Intelligent Data Analysis 18 (2014) 609–636
DOI 10.3233/IDA-140660
IOS Press

Discovery of temporal neighborhoods through discretization methods

Sandipan Dey[a,*], Vandana P. Janeja[b] and Aryya Gangopadhyay[b]
[a]*Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County, MD, USA*
[b]*Department of Information System, University of Maryland, Baltimore County, MD, USA*

Abstract. Neighborhood discovery is a precursor to knowledge discovery in complex and large datasets such as Temporal data, which is a sequence of data tuples measured at successive time instances. Hence instead of mining the entire data, we are interested in dividing the huge data into several smaller intervals of interest which we call as temporal neighborhoods. In this paper we propose a class of algorithms to generate temporal neighborhoods through unequal depth discretization.

We describe four novel algorithms (a) Similarity based Merging (*SMerg*), (b) Stationary distribution based Merging (*StMerg*), (c) Greedy Merge (*GMerg*) and, (d) Optimal Merging (*OptMerg*). The *SMerg* and *STMerg* algorithms are based on the robust framework of Markov models and the Markov Stationary distribution respectively. *GMerg* is a greedy approach and *OptMerg* algorithm is geared towards discovering optimal binning strategies for the most effective partitioning of the data into temporal neighborhoods. Both these algorithms do not use Markov models. 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 algorithms through precision and recall of labeled bins, (ii) the ground truth validation in real world traffic monitoring 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: Spatial/temporal databases, data mining, temporal neighborhoods, mining methods and algorithms

1. Introduction

In this paper we focus on discovering temporal neighborhoods by discretization of temporal data, essentially dividing the data into unequal depth bins. For a knowledge discovery task such as anomaly detection it is important to compare the data points to the most relevant points. The temporal neighborhoods or discrete intervals provide that framework [7]. 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, 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

*Corresponding author: Sandipan Dey, Epartment of Computer Science and Electrical Engineering, University of Maryland, Baltimore County, MD, USA. E-mail: sandip2@umbc.edu.

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 [6] 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 [10,22]. 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. This is equivalent to 105 million weeks of vacation and 58 fully-loaded super tankers [22].
- 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 [22].
- To compensate for congestion, companies add vehicles, hire more drivers and employees, and extend their hours to accommodate us, eventually passing through these costs to shippers and consumers. FHWA estimates increases, to the company budgets to compensate for traffic, to be between $25 and $200 per hour depending on the product carried [10].

If we analyze the traffic of any one specific 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 [6]). 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 [24] 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. We next address an important aspect of temporal dependence which is key to analysing any type of temporal data.

1.1. Temporal dependence

Two instances in time occurring one after the other, reflected as attribute values measuring a process, have a temporal relationship of adjacency. Since these instances occurred one after the other, they are possibly alike or are related in terms of their values. However, the temporal data may still behave differently even in close proximity in time. Essentially we may find pockets of variations in this data. If similarity exists in temporal data it can be assumed that data is coming from the same underlying process. However, the variability of the data may indicate that the data may be originating from multiple processes. Temporal dependence captures both these aspects, namely variation and similarity, of temporal data in demarcating the neighborhoods. Thus intra neighborhood similarity should be high and inter neighborhood dissimilarity should be high. We adapt this from the spatial data mining where Tobler's law states the spatial dependence as , every thing is related to everything else however nearby things are more related to each other. Similarly for time we can say that every temporal data point is related to everything else, but nearby temporal data points are more related to each other. We use this notion of temporal dependence in our approach.

Specifically, in this paper we make the following contributions:

- We present a novel approach to generate temporal neighborhoods 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 present a stationary distribution of Markov Model to approximate the original data keeping the properties of the temporal data intact. We present a Greedy approach which attains a local optimum in the binning and an optimal binning strategy which, attains a global optimum to find optimal binning.
- All the algorithms for our approach 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 discuss the related work. In Section 3 we outline our approach. In Section 4 we discuss the experimental results. Finally we conclude in Section 5.

2. Related work

Here we discuss related works in histogram binning, clustering and symbolic representation, which are relevant to our work. As mentioned in [14,27], histograms depend on the choice of the number of bins and the bin width, thus, selecting optimal number of bins or alternatively bin width selection, becomes a key factor in any histogram based analysis. However, there has been very little research into estimation of the optimal bin width [27]. In general, most histogram approaches select sufficiently large number of bins to capture the major features in the data while ignoring fluctuations due to random sampling. Several rules of thumb exist for determining the number of bins [14]. Such methods tend to be adhoc and may result in sub-optimal binning. Researchers have proposed some variants for discovering the optimal bin width [11,21,23,26]. Most of these approaches are based on minimizing errors of the histogram model.

However, as pointed out in [14], since the underlying density is not known, it is not reasonable to use an optimization criterion that relies on the error between the histogram density model and the true density. Instead, Knuth et al. [14] considered the histogram to be a piecewise-constant model of the underlying probability density. The results from the algorithms proposed [14] outperform several well-accepted rules for choosing bin sizes. However, since they perform a brute force search of the number of bins the performance is affected for large data. In general, equal frequency bins can be inefficient in describing multi-modal density functions where variable bin-width models can be more useful.

Similarity based techniques have been proposed for symbolic representation which are used to generate a representation of temporal data. [16] 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, this technique does not check the similarity in between adjacent bins before assigning symbols to the individual bins. It essentially computes the interval means by PAA and find break points in the attribute range. Although the SAX compression uses RLE (image processing) like technique to represent all the adjacent bins that are assigned the same symbol, it still does so after the symbol assignment and not before that. Hence, SAX technique misses the measures of dispersion information in between adjacent bins so two adjacent bins that are assigned same symbol by SAX technique may be not similar (despite their mean being same), so if the SAX technique is used to merge the bins for temporal neighborhood discovery, then it may sometimes merge two bins incorrectly. Extended SAX technique [17] (ESAX) keeps 3-tuple (min, max, mean) for each interval in order and represents each bin by 3-symbols. However standard deviation is a much better dispersion technique for measuring dispersion than min, max (which are more affected by extreme values or noise), if we design a merging using ESAX technique to merge adjacent bins, it may again merge adjacent bins incorrectly as in SAX technique.

In addition [9,13] discuss clustering temporal data. [19] proposes a new method for unsupervised discretization of univariate time series data by taking the temporal order of values into account unlike the pervious approaches. 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. [7] proposes temporal neighborhood discovery methods, however this approach does not identify optimal divisions in the data in presence of noise. Thus, the discretization should go beyond an approximation and be robust in the presence of noise. In this paper we address these issues to propose a class of algorithms for temporal neighborhood discovery using Markov model based discretization technique and Dynamic programming to find optimal binning strategies.

3. Approach

In this paper we propose an approach to generate temporal neighborhoods by discretizing temporal data into unequal depth bins. Specifically we propose a series of algorithms with a goal is to produce an unequal depth binning on a series of temporal observations. We describe this process in the following distinct steps as outlined in Fig. 1:

(a) *Equal Frequency binning*: Given a series of temporal observations we first begin with an equal frequency binning to divide data into initial equal depth bins. This simple binning serves as an input to the rest of the steps of our approach. We also define various distance measures which can be used to find the similarities between these bins.

