Automatic Segmentation of Data Sequences

Liangzhe Chen, Sorour E. Amiri, B. Aditya Prakash
Department of Computer Science, Virginia Tech.
Email: {liangzhe, esorour, badityap}@cs.vt.edu

Abstract

Segmenting temporal data sequences is an important problem which helps in understanding data dynamics in multiple applications such as epidemic surveillance, motion capture sequences, etc. In this paper, we give DASSA, the first self-guided and efficient algorithm to *automatically* find a segmentation that best detects the change of pattern in data sequences. To avoid introducing tuning parameters, we design DASSA to be a multi-level method which examines segments at *each level* of granularity via a compact data structure called the *segment-graph*. We build this data structure by carefully leveraging the information bottleneck method with the MDL principle to effectively represent each segment. Next, DASSA efficiently finds the optimal segmentation via a novel average-longest-path optimization on the segment-graph. Finally we show how the outputs from DASSA can be naturally interpreted to reveal meaningful patterns.

We ran DASSA on multiple real datasets of varying sizes and it is very effective in finding the time-cut points of the segmentations (in some cases recovering the cut points perfectly) as well as in finding the corresponding changing patterns.

1 Introduction

Given a data-sequence of Ebola infections, can we quickly tell when the characteristics of infected people change, possibly due to a mutation? In this paper, we study the problem of *automatically* segmenting sequences of multi-dimensional data point (with categorical and/or real-valued features like age, gender, speed etc.) so as to capture relevant trends and changes. The data observations can be unevenly distributed temporally and repeated multiple times in the sequence, naturally generalizing multi-variate time-series.

Such segmentations can be helpful in many real applications, as they may shed light on the underlying dynamics and patterns, thereby helping in modeling, anomaly detection, and also visualization. Consider epidemiological surveillance, where tracking disease propagation (Thompson, Comanor, and Shay 2006) can enhance the chance of a successful intervention and increase the situation awareness. For example, automatically finding changes in patient characteristics in a sequence of infected cases can help us point to changes in the disease itself. In Fig. 1(a), in a series of flu-infections, DASSA finds that the disease infects elder, richer people first, and then spreads to younger people with lower income. Similarly, figuring out the changes in how words are used together in user tweets (due to changes in users' health status) can help in estimating disease incidence (Chen et al. 2014). See Fig. 1(b): in a series of flu-related tweets, we observe a transition between word usage in each segment from infection to recovery. Our motivation in this paper is to design a general-purpose scalable segmentation algorithm for data sequences.

Informal Problem: Given a multi-dimensional data sequence, *automatically* find a *time segmentation* s.t. consecutive segments are not *similarly* informative.

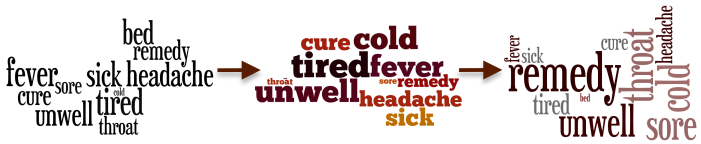
Surprisingly, despite its importance, this general problem has not been studied widely (see related work in Sec. 5). Such a problem cannot be trivially converted to one in time series, and has many challenges. The main properties we want a good solution to satisfy are:

- **P1 (Generality):** No prior assumption on either data types or data distributions. The algorithm should work well regardless even if the data is skewed, or not forming clusters or not drawn from a known distribution.
- **P2 (Self-Guided):** Automatically find the appropriate number and identity of cut-points without user input.
- **P3 (Efficiency):** Finish within reasonable time for real datasets.

In this paper, we present an algorithm DASSA (DAta Sequence Segmentation Automatically), which satisfies all the three properties. To this end, we introduce three main ideas which may be useful for other segmentation problems as well: (a) looking at all possible segmentations efficiently using the so-called segment-graph; (b) compressing data in each segment based on temporal data distributions using Information Bottleneck and Minimum Description Length; and (c) using a novel path optimization to find the best segmentation, which automatically regularizes the number of segments and total segment difference. Via extensive experiments ranging from epidemiological, social to motion capture datasets we show how DASSA can recover high-quality segmentations, and meaningful patterns in practice. To the best of our knowledge, we are the *first* to present an efficient, self-guided method for the purpose of segmenting general data sequences.

