equal depth intervals (bins), we introduce the notion of optimal merging by defining optimal partitioning. We denote all the intervals in between i^{th} and j^{th} bin (inclusive) by $[i,j] \equiv \{B_i B_{i+1} \ldots B_j\}$. Now, in order to obtain an optimal merging of the all the intervals in $[i,j]$ we define an optimal partitioning of $[i,j]$. We define a partition on $[i,j] = [i,k] \cup [k+1,j]$, i.e., [i, j] is to be divided into two subintervals $[i,k]$ and $[k+1,j]$, where

$i \leq k < j$. Now we introduce the notion of optimality in partitioning. The partition will be optimal iff
1. all the bins (intervals) in the same side of the partition have the maximum similarity in between them
2. any two bins (intervals) belonging to different sides of the partition will have maximum dissimilarity in between them (in terms of distribution).
Now, let us recursively define the intra-partition similarity, keeping in mind that for optimal partitioning we shall want to maximize this similarity.

$$sim[i,j] = \left\{ \begin{array}{ll} 0 & i = j \\ e^{-d(i,j)} & j = i \pm 1 \\ \max\limits_{i \leq k < j} (sim[i,k] + sim[k+1,j]) & \text{otherwise} \end{array} \right\}$$

Maximizing the intra-partition intervals similarity is not sufficient, we need to maximize the across-partition dissimilarity along with it. Maximizing $d(k, k+1)$ across the partition serves this purpose. Hence, in order to obtain optimal partitioning of the intervals $[i, j]$ into subintervals $[i, k_{opt}]$ and $[k_{opt}+1, j]$, we need to find k_{opt}, such that

$$k_{opt} = \underset{i \leq k < j}{argmax} \left(\underbrace{(sim[i,k] + sim[k+1,j])}_{similarities} + \underbrace{d(k,k+1)}_{dissimilary} \right)$$

As we can see that the top-down recursive approach will be expensive in terms of time complexity since for initial bin size n, the recurrence relation $T(n)$ is given by,

$$T(n) = \left\{ \begin{array}{ll} 1 & n = 1 \\ \sum\limits_{k=1}^{n-1} T(n-k)T(k) & otherwise \end{array} \right\}$$

It can be shown that the complexity is $\Omega(4^n n^{\frac{3}{2}})$, which is the Catalan number and hence exponential. This is because similarity between any two intervals is computed multiple times and not reused. Hence, we use dynamic programming to do the optimal merging, here we use a bottom-up approach instead and reuse the similarities of subproblems already computed.

4.3 Optimal merging using Dynamic Programming

Figure 3 and algorithm 5 together explains dynamic programming algorithm. In this algorithm, as described, we generate the neighborhoods by optimal merging, starting from the initial equal-depth intervals. On line-4 we compute the initial adjacent-bin-similarity, which is inversely proportional to the distance in between the distributions of these intervals. The element $sim[i, j]$ refers to the similarity in between i^{th} and j^{th} interval, where $part[i, j]$ denotes the optimal partitioning of the interval $[i, j]$. Interval $[i, j]$ to be partitioned into 2 optimal sub-intervals $[i, k]$ and $[k + 1, j]$, such that the intra-sub-interval similarities and the inter-sub-interval dissimilarity $d(B_k, B_{k+1})$ are maximized, as shown in figure 3. On line 14, s measures intra-interval similarities of the intervals $[i, k]$ and $[k + 1, j]$, that is maximized along with the dissimilarity $d(B_k, B_{k+1})$ across the partition $[i \ldots k|k + 1 \ldots j]$.

Algorithm 5 OPTPart: Generation of optimal partitions maximizing intra-bin similarities and inter-bin dissimilarities using dynamic programming

```
1:  for i = 1 to n do
2:      sim[i, i] ← 0
3:      if (i < n) then
4:          sim[i, i + 1] ← e^{−d(B_i, B_{i+1})}
5:          part[i, i + 1] ← i
6:      end if
7:  end for
8:  for l = 2 to n do // length of a neighborhood
9:      for i = 1 to n − l + 1 do
10:         j ← i + l
11:         sim[i, j] ← −1
12:         max ← −1
13:         for k = i to j − 1 do
14:             s ← sim[i, k] + sim[k + 1, j]
15:             if (s + d(B_k, B_{k+1}) > max) then
16:                 sim[i, j] ← s
17:                 max ← s + d(B_k, B_{k+1})
18:                 part[i, j] ← k
19:             end if
20:         end for
21:     end for
22: end for
23: CreateOPTPartIntervals(part, 1, n) //create OPT partitions
24: CreateOPTPartTree(part, 1, n) //create OPTPart tree
```

4.4 Proof of Optimality

Optimal substructure property of this problem is as follows: if optimal partition of $\{B_i B_{i+1} \ldots B_j\}$ partitions the intervals in between B_k and B_{k+1}, then both the partitioning of the sub-intervals $\{B_i B_{i+1} \ldots B_k\}$ and $\{B_{k+1} B_{k+2} \ldots B_j\}$ must be optimal partitioning. We can prove this claim by simple cut and paste argument.

Indirect proof:
Let us assume to the contrary. If there were a better way (with larger intra-similarity and inter-dissimilarity) to partition $\{B_i B_{i+1} \ldots B_k\}$ (or similarly $\{B_{k+1} \ldots B_j\}$), substituting that partitioning in the optimal partitioning of $\{B_i B_{i+1} \ldots B_j\}$ would produce yet another partitioning

Algorithm 6 CreateOPTPartIntervals: Create Optimal Partition Intervals

```
1: CreateOPTPartIntervals(part, i, j)
2: if (j > i) then
3:     return
4:     '('
5:     + OPTPart(part, i, part[i, j])
6:     + ','
7:     + OPTPart(part, part[i, j] + 1, j)
8:     + ')'
9: else
10:    return B_i
11: end if
```

Algorithm 7 CreateOPTPartTree: Create Optimal partition tree

```
1: CreateOPTPartTree(part, i, j, node)
2: if node == nil then
3:     Create new node
4: end if
5: if (j > i) then
6:     node.interval ← (i, j)
7:     k ← part[i, j]
8:     node.score ← e^{-d(k,k+1)} //degree of merge-ability
9:     node.left ← CreateOPTPartTree(part, i, part[i, j], node.left)
10:    node.right ← CreateOPTPartTree(part, i, part[i, j], node.right)
11: else // if i == j
12:    node.interval ← i
13:    node.score ← 1
14: end if
15: return node
```

of $\{B_i B_{i+1} \dots B_j\}$, where the intra-interval similarities would have been more that the optimal, a contradiction.