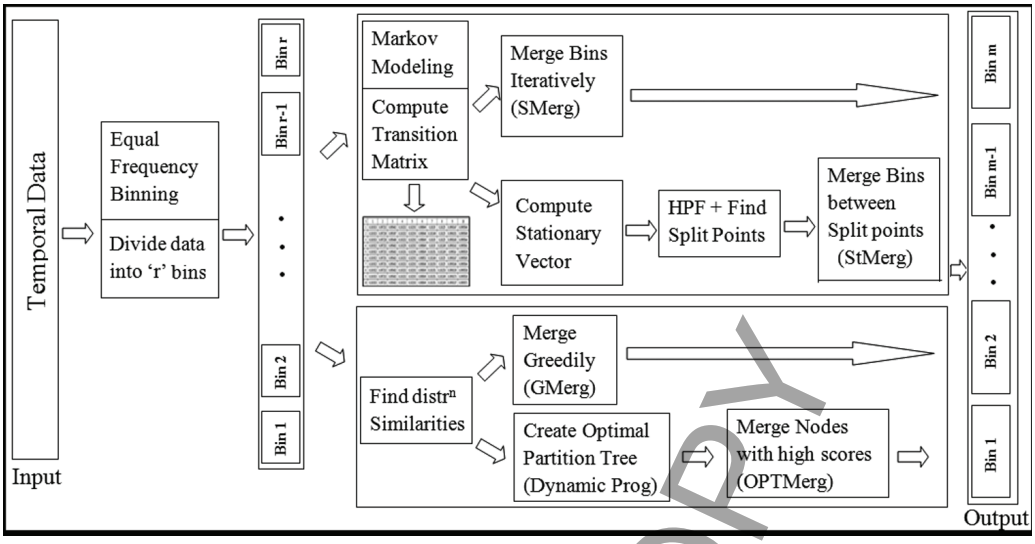

Fig. 1. Temporal neighborhood discovery.

(b) *Markov Modeling*: We consider the equal frequency 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 3.1.1. We then generate a transition matrix based on these similarities for this Markov model and subsequently normalize to obtain a row-stochastic matrix.

(c) *Unequal depth discretization using Markov Models*: In order to form an unequal depth discretization we propose two solutions using Markov Models. 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 Similarity based $merg$ing (*SMerg*) and a Markov Stationary distribution based $merg$ing (*StMerg*).

(d) *Unequal depth discretization without using Markov Models*: Additionally we propose two solutions without using Markov Models focusing on generating an optimal binning. We propose a greedy merging algorithm *GMerg* which attains a local optimum and an optimal merging *OptMerg* algorithm based on dynamic programming which attains a global optimum.

Figure 1 presents our approach using the various algorithms. We next describe our approach in details.

3.1. Equal frequency binning

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 $I_N = \{1, 2, \ldots, N\}$, 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-partition 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 $B_i = \{X_j | t_j \in T,\ i.m + 1 \leqslant j \leqslant (i+1)m\}, \forall i = 1 \ldots n$. (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 $\tau_i = t_{mid}(B_i) = \frac{1}{2}((i.m+1) + (i+1)m)$. By definition of temporal data this summarization set also forms a temporal dataset. This temporal summarization now represents our equal frequency bins.

3.1.1. 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 [16], 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 3 (*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 4 (Mahalanobis Distance Measure). The Mahalanobis Distance Measure = $D_{\text{Mahalanobis}}(\aleph_0 \| \aleph_1) = ((\mu_1 - \mu_0)^\top \left(\frac{\Sigma_0 + \Sigma_1}{2}\right)^{-1}(\mu_1 - \mu_0))$. 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 5 (Bhattacharyya Distance Measure). The Bhattacharyya Distance Measure = $D_{\text{Bhattacharya}}(\aleph_0 \| \aleph_1) = \frac{1}{8}\left((\mu_1 - \mu_0)^\top \left(\frac{\Sigma_0 + \Sigma_1}{2}\right)^{-1}(\mu_1 - \mu_0)\right) + \frac{1}{2}\log_e\left(\frac{|\frac{\Sigma_0 + \Sigma_1}{2}|}{\sqrt{|\Sigma_0||\Sigma_1|}}\right)$ 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 6 (Hellinger Distance Measure). Hellinger Distance in between two bins B_i and B_j (such that $B_i \sim \aleph(\mu_i,\ \sigma_i^2)$, $B_j \sim \aleph(\mu_j,\ \sigma_j^2)$), $D_{\text{Hellinger}}(B_i, B_j) = d(B_i, B_j) = \sqrt{1/2 - (\sqrt{\sigma_i\sigma_j/2\left(\sigma_i^2 + \sigma_j^2\right)}e^{-1/2(\mu_i - \mu_j)^2/\sigma_i^2 + \sigma_j^2}}$.

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

3.2. Markov modeling

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 7 (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 $Y(\tau)$, $\tau = \tau_1 \ldots \tau_n$, where $Y(\tau)$

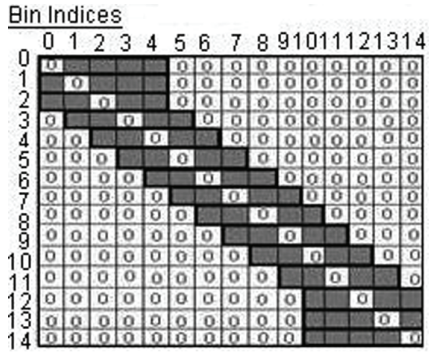

Fig. 2. Transition probability matrix with Temporal window.

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 [3], (b) Kullback-Leibler or differential entropy [15], (c) Mahalanobis [18], (d) Hellinger [20].

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 $W(B_i) = \{B_k : |k - i| \leqslant \lfloor \frac{w}{2} \rfloor\}$, when $i - \lfloor \frac{w}{2} \rfloor \geqslant 1$ and $i + \lfloor \frac{w}{2} \rfloor \leqslant n, \forall i$.

The temporal windows, within the transition probability matrix, for the bins are shown in Fig. 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:

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

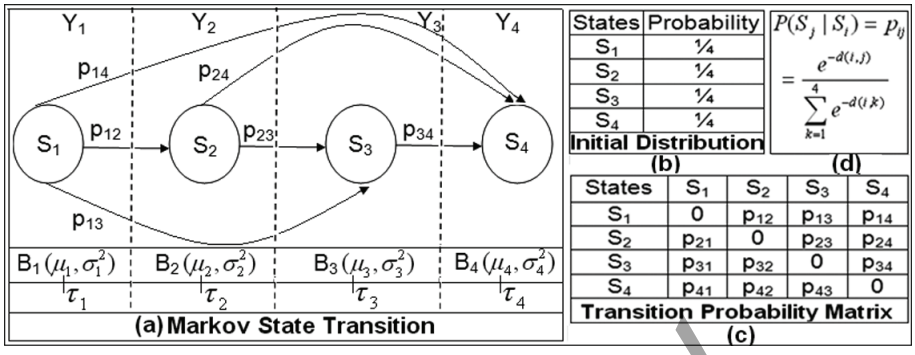

Fig. 3. Example markov model.