	age	Y	X	Income	Size	#Workers	#Vehicles
Segment:1	4.0	4.0	4.0	10.0	0.0	3.0	5.0
	4.0	3.0	4.0	10.0	0.0	3.0	2.0
	4.0	4.0	2.0	10.0	2.0	5.0	5.0
	4.0	3.0	4.0	10.0	2.0	3.0	2.0
	age	Y	X	Income	Size	#Workers	#Vehicles
Segment:2	1.0	5.0	7.0	6.0	5.0	1.0	2.0
	4.0	5.0	7.0	3.0	0.0	1.0	1.0
	4.0	6.0	6.0	7.0	1.0	5.0	4.0
	2.0	5.0	5.0	7.0	0.0	3.0	2.0

(a) Segmenting a sequence of flu-cases



(b) Segmenting a sequence of words appearing in tweets

Figure 1: *Our method DASSA gives meaningful cut-points: (a) Most frequent data values (discretized) in segments detected in flu-infection data Portland. (b) Word clouds for each detected segment for the Twitter dataset Peru. Size of a word is proportional to its frequency in the corresponding segment. More discussion in Sec. 6.*

2 Preliminaries

Information Bottleneck (IB): The IB method (Tishby, Pereira, and Bialek 1999) compresses one signal X to the 'bottleneck' $\tilde{X}$ (much smaller than X in size) without much loss of its information related to another signal Y. The optimization problem it solves is:

$$\min I(\tilde{X};X) - \beta I(\tilde{X};Y) \tag{1}$$

where $I(\cdot;\cdot)$ represents the mutual information between two variables, β is the Lagrange multiplier. Given $|\tilde{X}|$ and $p(X,Y)$, this optimization problem can be solved iteratively (Tishby, Pereira, and Bialek 1999) to provide these distributions: $p(\tilde{x}|x)$, $p(\tilde{x})$ and $p(y|\tilde{x})$; where x, $\tilde{x}$ and y are possible values of X, $\tilde{X}$, and Y.

Slonim et. al. (Slonim and Tishby 2000) develop a variation of IB for word-document clustering, where they use words as the X signal, documents as the Y signal, and $\tilde{X}$ as the labels for words. In this formulation, they use hard-clustering (each x is mapped to exact one $\tilde{x}$) to cluster words so that the information in the documents are maximally kept. They initialize the problem with no compression ($\tilde{X} = X$), and greedily choose the label pairs with smallest marginal loss $\delta I(\tilde{X},Y)$ (mutual information between $\tilde{X}$ and Y) to merge. The loss by merging $\tilde{x_i}$ and $\tilde{x_j}$ is defined as follows:

$$\delta I(\tilde{x_i},\tilde{x_j}) = (p(\tilde{x_i}) + p(\tilde{x_j})) * D_{JS}[p(y|\tilde{x_i}), p(y|\tilde{x_j})] \tag{2}$$

where D_{JS} is the Jensen-Shannon divergence.

Minimum Description Length (MDL): The MDL principle (Grünwald 2007) suggests that the best hypothesis for a given set of data, which captures the most regularity in the data, is the one that leads to the best compression of the data (Waggoner et al. 2013; Chen et al. 2014). MDL finds the best model which minimizes

$$Cost_T = Cost(M) + Cost(X|M) \tag{3}$$

where $Cost(M)$ is the cost to describe the model, $Cost(X|M)$ is for describing the data using the model.

3 Overview

We present the main principles of DASSA (Alg. 1 shows the basic steps) next.

Algorithm 1 Pseudo-code for DASSA

Input: $\mathbb{D}$
Output: The best segmentation S^*

1: $[\tilde{X}, p(\tilde{x}|x)]$=Cluster ($\mathbb{D}$).//Finding data clusters using IB and MDL (Sec. 4.2)
2: Build a node for every possible time segment y.//Constructing $\mathcal{G}$ (Sec. 4.1)
3: Add node s and t to represent the start and end time of $\mathbb{D}$.
4: Create edges for adjacent y's.
5: Calculate the edge weights as the Euclidean distance between the corresponding conditional cluster distribution $p(\tilde{x}|y)$.
6: S^* = DAG-ALP ($\mathcal{G}, h, s, t$).//Finding the ALP as S^* (Sec. 4.3)

Definition 1 (Data sequence) *A data sequence $\mathbb{D}$ is a list of tuples $(\mathbf{x}_1,t_1),\ldots,(\mathbf{x}_N,t_N)$, where $(\mathbf{x}_i,t_i)$ is an observation of d-dimensional vector $\mathbf{x}_i$ at time t_i.*

Let $X = \bigcup_i \{\mathbf{x}_i\}$. W.L.O.G we assume time stamps are sorted, i.e. $t_1 \leq t_2 \leq \ldots \leq t_N$. Note that both $\mathbf{x}_i$'s and t_i's are not necessarily unique. We can have observations with the same data value, and there can be multiple data observations having the same time stamp. This general definition of data sequences covers special cases like time series, where the number of data observations at each time is the same; and event sequences, where $x_i's$ are one-dimensional categorical values. We want to design an algorithm that automatically finds segmentations for such data sequences, and that satisfies all the desired properties (**P1**-**P3**).