4.5 Stopping Criterion

OPTMerg gives us the optimal merging points in the form of a binary tree, where every node represent certain interval (bin or collection of bins). While merging, every time two children nodes (smaller intervals) are merged, in order to obtain the larger parent interval. Intervals that are highly similar to each other, mostly reside in the leaf level (or the lower levels) of the binary tree and similarity decreases from bottom to top. Hence, if we go on merging every two nodes across the hierarchy, we shall eventually end up with one root node and risk ourselves by merging intervals for which the degree of similarity is not that high. Hence, we must stop merging somewhere in between. This degree of similarity across the two subintervals is chosen be the deciding factor for the decision of whether or not the subintervals $[i, k]$ and $[k+1, j]$ are to be merged. It is measured by by computing the score (6) $e^{-d(k,k+1)}$, (exponential function is chosen to make it sure that two sub-intervals with high degree of similarity will never be merged, since the score will decrease exponentially with increasing dissimilarity), where k is the split point of the interval $[i, j]$.

Algorithm 8 OPTMerg: Optimal Merging of temporal neighborhoods with stopping criterion

Inputs

(1) $X_{temporal}$: Temporal data (of size N).
(2) n: Initial number of bins.
(4) n_{min}: Minimum number of output bins (defaults to 2).
(5) $score_{threshold}$: Threshold on intra-bin-similarity and inter-bin-dissimilarity (defaults to 0.8).

Outputs

Unequal-depth temporal bins $\{B_1, \dots B_{n_{final}}\}$.

```
1: Call InitBin. //obtain n initial equal-depth bins
2: Call OPTPart. //create the OPT parttion tree
3: Do a BFS (breadth-first-search, level-order traversal) on the OPT Partition Tree:
4: for all traversed nodes of the tree do
5:     if (node.score > score_threshold) then
6:         Merge all the bins in node.interval
7:         // no need to traverse its children
8:     end if
9: end for
```

4.6 Complexity

The OPTPart is $\theta(n^3)$. This is the complexity of the matrix-chain-multiplication dynamic programming algorithm as well, that can be improved to $\theta(nlogn)$, so our dynamic programming algorithm can also be improved to $\theta(nlogn)$. Both the algorithms 6 and 7 follow the recurrence relation $T(n) = 2T(n/2) + O(1)$ which is basically $O(n)$. For the algorithm OPTMerg, line 1 is $\theta(n)$, line 2 is $\theta(n^3)$, line 3 is $\theta(nlogn)$ in the worst case, with lines $4-9$ is $\theta(n)$ in the worst case afain. Hence, the overall complexity of OPTMerg is $\theta(N + n^3)$, which again can be improved to $\theta(N + nlogn)$.

4.7 Threshold for Stopping Criterion

The threshold score to stop merging (below that score) can be kept as high as 0.8, alternatively we tried the mean scores as threshold, also, the threshold can be chosen by user externally as well.

5 Merging by SAX

We can use SAX technique to merge the bins and obtain temporal neighborhood. The following algorithm (algorithm 9) describes a straight-forward approach for doing it:

Using the SAX technique, first the temporal data C of length N, with data tuples $c_1 \dots c_N$ is represented in a w-dimensional space ($w = n$ in our case) by a vector $\bar{C} = \bar{c_1}, \dots \bar{c_n}$, (dimenionality reduction via PAA) [18]