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 $\lim_{d\to\infty} e^{-d} = 0 \leqslant e^{-d} \leqslant 1 = e^{-0}, \forall d \geqslant 0$, 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 [25].

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

We can see from Fig. 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 Fig. 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 Figs 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 Fig. 3(d).

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: $(3 \leqslant w \leqslant n)$.

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** $n \leqslant n_{\min}$ **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 $p_{ij} = \left\{ \begin{array}{cc} e^{-d(i,j)} & |i-j| \leqslant \lfloor \frac{w}{2} \rfloor \\ 0 & \text{otherwise} \end{array} \right\}$
8: Normalize P to obtain row-stochastic T, with $p_{ij} = \frac{p_{ij}}{\sum\limits_{|i-k| \leqslant \lfloor \frac{w}{2} \rfloor} p_{ik}}$, where $|i-j| \leqslant \lfloor \frac{w}{2} \rfloor$.
9: Define symmetric transition probability between i and j as: $P(i,j) = P(i \leftrightarrow j) = \frac{1}{2}.(P(i \rightarrow j) + P(j \rightarrow i)) = \frac{1}{2}.(p_{ij} + p_{ji}), \forall (i,\ j)$.

3.3. Unequal depth discretization using Markov models

3.3.1. Similarity based merging (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 $p_{\mathbf{j}k}$, $\forall k = 1 \ldots n$. 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.

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

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

1: Call InitBin.
2: $\lambda_{threshold} \leftarrow \frac{k}{n-1}$.
3: $(\mathbf{i}, \mathbf{j}) \leftarrow \underset{i,j}{\operatorname{argmax}}\{P(i \leftrightarrow j) | (B_i, B_j) \text{ are adjacent}\}$.
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 re-normalize.
8: $n \leftarrow n - 1$.
9: $\lambda_{threshold} \leftarrow \frac{k}{n-1}$.
10: $(\mathbf{i}, \mathbf{j}) \leftarrow \underset{i,j}{\operatorname{argmax}}\{P(i \leftrightarrow j) | (B_i, B_j) \text{ are adjacent}\}$.
11: **end while**
12: $n_{final} \leftarrow n$.
13: Terminate and output final (merged) bins.

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 $w \leqslant 20$), 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.

3.3.2. Markov Stationary distribution based Merging (StMerg)

Given the initial equal-depth bins (Fig. 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 distribu-

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: $(0 \leqslant \epsilon_{error} \leqslant 1) \wedge (0 < i_{\max} \leqslant 4)$.

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

tion (Fig. 4(b)) which is formally defined as follows which is adapted from [2,8] 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, $\lim_{k\to\infty} \mathbf{T}^k = \mathbf{1}\pi$. This is stated by the Perron-Frobenius theorem [12].

Since every Markov state is reachable from every other state (since every bin has nonzero transition probability for the adjacent bins for a length w window), the transition matrix is irreducible. Essentially

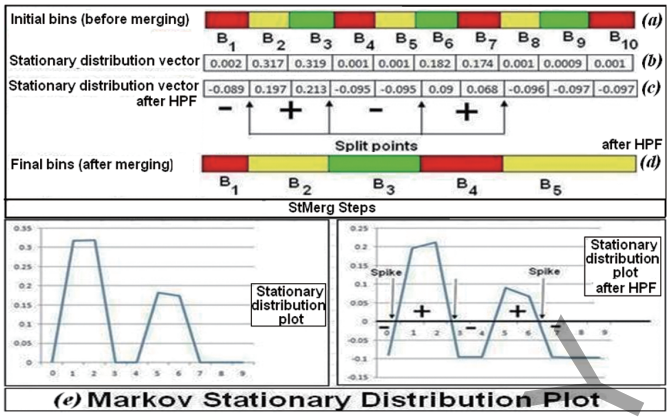

Fig. 4. STMerg. (Colours are visible in the online version of the article; http://dx.doi.org/10.3233/IDA-140660)

all states are reachable from every other state which makes all the probabilities in the transition matrix strictly positive at some iteration of the transition matrix.

For the convergence we consider the difference in the rows of the matrix such that the error term ϵ_{error} defining this difference is minimized. However if the convergence can not be achieved in a small number of steps 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) [5] with higher h coefficients as High Pass Filter (HPF) (Fig. 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 (Figs 4(e) and (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 Fig. 4(e)) resulting in the final bins (Fig. 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 [8] 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 $w=n$.

3.4. Unequal depth discretization for optimal binning

In performing discretization for temporal neighborhood generation we create a binning which can be used for subsequent knowledge discovery. However we would like to identify the best binning possible or the optimal binning. We next discuss a greedy approach and an approach based on dynamic programming to discover an optimal binning such that the inter bin dissimilarity and intra bin similarity is maximized.

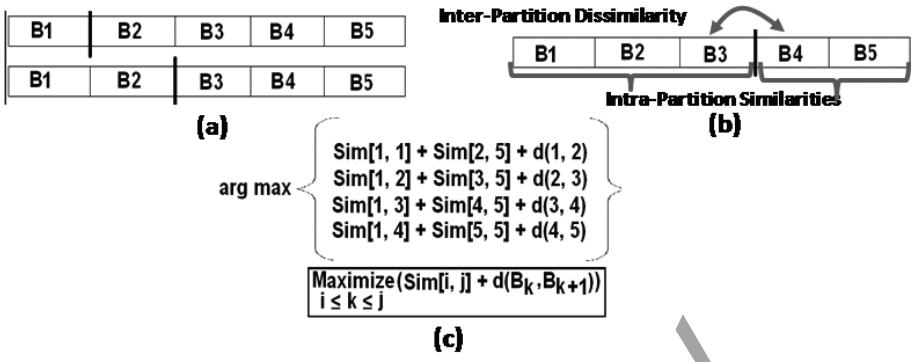

Fig. 5. Optimal binning.

3.4.1. Greedy approach

We can simplify the Algorithm 3 to form a greedy version without using a Markov model. The Algorithm 4 outlines this process. This algorithm is greedy in the sense that it chooses the current most similar intervals for merging and hence may find a local optimum instead of obtaining a globally optimal merging. Also, we must be careful when choosing the threshold $\lambda_{threshold}$ for merging, we may use $\mu+3\sigma$ limit on the similarities to choose the threshold. Note here $sim[i]$ refers to the similarity between the i^{th} and $(i+1)^{\text{th}}$ 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 as discussed in Section 4. Next we focus on obtaining optimal merging which ensures a global optimum.