Main Ideas: To avoid introducing parameters like the desired number of segments and to find the segmentation in an automatic manner (**P2**), our search space would inevitably be the set of all possible segmentations which is exponential in size. Our first main idea is to use a graph data structure (called the *segment-graph* $\mathcal{G}$) to efficiently represent and search among all possible segmentations of the data sequence. See Fig 2(c) for an example $\mathcal{G}$. The node set of $\mathcal{G}$ mainly represents all possible time segments $Y = \{y_{i,j}\}$ ($y_{i,j}$ is the segment from time i to j), s and t represent the start and end time, and the edge weights are distances (i.e. the 'difference') between adjacent time segments. With this data structure, segmentations of the data sequence are now mapped to paths from start time s to end time t in $\mathcal{G}$. Hence, finding the best segmentation is now converted to the problem of finding the 'best' start-to-end path in the segment-

graph $\mathcal{G}$. To solve the segmentation problem using $\mathcal{G}$, two important questions remain unsolved: **Q1**: how do we define the difference metric $w(\cdot)$ between two time segments, and **Q2**: what is the best start-to-end path in the segment-graph, and how do we find it efficiently?

Q1: Segment difference. Due to **P1**, we cannot use model-based methods (which typically assume certain data distributions like Gaussians in each segment or overall in $\mathbb{D}$) for our problem. Our second main idea is to cluster data values based on their 'temporal closeness', and then represent each segment using their conditional cluster distributions ($p(\tilde{x}|y)$, the probability of cluster $\tilde{x}$ given a segment y). We can then measure the segment difference simply as the difference between their $p(\tilde{x}|y)$'s. Intuitively, clusters based on how data values are temporally distributed *over all possible segments* Y naturally captures the 'similarity' between data values, which is well-suited for segmentation problems: if two data values always occur close in time at multiple granularities, they contain similar information as to defining the best segmentation. A major advantage is that clustering 'temporally close' data values is not data-type specific and it does not need any prior assumptions on the data distributions. It is also more general than the traditional clustering of data with *similar* values, as data values with similar temporal occurrence *may not* have similar values.

Due to **P2**, we want to find these temporally similar data clusters in a principled, unsupervised fashion. The **I**nformation **B**ottleneck (IB) formulation is very well-suited for this task—thinking of segments Y as 'documents' and data values X as 'words' allows us to leverage IB to cluster data values with similar segment distributions $p(y|x)$ *without* specifying an explicit distance metric. As IB is non-parametric, to automatically find the appropriate number of clusters, we further design and optimize a novel **M**inimum **D**escription **L**ength code. Both IB and MDL are based on sound information theory principles. Note that in contrast to other methods (topic modeling, biclustering, etc), IB has exact formal solutions and other advantages (see Sec. 5).

Q2: Best path. The main challenges are (a) how to define this best path; and (b) how to find it efficiently in the potential exponential search space. In the optimal segmentation, we require the adjacent time segments to be different which may naïvely suggest choosing a path with the maximum sum of weight. At the same time, we want to avoid over-segmenting (having more segments than needed). Due to these considerations, instead, we propose to define the best path in $\mathcal{G}$ as the one that has the maximum *average* edge weight. This definition intrinsically balances the difference of segments and the number of segments, and finds the segmentation automatically without setting the number of segments as an input parameter (**P2**). We further propose a novel efficient DAG-ALP algorithm for finding the average longest path for DAG.

4 Details of DASSA

We now give details about DASSA. First, we introduce s_{min} as the unit time length, and divide the time period into these small time units. Hence, a time cut point c_i can be defined as $c_i = t_{min} + i \cdot s_{min}$, $i \in \mathbb{N}$, and $t_{min} \leq c_i \leq t_{max}$, where $t_{min} = \min(t_i)$, and $t_{max} = \max(t_i)$. Now we define a time segment.

Time segments and MTS: A time segment $y_{i,j}$ is a time interval between any two cut points $[c_i, c_j)$. A Minimum Time Segment (MTS) is a time segment $y_{i,j}$ between two adjacent cut points, i.e. $j = i + 1$.