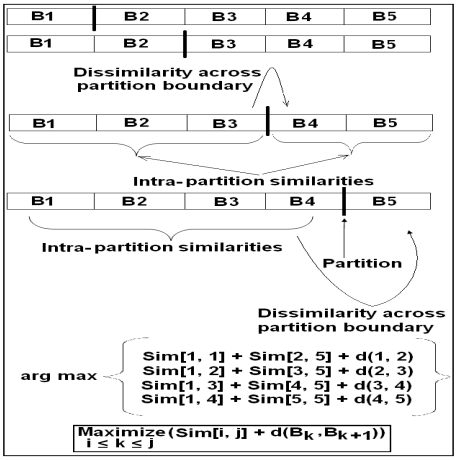

Figure 5. OPTMerg: How the Dynamic programming works

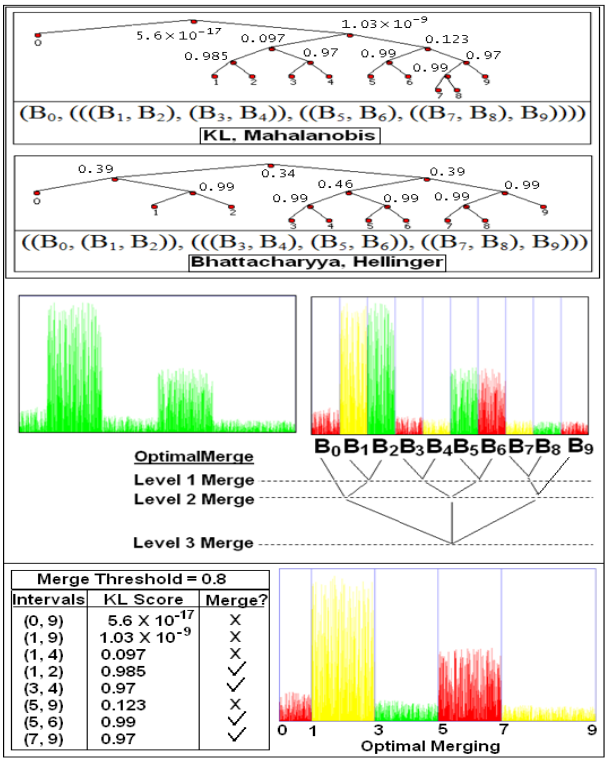

Figure 6. OPTMerg: Dynamic programming with initial bin size 10

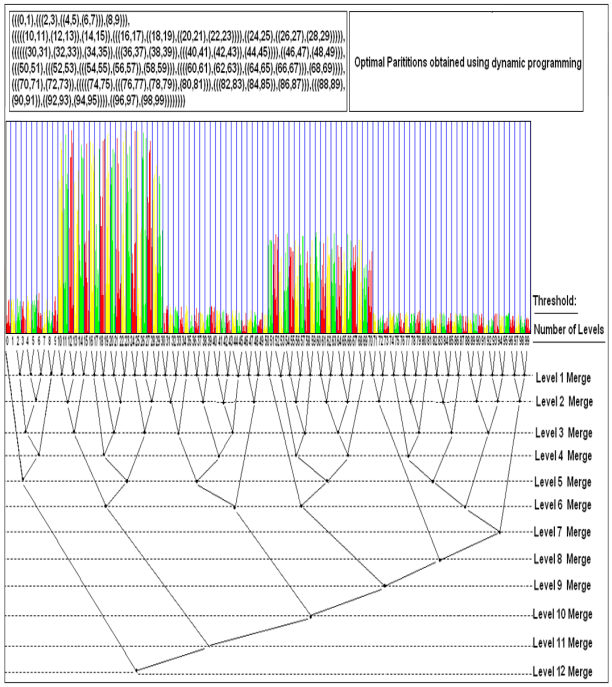

Figure 7. OPTMerg: Dynamic programming with initial bin size 100

Algorithm 9 SAXMerg: Merging similar bins after symbolic representation by SAX

Inputs

(1) $X_{temporal}$: Temporal data (of size N).
(2) n: Initial number of bins.
(4) a: Size of the alphabet set to be used.

Outputs

Unequal-depth temporal bins.

1: Call InitBin. //obtain n initial equal-depth bins
2: **for** $i = 1$ to n **do**
3: $\bar{c}_i = \frac{n}{N} \sum_{j=\frac{N}{n}(i-1)+1}^{\frac{N}{n}i} c_j.$
4: **end for**
5: Find breakpoints, a sorted list of $a-1$ numbers $B = \beta_1 \ldots \beta_{a-1}$, s.t. $\Phi(\beta_{i+1}) - \Phi(\beta_i) = \frac{1}{a}$
and discretize the bins (assign the alphabets to bins) by comparing the PAA coefficients and breakpoints.
6: Merge all adjacent bins that are assigned same alphabet.

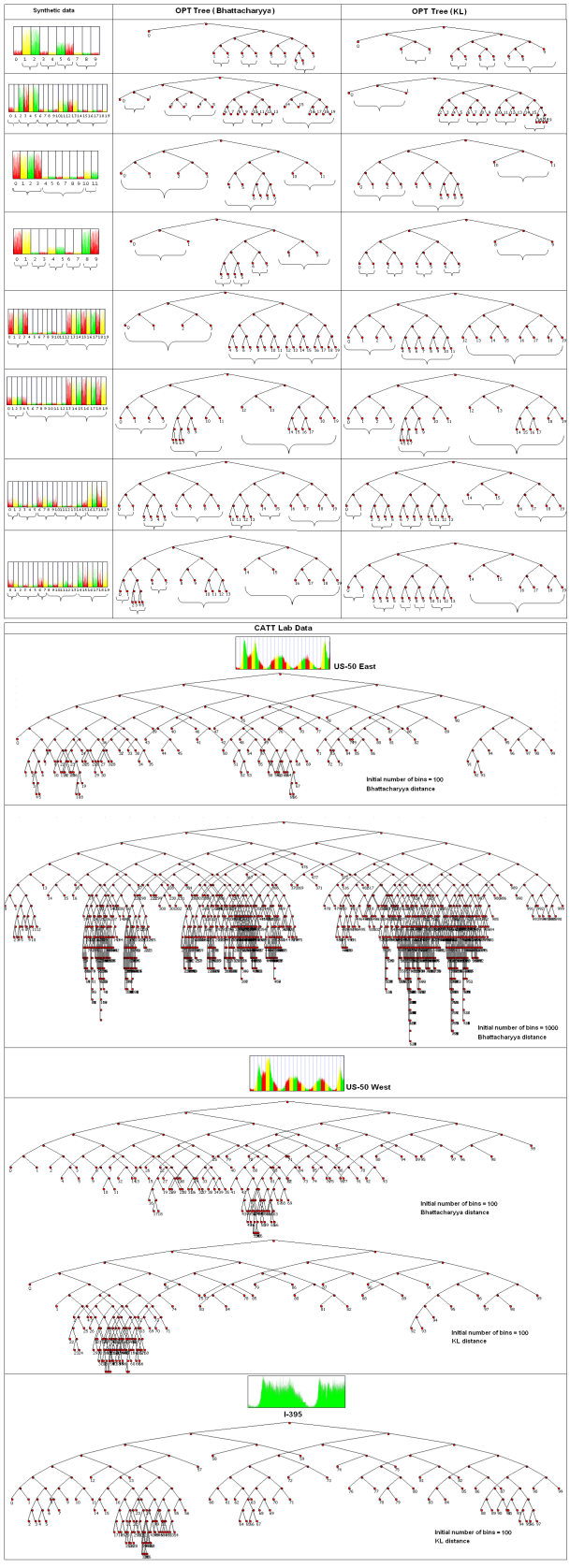

Figure 8. OPTMerg: OPT Partition Trees

with $\bar{c}_i = \frac{n}{N} \sum_{j=\frac{N}{n}(i-1)+1}^{\frac{N}{n}i} c_j$. Next, for a given a, the breakpoints are to be found and the temporal data areto be discretized into the bins (assign the alphabets to bins) by comparing the PAA coefficients and breakpoints.