3.4.2. Optimal merging: Naive approach

In demarcating the neighborhoods we aim to create optimal binning such that the merging of the initial bins is optimal because it always merges any two adjacent bins iff they are very similar in nature (with the notion of similarity defined by our distance measures) . Thus the aim is for the similarity to be maximized.

We first consider a naive approach to finding optimal binning. 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.

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.

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 $\{B_1, \ldots B_{n_{final}}\}$.

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

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 four subintervals, two of these subintervals are shown in Fig. 5(a), $[i, k]$ and $[k+1, j]$, where $i \leqslant k < j$. 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 , as shown in Fig. 5(b), keeping in mind that for optimal partitioning we want to maximize this similarity.

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

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}, as shown in Fig. 5(c), such that

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

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,

$$T(n) = \begin{cases} 1 & n = 1 \\ \sum_{k=1}^{n-1} T(n-k)T(k) & \text{otherwise} \end{cases}$$

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.

3.4.3. Optimal merging using dynamic programming

Figure 5 and Algorithm 5 outlines the intuition on how we utilize dynamic programming to find optimal binning. In Algorithm 5, 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 Fig. 5(b). 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 also creates optimal partitions and calls the procedure to create an optimal partitioning tree, outlined in Algorithm 6. Essentially the optimal partitioning tree considers all possible partitions. Each node in the opt-part-tree is assigned a score. If the node represents interval $[i \ldots j]$ and the optimal partition index is $k (i \leqslant k < j$, obtained using dynamic programming), the left-child of the node will represent intervals $[i \ldots k]$ and right child will represent intervals $[k+1 \ldots j]$, with the score for the node defined by how similar are the intervals $[i \ldots k]$ and $[k+1 \ldots j]$. If they are sufficiently similar (above a threshold), they will be merged to interval $[i \ldots j]$. Two nodes will be merged iff their parent has score above the mergability threshold. This algorithm basically enables a step by step "controlled" merging. Since the lower level nodes tend to be more similar than upper level nodes, one can start with the lowest level, merge the nodes with score greater than threshold value.

The overall process of generating the optimal merging for generation of the temporal neighborhoods is outlined in Algorithm 7, where a breadth first search is performed on the tree, if the score is greater than the threshold then the nodes falling below the threshold are merged. The entire process is depicted in Fig. 6. The figure shows the optimal partition tree created using *KL*, Mahalanobis, Bhattacharya and Hellinger distances. The initial bin size is 10. The score threshold is set to 0.8, so essentially the bins in the node interval will be merged only if the intra bin similarity is greater than 0.8. In this case we do not need to traverse the child nodes. As an example in Fig. 6 interval 0.9 has a score of 5.6×10^{-17} which is less than the merge threshold of 0.8 so no merge takes place. However for intervals 1.2 the score is 0.985 which is greater than the merge threshold of 0.8 so a merge takes place and so on.

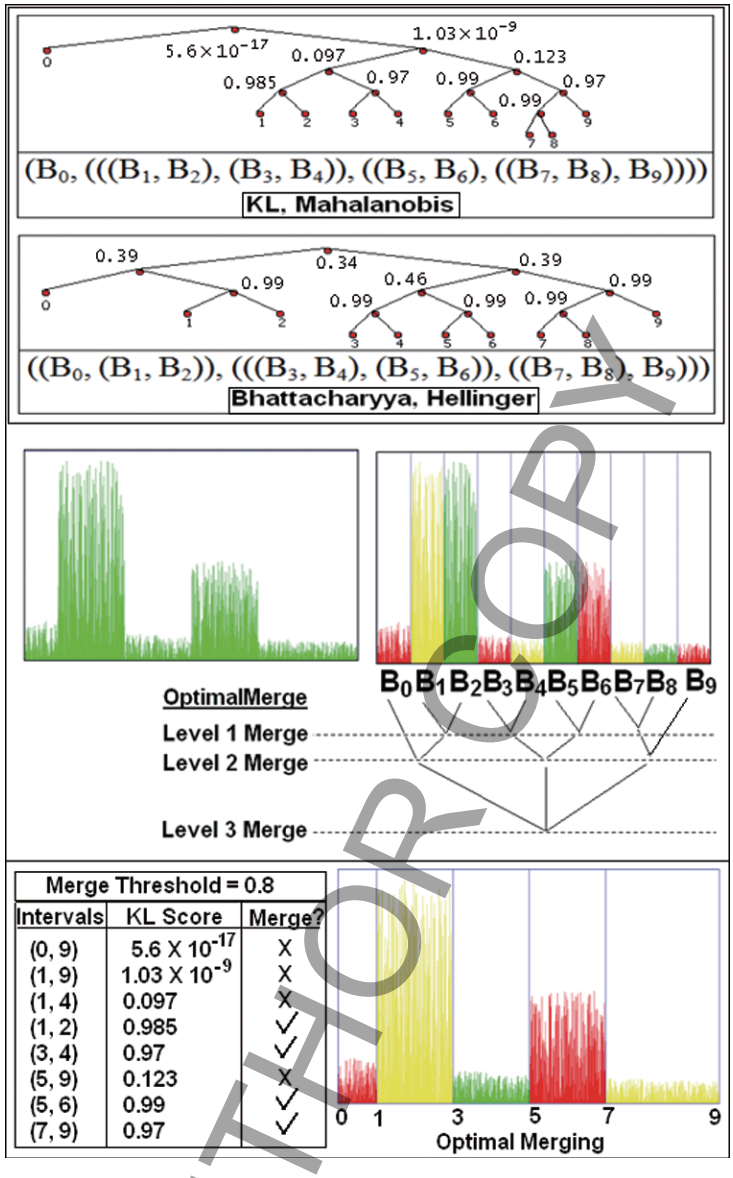

Fig. 6. OPTMerg: Dynamic programming with initial bin size 10. (Colours are visible in the online version of the article; http://dx.doi.org/10.3233/IDA-140660)

3.5. *Discussion of OptMerg*

We next discuss important properties of the *OptMerg* algorithm.

Proof of optimality: An Optimal substructure property of problem holds if an optimal solution can be constructed from optimal solutions to its subproblems. We show the optimal substructure property of this problem as follows: if optimal partition of $\{B_i B_{i+1} \ldots B_j\}$ generates a partitioning of 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 contradiction. Let us assume to the contrary. If there were a better way (with larger intra-similarity and inter-dissimilarity)

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

for $i = 1$ to n **do**
  $sim[i, i] \leftarrow 0$
  if $(i < n)$ **then**
    $sim[i, i+1] \leftarrow e^{-d(B_i, B_{i+1})}$
    $part[i, i+1] \leftarrow i$
  end if
end for
for $l = 2$ to n **do** { length of a neighborhood}
  for $i = 1$ to $n - l + 1$ **do**
    $j \leftarrow i + l$
    $sim[i, j] \leftarrow -1$
    $\max \leftarrow -1$
    for $k = i$ to $j - 1$ **do**
      $s \leftarrow sim[i, k] + sim[k + 1, j]$
      if $(s + d(B_k, B_{k+1}) > \max)$ **then**
        $sim[i, j] \leftarrow s$
        $\max \leftarrow s + d(B_k, B_{k+1})$
        $part[i, j] \leftarrow k$
      end if
    end for
  end for
end for
create OPT partitions
CreateOPTPartTree(part, 1, n) {create OPTPart tree}