Naturally following, all MTS's have length s_{min}, and they are the smallest time segments of our interest. We further define the set of all possible segments $Y = \{y_{i,j} | c_j - c_i \leq s_{max}\}$, where we assume s_{max} is the maximum segment size we allow in a segmentation (like a year in a twitter data). In experiments, when a natural upper bound is available, we set the s_{max} accordingly, otherwise we set it trivially as $t_{max} - t_{min}$. Note that, we introduce s_{min} and s_{max} mainly to incorporate domain knowledge if there's any. Our algorithm still looks at segments at all granularities of all sizes (in multiples of s_{min}) as we explain later. In principle, we can set these parameters via cross validation, but our results are robust when there are slight changes of them.

Segmentation: A segmentation S is a set of consecutive segments $S = \{y_{a_1,a_2}, \ldots, y_{a_m,a_{m+1}}\}$ where each $y_{a_i,a_j} \subset Y$ and $c_{a_1} = t_{min}, c_{a_{m+1}} = t_{max}$.

We show a running example data sequence in Fig. 2(a), the optimal segmentation is shown with the red dash line, which captures the fact that 1, 100, 2 occur together in the sequence.

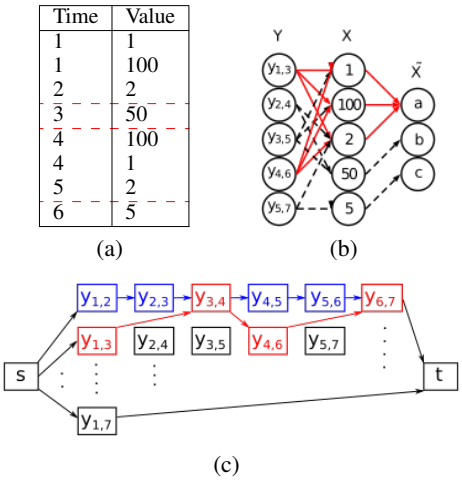


Figure 2: *(a) shows an example data sequence, (b) results from our* Cluster *algorithm, X and Y are connected if the data value x appears in the corresponding y. Value* 1, 100, 2 *are merged to cluster a because they occur together in the sequence, (c) the segment-graph $\mathcal{G}$, the path/segmentation found by* DASSA *is marked as red.*

4.1 Segment-graph

We construct a Directed Acyclic Graph (DAG) segment-graph $\mathcal{G}(V, E, W)$ to efficiently represent and search among all possible segmentations. We show $\mathcal{G}$'s structure below.

Nodes (V): For each segment $y_{i,j}$ in Y, we construct a corresponding node in a graph $\mathcal{G}$. We also add two extra

nodes to the graph: a source node s and a target node t (i.e. $V = \{y_{1,2}, y_{1,3}, \ldots, y_{2,3}, \ldots\} \cup \{s, t\}$).

Edges (E): We create a directed edge from node $y_{i,j}$ to node $y_{k,l}$ iff $j = k$, i.e. they are adjacent segments. Source node s links to all nodes with start time t_{min}; target node t, absorbs links from all nodes with end time t_{max}.

$\mathcal{G}$ is clearly a DAG (as we cannot go back in time), and every path from s to t is one-to-one mapped to a segmentation of the sequence. Hence the segmentation problem is now converted to the problem of finding the best path in $\mathcal{G}$.

4.2 Q1: Defining edge weights

The edge weight $w(e(y_{i,j}, y_{j,k}))$ measures the difference between adjacent segments $y_{i,j}$ and $y_{j,k}$. We now propose our algorithm Cluster, which combines IB and MDL to automatically cluster data values based on their segment distributions $p(y|x)$ to capture their 'temporal similarity', and define the edge weight as the distance between $p(\tilde{x}|y)$. To facilitate calculating the occurrence of the same value, for features with real values, we discretize them to a constant number of bins of equal size/width as in past literature (Shokoohi-Yekta et al.). In the following, we assume all $\mathbf{x_i}'s$ are discretized.

Finding clusters using IB We define the set of clusters we want to find as $\tilde{X} = \{\tilde{x_1}, \tilde{x_2}, \cdots, \tilde{x_l}\}$, where each $\tilde{x_i}$ contains a set of $x's$ in the data space X, and l is the number of clusters (we discuss how to automatically find l in the next section). We assume each x belongs to only one $\tilde{x}$.