β_i \ a	3	4	5	6	7	8	9	10
β_1	-0.43	-0.67	-0.84	-0.97	-1.07	-1.15	-1.22	-1.28
β_2	0.43	0	-0.25	-0.43	-0.57	-0.67	-0.76	-0.84
β_3		0.67	0.25	0	-0.18	-0.32	-0.43	-0.52
β_4			0.84	0.43	0.18	0	-0.14	-0.25
β_5				0.97	0.57	0.32	0.14	0
β_6					1.07	0.67	0.43	0.25
β_7						1.15	0.76	0.52
β_8							1.22	0.84
β_9								1.28

Figure 9. SAXMerg: Lookup table containing the breakpoints [18]

The following figure (figure 10) shows how SAXMerg works:

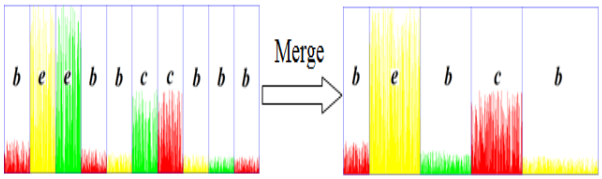

Figure 10. SAXMerg: Symobolic representation of bins by SAX (w = 10, a = 5) before and after merging

Complexity of SAXMerg is $\theta(N+n)$, i.e., computationally very efficient. As can be seen, SAXMerg is outperformed by other proposed methods in some cases and not only the recall but also the precision gets affected as shown in case of the synthetically generated Gaussian data in the following figure:

5.1 Comparison of SAXMerg with other methods

6 Experimental Results

We discuss detailed experimental results of our approach. We use synthetic and real world datasets for these results as described below:

Synthetic data : we used (a) Uniform : randomly generated by C function rand() with different seeds and , (b) Gaussian: randomly generated using Box-Muller transformation [5]. The data ranged from 1000 to 20,000,000 temporal data points. This data was generated by mixing 3 to

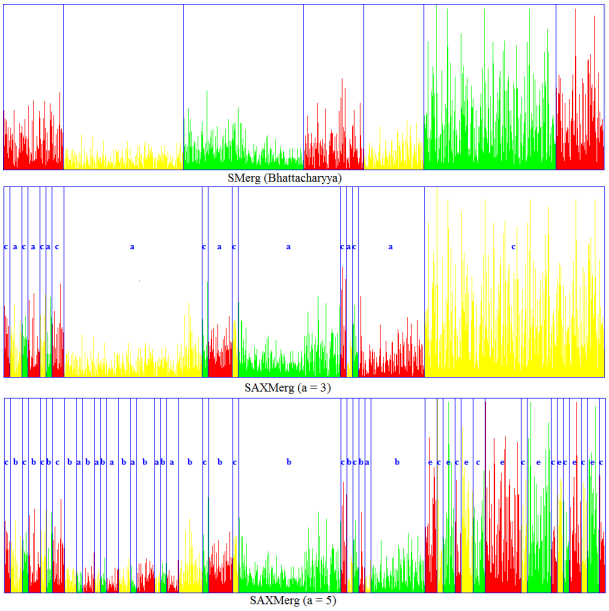

Figure 11. Comparison of SAXMerg with SMerg for synthetically generated data

6 different probability distributions leading to distinct bins, specifically Synthetic (1) has 5 distributions, Synthetic (2) has 6 distributions and Synthetic (3) has 3 distributions.

CATT Lab [1] data: is recorded by traffic sensors measuring attributes such as vehicle counts or speed. We considered 500 to 20,000 temporal data points ranging from measurements throughout a day, a week, 2 weeks, months and several months. This data is captured at every 5 minute intervals. We specifically analyzed the data from various individual sensors located along highways in Maryland specifically (a) US$-$50 West @ Church Rd West and US$-$50 East @ Church Rd East, (b) I$-$395 near Seminary Rd.

Metrics for evaluation:

For the synthetic data: the results are discussed in terms of the efficacy of discovering known bins in the synthetic data, measured using recall and precision. The known bins are labelled based on the mix of distributions used to generate the synthetic data. The precision and recall of the expected bins is depicted in each of the figures and is discussed below. We define the metrics we use to measure the accuracy of the methods:

(1) Precision $= \frac{\text{number of bins correctly merged}}{\text{total number of bins actually merged}}$ This measure reflects the numbers of bins accurately merged, hence it is very crucial for the algorithms to have this metric value as close to 1 as possible.

(2) Recall $= \frac{\text{number of bins correctly merged}}{\text{total number of bins expected to be merged}}$ This measure additionally takes care of over-splitting, i.e., reflects the numbers of bins expected to be merged but actually not merged (due to threshold etc.). This metric value should also be closer to one. For the correctness of an algorithm precision is more important and for the efficiency recall can be used.

For the CATT lab data: we did not have the labels for the expected bins however we validate the bins we found with real world *ground truth* for which we used visual analysis and heuristics as will be discussed in the subsequent sections.

6.1 Results in the CATT Lab data

US-50: In figure 5 and 6 we perform various runs of the two algorithms with various parameters. We discuss some key observations here:

(1) In figures 5 and 6 (a, b) we can see that the weekend days are clearly different from the weekdays in 5 and 6 (d, e). Further if we consider a very high level granularity such as in figure 5 and 6 (c) using algorithm 3 with the KL distance we can entirely demarcate the weekends as compared to the weekdays.

(2) On the weekend days it was observed in figure 5 and 6 (a, b) that the weekend days are further broken down such that there is a high vehicle count on the roads in the afternoons and the early mornings and evenings have relatively less traffic (time stamps shown in the x axis of the figures). This is the reverse of the weekdays traffic. In fact it was interesting to see that the weekends the traffic is very low in the mornings and picks up during the afternoon and is again low in the evenings. So this is an interesting phenomenon that during weekdays afternoons is the lowest intensity of traffic where as on weekdays afternoon is the highest traffic intensity.

(3) For the week days traffic let us consider figure 5 and 6 (e). These are demarcated at a higher level granularity combining the entire days traffic into one bin. However in figure 5 and 6 (d, f) we can see that the day is further broken down with the morning being the most traffic intensity. It is interesting to note that this pattern of high intensity is not repeated in the evening. This is because the traffic is directional therefore the traffic intensity on one sensor in the morning may be mirrored in the sensor on the opposite direction in the pm hours. This is validated by looking at the two East and west sensors in figures 5 and 6 (d). Also, we see that even from the stationary distribution obtained in StMerg algorithm (using KL distance measure) we can find the outlier (circled in the figure 5 and 6 (c)), which is weekend pattern here (that is different from weekdays) and the stationary distribution closely resembles the original trend of temporal data.

(4) It can be seen from the figures 5 and 6 (c) that the global outliers can also be detected the stationary distribution vector (marked by a circle), which represents an approximation for the temporal data.