Algorithm 6 CreateOPTPartTree: Create Optimal partition tree

CreateOPTPartTree(part, i, j, node)
if $node == nil$ **then**
  Create new node
end if
if $(j > i)$ **then**
  node.interval $\leftarrow (i, j)$
  $k \leftarrow part[i, j]$
  node.score $\leftarrow e^{-d(k,k+1)}$ {degree of merge-ability}
  node.left $\leftarrow$ CreateOPTPartTree(part, i, part[i, j], node.left)
  node.right $\leftarrow$ CreateOPTPartTree(part, i, part[i, j], node.right)
else { if $i == j$ }
  node.interval $\leftarrow$ i
  node.score $\leftarrow 1$
end if
return node

to partition $\{B_iB_{i+1}\dots B_k\}$ (or similarly $\{B_{k+1}\dots B_j\}$), substituting that partitioning in the optimal partitioning of $\{B_iB_{i+1}\dots B_j\}$ would produce yet another partitioning of $\{B_iB_{i+1}\dots B_j\}$, where the intra-interval similarities would have been more that the optimal, a contradiction, thus substantiating our proof of optimality.

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 child 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 will eventually end up with one root node and risk 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 to 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 computing the score $e^{-d(k,k+1)}$, where k is the split point of the interval $[i, j]$. Here an exponential function is chosen to make sure that two sub-intervals with high degree of similarity will never be merged, since the score will decrease exponentially with increasing dissimilarity.

Algorithm 7 *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 partition 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
```

Complexity: The complexity of Algorithm 5, OPTPart is $\theta(n^3)$. This is the complexity of the matrix-chain-multiplication dynamic programming algorithm as well, that can be improved to $\theta(n \log n)$, so our dynamic programming algorithm can also be improved to $\theta(n \log n)$. Both the Algorithms 5 and 6 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 $\theta(n \log n)$ in the worst case, with lines 4–9 is $\theta(n)$ in the worst case. Hence, the overall complexity of *OPTMerg* is $\theta(N + n^3)$, which again can be improved to $\theta(N + n \log n)$.

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.

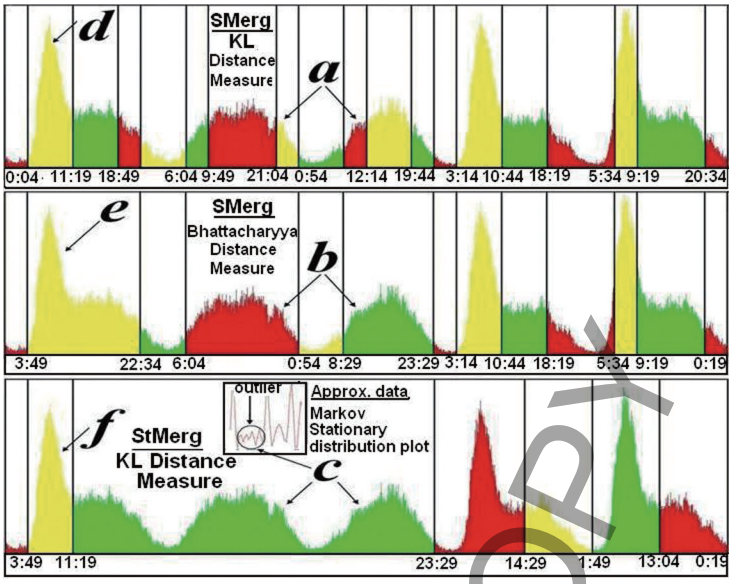

Fig. 7. US−50 West @ Church Rd West Vehicle Counts @ MM 4.51 (5/1/2009 to 5/16/2009). (Colours are visible in the online version of the article; http://dx.doi.org/10.3233/IDA-140660)

3.6. Temporal neighborhoods

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

4. Experimental results

We discuss detailed experimental results of our approach as follows:

- Results of algorithms using Markov Models namely *Smerg* and *STMerg*.
- Results for Optimal binning algorithms *GMerg* and *OPTMerg*.
- Comparative results across alorithms proposed.
- Comparing with existing approach [16].

4.1. Datasets

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 [4]. The data ranged from

1000 to 20,000,000 temporal data points. This data was generated by mixing 3 to 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 labeled 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) $\text{Precision} = \frac{\text{number of bins correctly merged}}{\text{total number of bins actually merged}}$ 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) $\text{Recall} = \frac{\text{number of bins correctly merged}}{\text{total number of bins expected to be merged}}$ 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. We next discuss the results in detail. We break up the results by discussing the results of algorithms using Markov Models namely *Smerg* and *STMerg* and results for Optimal binning algorithms *GMerg* and *OPTMerg* which do not use Markov Models as a base, subsequently we discuss overall observations and comparative results.

4.2. Results with SMerg and STMerg

4.2.1. Results in the CATT Lab data

US-50: We perform various runs of the two algorithms with various parameters. We discuss some key observations here:

(1) In Figs 7(a,b) and 8 we can see that the weekend days are clearly different from the weekdays in Figs 7(d, e) and 8. Further if we consider a very high level granularity such as in Figs 7(c) and 8 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 Figs 7(a,b) and 8 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.

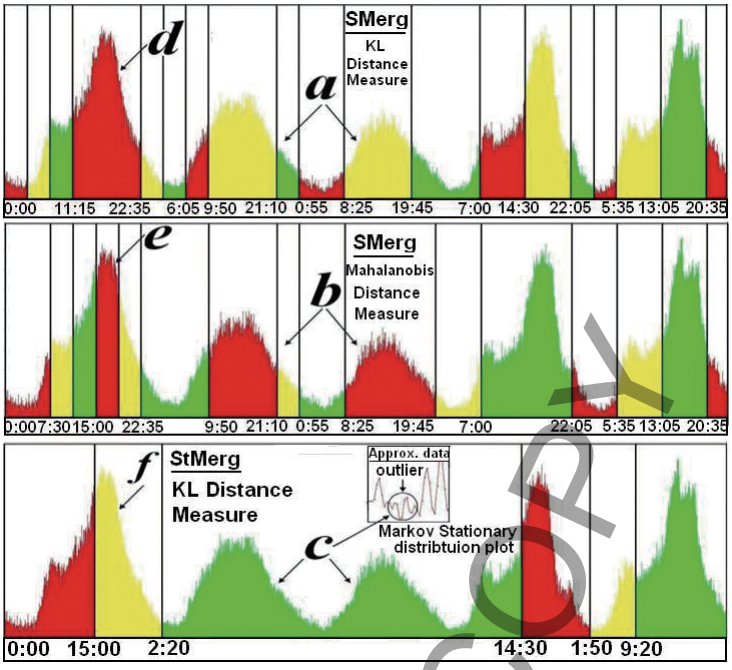

Fig. 8. US−50 East @ Church Rd East Vehicle Counts @ MM 4.51 (5/1/2009 to 5/16/2009). (Colours are visible in the online version of the article; http://dx.doi.org/10.3233/IDA-140660)

(3) For the week days traffic let us consider Figs 7(e) and 8. These are demarcated at a higher level granularity combining the entire days traffic into one bin. However in Figs 7(d, f) and 8 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 East and west sensors in Figs 7(d) and 8. 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 Figs 7(c) and 8), which is week-end 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 Figs 7(c) and 8 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 Fig. 9, temporal interval (a) and (b) obtained after merging the intervals shows that there is an anomaly 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.

4.2.2. 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

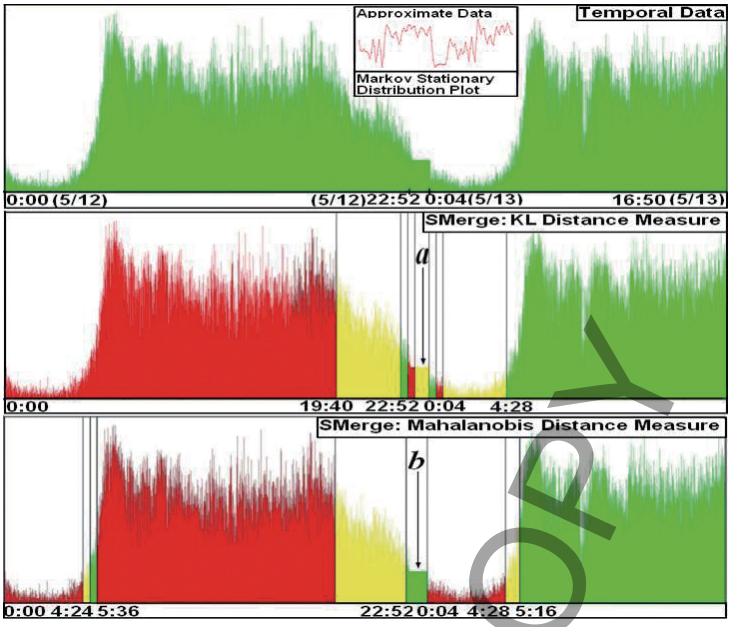

Fig. 9. I−395 NEAR Seminary Rd Vehicle Counts @ MM 4.51 (5/12/2009 to 5/13/2009). (Colours are visible in the online version of the article; http://dx.doi.org/10.3233/IDA-140660)

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 measure 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 Figs 10 and 11 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 Fig. 10 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 in Fig. 11, as the most scalable and outperforms the other measures with increasing data size, in terms of precision and recall values.

Initial number of bins: We found 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).

Threshold value k for SMerg and h for StMerg: Accuracy of the proposed methods is dependent on the threshold value to an extent. The best result is obtained when the bin merging threshold value is typically $\frac{1}{n-1}$ in case of the *SMerg* method and high pass filter threshold is between 5%–30% (10% on average) of the initial number of bins in case of the *StMerg* method. Also, the Mahalanobis is the distance measure that is least sensitive to the variation in threshold value for both *SMerg* and *StMerg*.

Discussion of results: It should be noted that the algorithm performance depends upon the initial number of bins. However this can be mitigated by striking a balance between the initial number of bins