I-395: In figure 7, temporal interval (a) and (b) obtained after merging the intervals shows that there is an anomaly

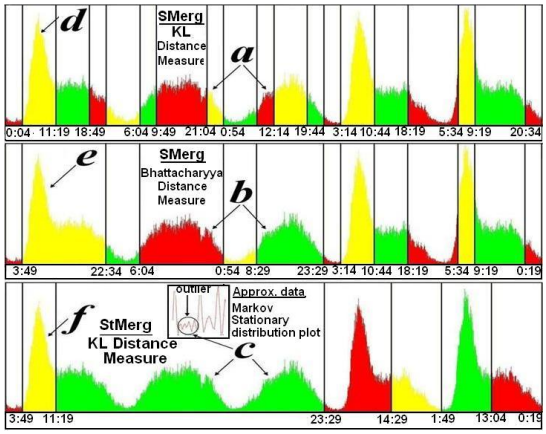

Figure 12. US−50 West @ Church Rd West Vehicle Counts @ MM 4.51 (5/1/2009 to 5/16/2009)

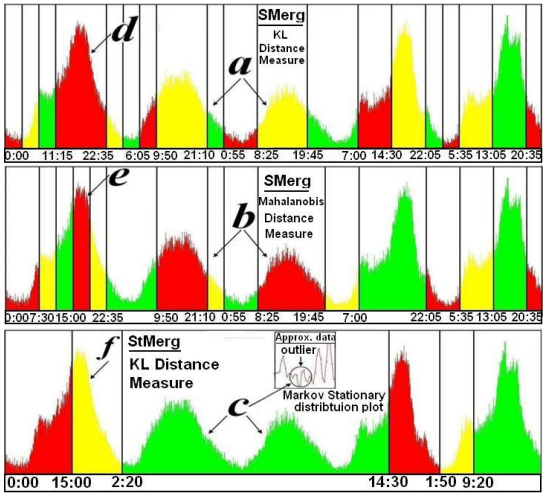

Figure 13. US−50 East @ Church Rd East Vehicle Counts @ MM 4.51 (5/1/2009 to 5/16/2009)

in the data from 05/12/2009 23 : 03 : 39 to 05/13/2009 00 03 39. We indeed find that speed value recorded by the censor is 20, constant all over the time period, and we verified from CATT lab representative that this is due to a possible "free flow speed condition" where the sensor may go into a powersave mode if the traffic intensity is very minimal or close to zero.

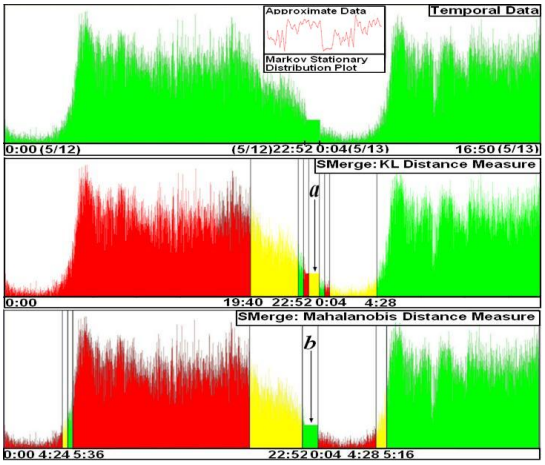

Figure 14. I−395 NEAR Seminary Rd Vehicle Counts @ MM 4.51 (5/12/2009 to 5/13/2009)

6.2 Results in the Synthetic data

The distributions of the synthetic data are known. Thus, we can use the experimental results on these synthetic data as a supervised test using precision and recall, because the ideal output bins are known here. Hence, we verified the results of our methods against them and verified the accuracy of our methods. Also, we verified that the stationary distribution obtained in StMerg can very well approximate the temporal data and can act as a compression method for the temporal data. We next discuss accuracy measures of results varying the following: (a) Number of temporal observations, (b) Initial number of bins, (c) Different Similarity measures, finally we also discuss the variation in the threshold value k for $SMerg$ and h for $StMerg$.

In each of the above we also discuss which distance metric performed the best. A series of experiments were conducted to obtain the results regarding the inter-dependency of any two of the parameters keeping others fixed.

Number of temporal observations

As it can be seen from figures 8 and 9 both $SMerg$ and $StMerg$ are scalable in terms of large data size upto 20 million observations, almost always giving precision and recall values more than 90 percent and close to 100 percent. However it was also seen that algorithm 2 in figure 8 is more scalable, since it works with equal accuracy (100 percent) for all the different similarity measures used. Algorithm 3 works most accurately for KL distance measure as shown

On Bounded Queries and Approximation

Richard Chang[*]
Department of Computer Science
University of Maryland Baltimore County
Baltimore, MD 21228
chang@cs.umbc.edu

William I. Gasarch[†]
Department of Computer Science
University of Maryland College Park
College Park, MD 20742
gasarch@cs.umd.edu

Carsten Lund
AT&T Bell Laboratories
Murray Hill, NJ 07974
lund@research.att.com

September 11, 1995

Abstract

This paper investigates the computational complexity of approximating several NP-optimization problems using the number of queries to an NP oracle as a complexity measure. The results show a trade-off between the closeness of the approximation and the number of queries required. For an approximation factor $k(n)$, $\log \log_{k(n)} n$ queries to an NP oracle can be used to approximate the maximum clique size of a graph within a factor of $k(n)$. However, this approximation cannot be achieved using fewer than $\log \log_{k(n)} n - c$ queries to *any oracle* unless P = NP, where c is a constant that does not depend on k. These results hold for approximation factors $k(n) \geq 2$ that belong to a class of functions which includes any integer constant function, $\log n$, $\log^a n$ and $n^{1/a}$. Similar results are obtained for graph coloring, set cover and other NP-optimization problems.

[*]Supported in part by NSF Research Grant CCR-9309137.
[†]Supported in part by NSF Research Grant CCR-9020079.