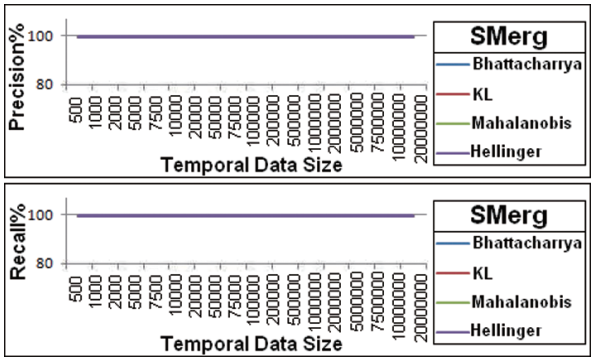

Fig. 10. Precision and recall of SMerg, temporal data size increasing. (Colours are visible in the online version of the article; http://dx.doi.org/10.3233/IDA-140660)

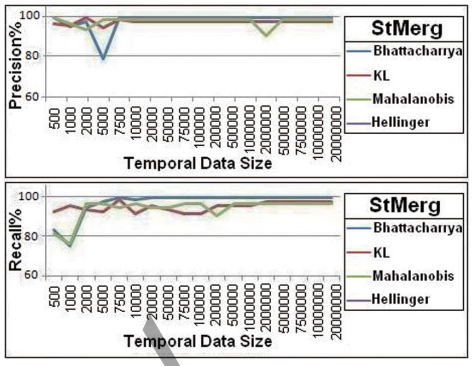

Fig. 11. Precision and recall of StMerg, temporal data size increasing. (Colours are visible in the online version of the article; http://dx.doi.org/10.3233/IDA-140660)

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 measure. 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.

4.3. Results with GMerg and OptMerg

We next discuss results varying the following: (a) Impact of number of temporal observations on *GMerg* and *OptMerg*, (b) Change of threshold for *GMerg* and *OptMerg*, (c) Initial number of bins and threshold for *OptMerg*.

Impact of number of temporal observations on GMerg and OptMerg: The results for change in the number of temporal observations for *GMerg* and *OPTMerg* are shown in Figs 12 and 13 respectively. It can be seen that both the algorithms are able to identify the neighborhoods accurately even with very large sized data.

Impact of change of threshold and initial number of bins on GMerg and OptMerg: In general it was found that the algorithms are dependent on the threshold. However this impact can be mitigated by corresponding changes in the number of initial bins. There is a range of threshold values or an operating region for which the precision and recall are maximum. This region varies with initial bin size and also with distance measures used.

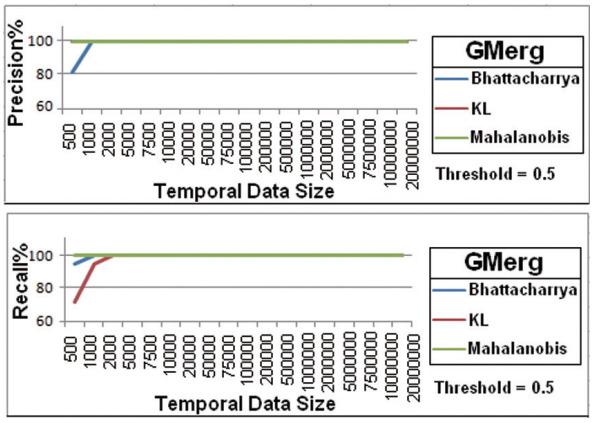

Fig. 12. Precision and recall of GMerg, temporal data size increasing. (Colours are visible in the online version of the article; http://dx.doi.org/10.3233/IDA-140660)

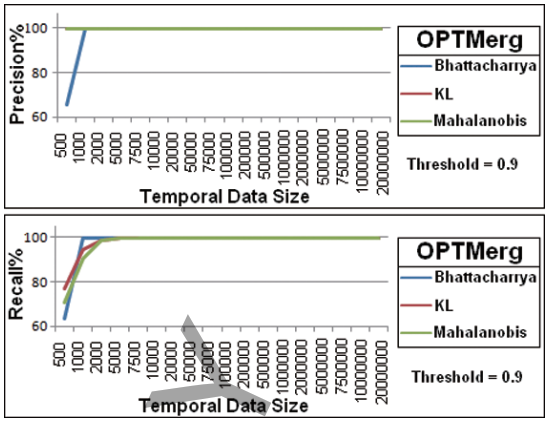

Fig. 13. Precision and recall of OPTMerg, temporal data size increasing. (Colours are visible in the online version of the article; http://dx.doi.org/10.3233/IDA-140660)

Essentially a balance has to be struck between the initial number of bins and the threshold setting for the algorithms. For example if the initial number of bins is 10 to 100 the threshold must be equivalently decreased from 0.8 to 0.4.

Runtime: As can be seen from the Fig 14, the algorithms *SMerg* and *GMerg* perform better than the more sophisticated *STMerg* and *OPTMerg* in terms of the runtime, when appropriate thresholds are chosen as stopping criteria.

4.4. Overall observations

The overall comparison of all algorithms is summarized below. In general we can see from the precision and recall numbers that all algorithms *SMerg*, *STMerg*, *OPTMerg*, *GMerg* preform well with the lowest recall for *STMerg* at 95.79%. We make additional observations about the various algorithms and distance measures in terms of the various settings and parameters.

- In general *SMerg* and *OptMerg* perform the best in terms of precision and recall.
- For *OptMerg* and *GMerg*e techniques as we go on increasing initial bin size (e.g. from 10 to 100 initial bins for 1000 data tuples) the threshold must be decreased equivalently for *OptMerg* and *GMerg* technique (e.g. from 0.8 to 0.4)
- *OptMerg* performs best with Bhattacharrya distance measures for higher threshold (outperforms all distance measures), unlike *KL* which performs better at lower thresholds for this algorithm. However, Bhattacharyya distance measure performance deteriorates when data is noisy. Additionally we notice that, Bhattacharyya and Hellinger distance measures perform similarly in all cases. *KL* performs better for lower thresholds in *OptMerg*. However, *KL* is more sensitive to change of window size than Bhattacharyya.
- Mahalanobis distance measures performs best for noisy data for *SMerg*, *OptMerg* and *GMerg*

4.5. Comparative results

For a comparative analysis we compare our algorithms with a variation of an existing technique [16]. Since, this approach does not produce merged bins which is the outcome of our approach, we propose

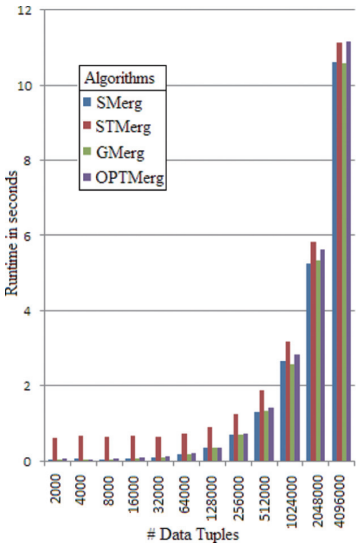

Fig. 14. Runtime comparison. (Colours are visible in the online version of the article; http://dx.doi.org/10.3233/IDA-140660)

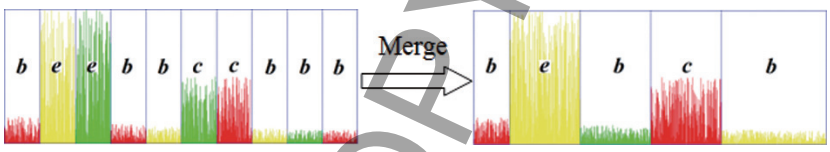

Fig. 15. SAX-Merg: Symbolic representation of bins by SAX ($w = 10$, $a = 5$) before and after merging. (Colours are visible in the online version of the article; http://dx.doi.org/10.3233/IDA-140660)

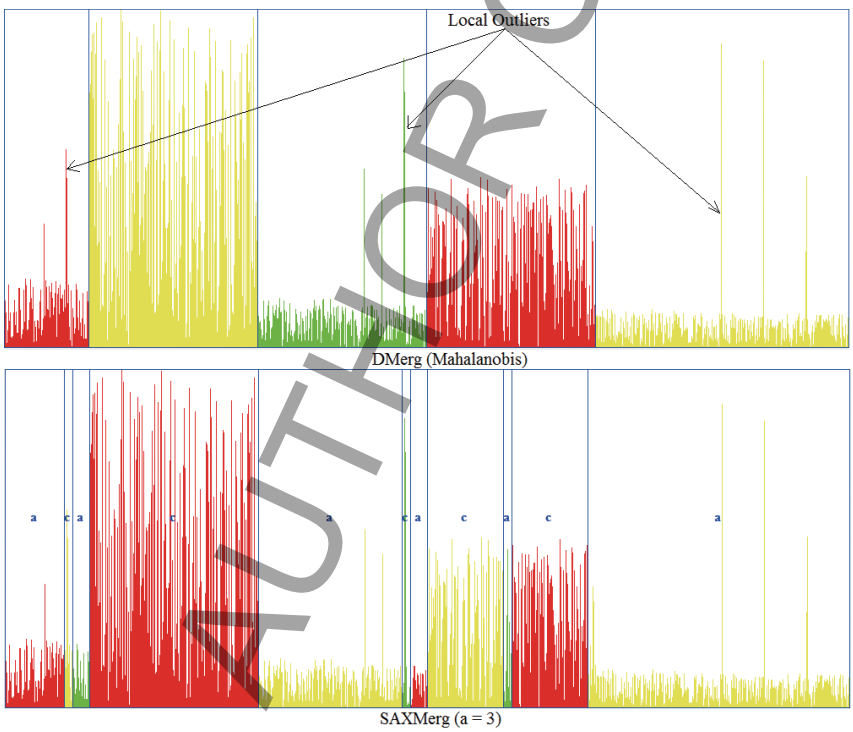

Fig. 16. Comparison of SAX-Merg with OPTMerg for synthetically generated data with (local) outliers. (Colours are visible in the online version of the article; http://dx.doi.org/10.3233/IDA-140660)

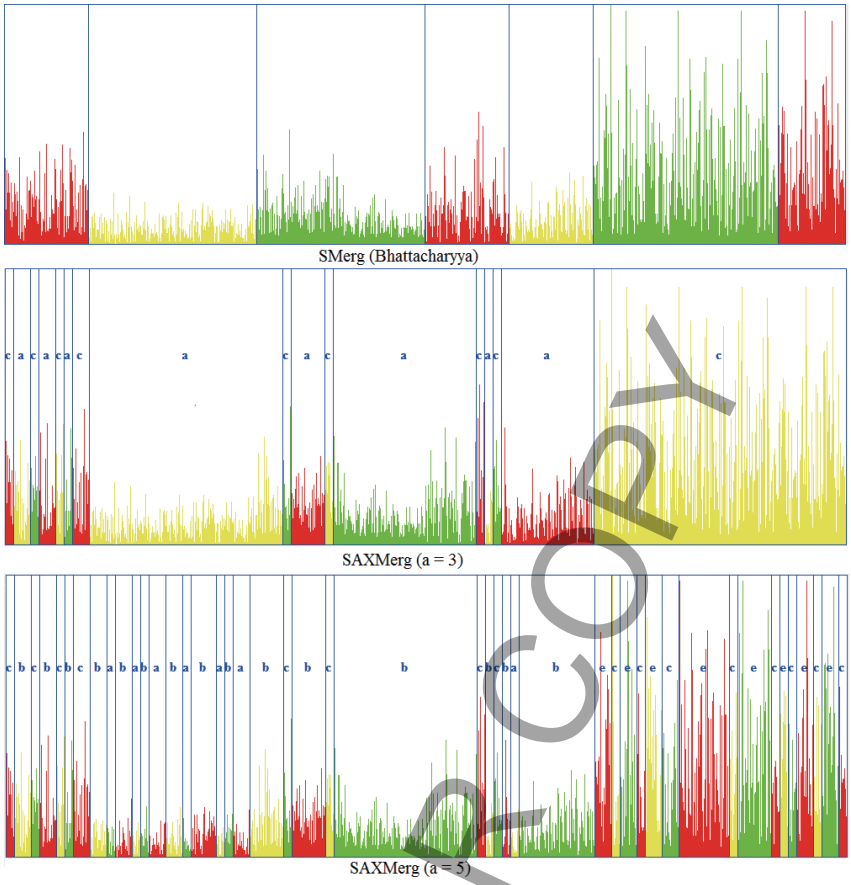

Fig. 17. Comparison of SAX-Merg with SMerg for synthetically generated data. (Colours are visible in the online version of the article; http://dx.doi.org/10.3233/IDA-140660)

a slight addition to take the output from SAX and generate the temporal neighborhoods. We use SAX technique to identify the bins and obtain temporal neighborhood by essentially merging the bins that are assigned the same alphabet using the SAX algorithm discussed in [16]. It is important to note that we do not modify their merging process but simply add a step to produce the output in the desired format for a comparative analysis.