Figure 15. Approximate representation of temporal data using stationary distribution

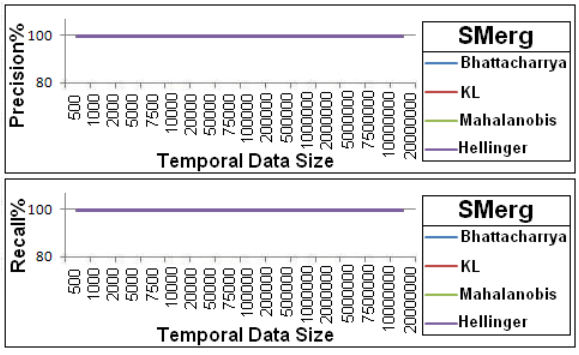

Figure 16. Precision and recall of SMerg, temporal data size increasing

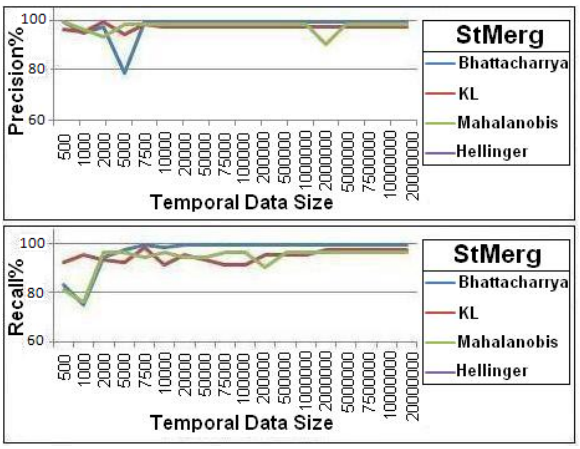

Figure 17. Precision and recall of StMerg, temporal data size increasing

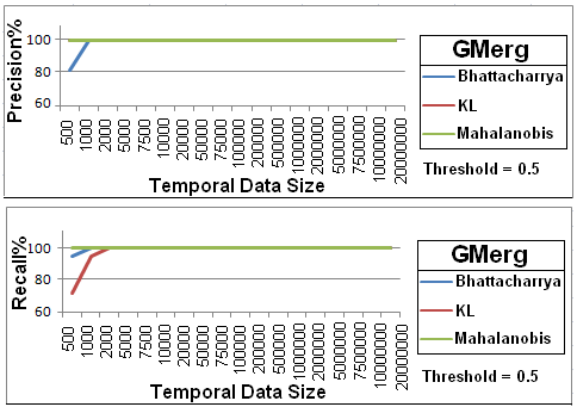

Figure 18. Precision and recall of GMerg, temporal data size increasing

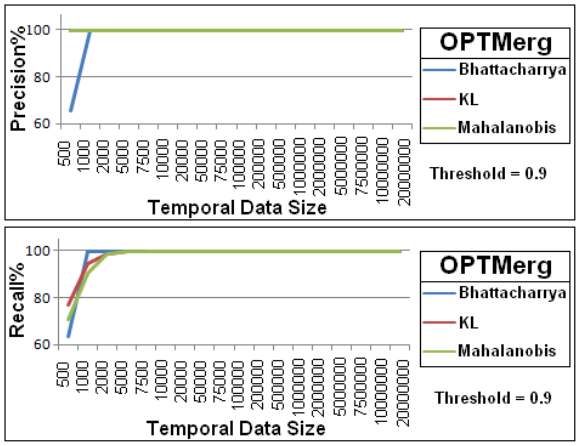

Figure 19. Precision and recall of DMerg, temporal data size increasing

in figure 9, as the most scalable and outperforms the other measures with increasing data size, in terms of precision and recall values.

Initial number of bins

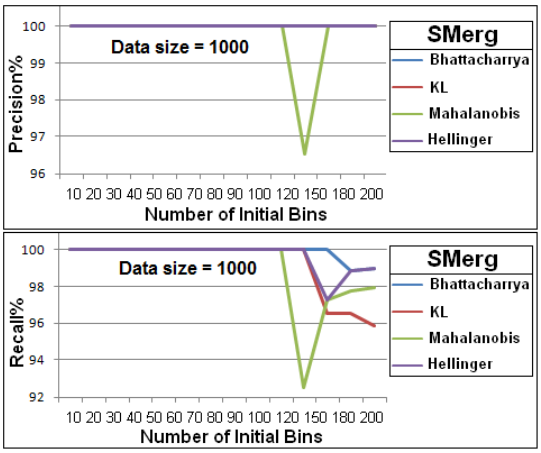

Figure 20. Precision and recall of SMerg, initial number of bins increasing

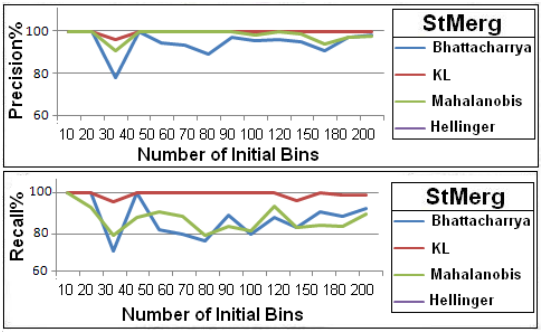

Figure 21. Precision and recall of StMerg, initial number of bins increasing

From figure 10 and 11 we can see that, the algorithm $SMerg$ is more scalable in terms of initial number of bins. For $SMerg$ and $StMerg$ KL distance measure gives the most accurate result (with precision and recall more than 90 percent most of the times).

6.3 Results in the Synthetic data with outliers

For some data the other merging techniques SAXMerg as is shown in the follwing figure (figure 28):

6.4 Comparison of SAXMerg with other methods

Discussion of results It should be noted that the algorithm performance depends upon the initial number of bins. How-

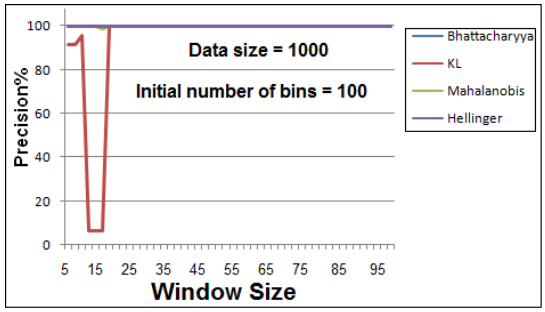

Figure 22. Precision and recall of SMerg, with change in window size

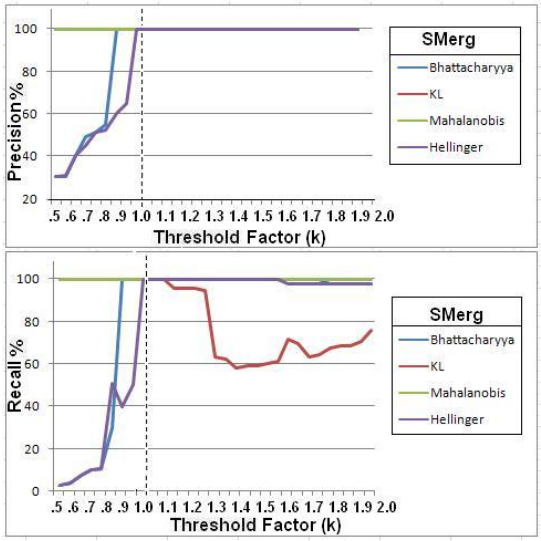

Figure 23. Precision and recall of SMerg, with change in threshold

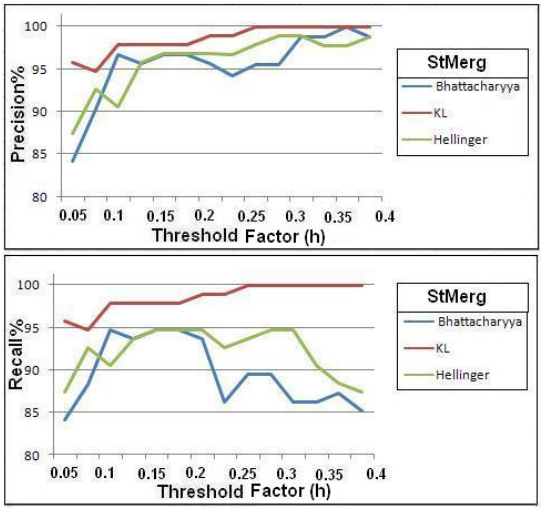

Figure 24. Impact on precision and recall of StMerg with change in threshold

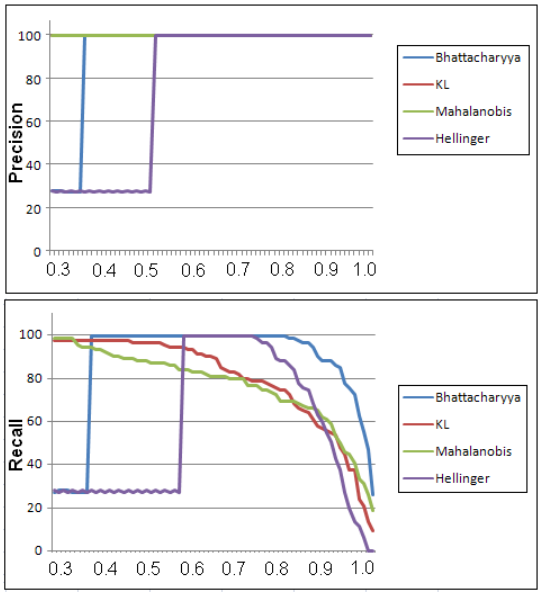

Figure 25. Impact on precision and recall of GMerg with change in threshold

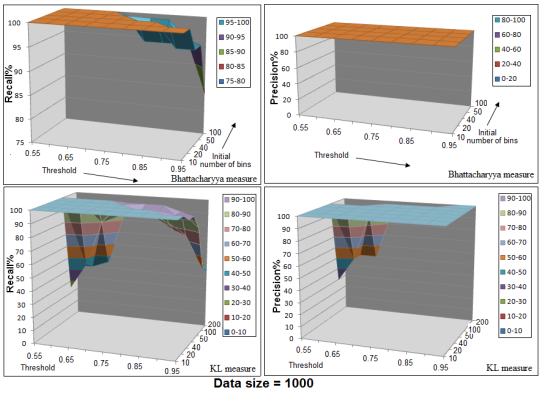

Figure 27. Precision and recall with change in number of intial bins and thresold for DMerg

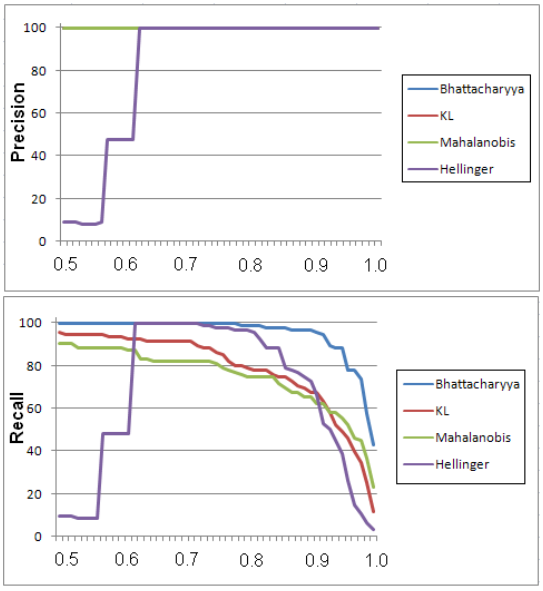

Figure 26. Impact on precision and recall of DMerg with change in threshold

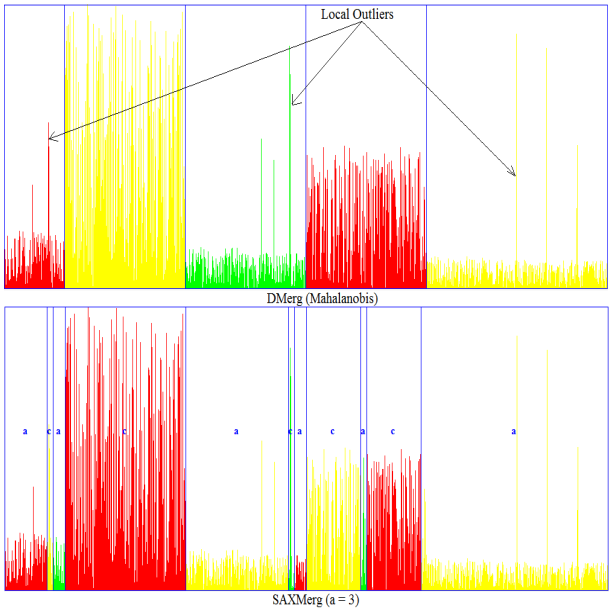

Figure 28. Comparison of SAXMerg with DMerg for synthetically generated data with (local) outliers