Using the SAX technique, first the temporal data C of length N, with data tuples $c_1 \ldots c_N$ is represented in a w-dimensional space ($w = n$ in our case) by a vector $\bar{C} = \bar{c_1}, \ldots \bar{c_n}$, (dimensionality reduction via PAA) [16] 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 found and the temporal data are discretized into the bins (assign the alphabets to bins) by comparing the PAA coefficients and breakpoints. An example of how we use SAX for merging bins is shown in Fig. 15 we refer to this merging with SAX as SAX-Merg.

Results in the synthetic data with outliers We compare our algorithms with the approach outlined in [16] as shown in Figs 16 and 17. We can see that [16] identifies very fine details in the temporal data so even the outliers may end up in their own bins and the temporal neighborhoods may not be accurately demarcated. [16] can work well if the main aim is to identify outliers however for a neighborhood discovery for further knowledge discovery, our approach is well suited since it is robust in the presence

of outliers. Mahalanobis distance is the most resistant to noise, hence using it, exactly same set of final bins are produced even after introducing noise in the data. *KL* being sensitive to noise tends to separate outliers in different bins (as SAX-Merg does).

Observations: SAX tends to capture more delicate changes with increase of alphabet size. Two bins are merged if their means are equal, since two bins with different distributions can have same means thus two dissimilar bins can be merged wrongly and two similar bins may not be merged occasionally (affecting precision and recall). Extended merge keeps min and max (range) for each bin, however standard deviation is a better dispersion measure than maintaining the range.

5. 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. Currently the *STMerg* algorithm does not perform well since we use Fourier transform as HPF, but DFT performs poorly to detect sharp changes – we plan to use Wavelet transform that performs much better as HPF to improve the performance. 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] A. Bhattacharyya, On a measure of divergence between two statistical populations defined by probability distributions, *Data Min Knowl Discov* **35**(99) (1943), 09.
[4] G.E.P. Box and M.E. Muller, A note on the generation of random normal deviates, *The Annals of Mathematical Statistics* **29**(2) (June 1958), 610–611.
[5] E. Brigham, *The Fast Fourier Transform*, New York: Prentice-Hall, 2002.
[6] I.C. 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 November 2007.
[7] S. Dey, V.P. Janeja and A. Gangopadhyay, Temporal neighborhood discovery using markov models, in: *ICDM* (2009), 110–119.
[8] A.P. Di, The stationary distribution of a markov chain, la sapienza of rome, (May 2005).
[9] 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* (27–31 August 1998), 239–241.
[10] U.D.O.T. Federal Highway Administration, Traffic bottlenecks: A primer focus on low-cost operational improvements, *Technical Report, Federal Highway Administration, Office of Transportation Management*, (July 2007), last accessed, Nov 2007.
[11] D. Freedman and P. Diaconis, On the histogram as a density estimator: L2 theory, *Probability Theory and Related Fields* **57**(4) (1981), 453–476.
[12] R. Horn and C. Johnson, *Matrix Analysis (chapter 8)*, Cambridge University Press, 1990.
[13] 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*, (29 Nov–2 Dec 2001), 273–280.

[14] K.H. Knuth, Optimal data-based binning for histograms, 2006.
[15] S. Kullback and R. Leibler, On information and sufficiency, *The Annals of Mathematical Statistics* **22**(1) (1951), 7986.
[16] 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).
[17] 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* (1–3 March 2006).
[18] P.C. Mahalanobis, On the generalised distance in statistics, in: *Proceedings of the National Institute of Sciences of India* **12**(49) (1936), 5.
[19] F. Mörchen and A. Ultsch, Optimizing time series discretization for knowledge discovery. in: *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining* **KDD'05** (2005), 660–665, New York, NY, USA, ACM.
[20] D.E. Pollard, *A User's Guide to Measure Theoretic Probability*, Cambridge, UK: Cambridge University Press, 2002.
[21] M. Rudemo, Empirical choice of histograms and kernel density estimators, *Scandinavian Journal of Statistics* **9**(2) (1982), 65–78.
[22] D. Schrank and T. Lomax, Urban mobility report, Technical report, 2007, last accessed, November 2007.
[23] D.W. Scott, On optimal and data-based histograms, *Biometrika* (1979), 605–610.
[24] 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).
[25] R. Sinkhorn, A relationship between arbitrary positive matrices and doubly stochastic matrices, *AMS* **35** (1964), 876–879.
[26] C.J. Stone, An asymptotically histogram selection rule, 1984.
[27] M.P. Wand, Data-based choice of histogram bin width **51**(1) (February 1997).