ever this can be mitigated by striking a balance between the initial number of bins and the threshold value. The merging of the bins is controlled by the threshold value. So say we start with an extremely large number of initial bins (say $N/2$). Then keeping a low threshold will lead to more merging which will result in a more accurate outcome even though initial number of bins is high. Essentially if initial number of bins is large then the final bins produced will grab more delicate changes in the data with respect to time, if it is small it will grab the less delicate changes. Similarly if threshold is small the final bins produced will grab more delicate changes in the data with respect to time and vice versa. For time series data that has huge change in values in a short period of time, we choose higher initial number of bins and smaller threshold, thereby increasing the complexity of the algorithms. Hence, there is a balance between initial number of bins and threshold value - typically initial number of bins in the range of $100 - 200$ (or in general 1-10% of N) and threshold $= \frac{1}{n-1}$ gives good results for CATT Lab data even of size several thousands. It is interesting to note that the $StMerg$ algorithm can be used to approximate the temporal data, using stationary Markov distribution.

In general it was seen that both $SMerg$ and $StMerg$ perform well with the KL distance metric. However $SMerg$ performs slightly better that $StMerg$. The performance of $StMerg$ can possibly be improved by using a better HPF for finding the split points. However in general both algorithms have a high recall and precision.

7 Conclusion

The techniques described in this paper to divide the temporal data into unequal depth bins to form temporal neighborhoods are not only novel in the sense that they are binning techniques based on similarity, but they also provide useful dimensionality reduction/summarization techniques for the temporal data which are typically very large. We provided both theoretical and experimental validation of the techniques proposed, for synthetically generated data as well as real-world data. Future work will extend the methods described to multi-attribute temporal data. We also intend to improve further the performance of $StMerg$ by using a better HPF for finding the split points. We would also like to compare our approach to existing temporal binning and segmentation approaches. Lastly we would like to extend this to spatio-temporal Markov models for spatial and spatio-temporal neighborhood discovery.

References

[1] Catt laboratory, center of advanced transportation technology laboratory, a research center at the university of maryland college park. (http://www.cattlab.umd.edu/).

[2] Markov chains chapter in american mathematical society's introductory probability book. (http://www.dartmouth.edu/∼chance/teaching_aids/books_articles/probability_book/Chapter11.pdf).

[3] N. R. Adam, V. P. Janeja, and V. Atluri. Neighborhood based detection of anomalies in high dimensional spatio-temporal sensor datasets. In *Proc. ACM SAC*, pages 576–583, New York, 2004.

[4] A. Bhattacharyya. On a measure of divergence between two statistical populations defined by probability distributions. *Data Min. Knowl. Discov.*, 35:99109, 1943.

[5] G. E. P. Box and M. E. Muller. A note on the generation of random normal deviates. *The Annals of Mathematical Statistics*, 29(2):610–611, June 1958.

[6] E. Brigham. *The Fast Fourier Transform*. New York: Prentice-Hall, 2002.

[7] I. Cambridge Systematics. Traffic congestion and reliability: Trends and advanced strategies for congestion mitigation. Technical report, Federal Highway Administration, U.S. Department of Transportation, September 2005. last accessed Nov 2007.

[8] C. S. Daw, C. E. A. Finney, and E. R. Tracy. A review of symbolic analysis of experimental data. *Review of Scientific Instruments*, 74(2):915–930, 2003.

[9] A. P. DI. The stationary distribution of a markov chain, la sapienza of rome, May 2005.

[10] K. E. and P. M. An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. In *4th International Conference on Knowledge Discovery and Data Mining. New York, NY*, pages 239–241, Aug 27-31 1998.

[11] R. Horn and C. Johnson. *Matrix Analysis (chapter 8)*. Cambridge University Press, 1990.

[12] K. K., G. D., and P. V. Distance measures for effective clustering of arima time-series. In *IEEE International Conference on Data Mining, San Jose, CA*, pages 273–280, Nov 29-Dec 2 2001.

[13] J. Kang and H.-S. Yong. Spatio-temporal discretization for sequential pattern mining. In *ICUIMC '08: Proceedings of the 2nd international conference on Ubiquitous information management and communication*, pages 218–224, New York, NY, USA, 2008. ACM.

[14] E. Keogh, S. Chu, D. Hart, and M. Pazzani. Segmenting time series: A survey and novel approach. *Data Mining In Time Series Databases*, 2004.

[15] E. Keogh, S. Lonardi, and B. Chiu. Finding surprising patterns in a time series database in linear time and space. In *Proceedings of The Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '02)*, Edmonton, Alberta, Canada, July 2002.

[16] H. Kling and H. Nachtnebel. Spatio-temporal discretization of catchment models for water balance modelling. *Geophysical Research Abstracts*, 7(04716), 2005.

[17] S. Kullback and R. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, page 22 (1): 7986, 1951.

[18] J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery (DMKD 2003)*, June 2003.

[19] B. Lkhagva, Y. Suzuki, and K. Kawagoe. Extended sax: Extension of symbolic aggregate approximation for financial time series data representation. In *IEICE 17th Data Engineering Workshop four times Japan Annual Conference Database, Okinawa Convention Center*, March 1-3 2006.

[20] P. C. Mahalanobis. On the generalised distance in statistics. In *Proceedings of the National Institute of Sciences of India*, page 12:4955, 1936.

[21] S. Mohammadi, V. Janeja, and A. Gangopadhyay. Discretized spatio-temporal scan window. In *The ninth SIAM International conference on Data Mining*, 2009.

[22] D. E. Pollard. *A user's guide to measure theoretic probability*. Cambridge, UK: Cambridge University Press, 2002.

[23] D. Schrank and T. Lomax. Urban mobility report. Technical report, 2007. last accessed, Nov 2007.

[24] S. Shekhar, C.-T. Lu, and P. Zhang. Detecting graph-based spatial outliers: algorithms and applications (a summary of results). In *Proc. 7th SIGKDD*, pages 371–376, New York, NY, USA, 2001. ACM Press.

[25] L. Shi and V. Janeja. Anomalous window discovery through scan statistics for linear intersecting paths (sslip). In *KDD '09: Proceeding of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009.

[26] R. Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices. *AMS*, pages 35:876–879, 1964.