We want to cluster X to $\tilde{X}$ so that $x's$ with similar occurrence in Y are merged in the same $\tilde{x}$. For this task, we re-purpose the word/document formulation of IB (Slonim and Tishby 2000), designed to cluster words based on the word-document structure. We interpret the set of data values X in our setting as the 'words' and the set of all possible time segments Y as the 'documents'. Since such IB formulation would cluster data values with similar segment distributions $p(y|x)$ (in an information theoretic way *without* specifying a distance metric), we essentially cluster data values with similar temporal occurrence over all time segments. We initialize $\tilde{X} = X$ (each x_i in its own cluster $\tilde{x}_i$), then iteratively merge $\tilde{x}_i, \tilde{x}_j$ pairs which minimizes the loss of temporal information specified as $\delta I(\tilde{x_i}, \tilde{x_j}) = (p(\tilde{x_i}) + p(\tilde{x_j})) * D_{JS}[p(y|\tilde{x_i}), p(y|\tilde{x_j})]$, where D_{JS} is the Jensen-Shannon divergence. Such an iteration process continues until we reach the desired number of clusters l^*. In implementation, we use a priority queue to efficiently find the best data values to merge in each iteration, and reduce the time complexity from $O((|X|-l)|X|^2)$ to $O((|X|-l)|X|\log|X|)$.

Number of clusters using MDL To *automatically* find the appropriate number of clusters l^* in $\mathbb{D}$, we propose to use the MDL principle: the best model for the data is the one that expresses the data losslessly with the smallest code length. To apply MDL, we construct a model class for any cluster number l which combines the corresponding cluster information and some other information (which is needed to express the data losslessly), and then select the best model (and the corresponding l^*) based on the data and the model cost.

Model description. As IB is a lossy compression method, we cannot express the data losslessly using just the cluster information. Hence, we augment the IB results with the following additional information: (a) $p(\mathbf{x}_j|\tilde{x}_i)$, data value distribution in each cluster; (b) $p(y|\tilde{x}_i)$, the probability of a cluster being in a time segment; and (c) $p(\tilde{x}_i)$, the prior for $\tilde{x}_i$. Formally our model is $\theta = \{l, N, |Y|, p(\tilde{x}_i|\mathbf{x}_j), p(y|\tilde{x}_i), p(\tilde{x}_i), p(\mathbf{x}_j|\tilde{x}_i)\}$, where $l = |\tilde{X}|$, and $N = |\mathbb{D}|$. To describe the model, we need to encode the set $\theta \in \mathcal{M}$. So our model description cost is:

$$C(M) = \log^* l + \log^* N + \log^* |Y| + N \log l - (l|Y| + |\tilde{X}| + l|X|) \log \epsilon \quad (4)$$

where ϵ is the precision for the probability values ($\epsilon = 10^{-5}$ indicates a precision of 0.00001), and $\log^*(n) = \log n + \log\log n + \ldots$ (it is roughly the number of bits to encode an integer $n \geq 1$).

Data description. To describe the data, naïvely one can describe $(\mathbf{x}_j, \{y|t_j \in y\})$ for all y covering t_j. We observe that all time segments $\{y|t_j \in y\}$ containing $\mathbf{x}_j$ must also contain the MTS that covers $\mathbf{x}_j$ (followed from our segment definition). Hence, the likelihood of observing $\mathbf{x}_j$ is equivalent to the likelihood of observing it in the MTS that contains it. Using this observation, we can reduce the number of $(\mathbf{x}, y)$ pairs we need to describe from $|X||Y|$ to $|X|$. We then derive the final data description cost as:

$$Cost(X|M) = -\log_2 L(X, Y|\theta) = -\sum_{(\mathbf{x}_j, y)} \log_2 p(\mathbf{x}_j, y|\theta) = -\sum_{(\mathbf{x}_j, y)} \log_2 p(\mathbf{x}_j|\tilde{x_*}, \theta) p(y|\tilde{x_*}, \theta) p(\tilde{x_*}|\theta) \quad (5)$$

where $\tilde{x_*}$ is the corresponding cluster for x.

The total cost. Combining the above, the total cost of this description based on the model we described is $C_T = C(M) + C(X|M) = Eq.\ 4 + Eq.\ 5$. The best model minimizes C_T, i.e. $\theta^* = \arg\min_\theta C_T$, and θ^*'s corresponding l value is the optimal number of clusters. This cost function is hard to optimize: hence we leverage a greedy approach that naturally fits the iteration process we introduced before. We keep greedily merging $\tilde{x}_i, \tilde{x}_j$, and for each s_{mdl} merges, we calculate the corresponding C_T. This iteration process stops (reaching optimal l^*) when C_T begins to increase.

Final edge weights Once we find $\tilde{X}$ and $p(\tilde{x}|x)$, we can calculate the cluster distribution $p(\tilde{x}|y)$ in each segment y by counting the number of times members of each cluster occur in the segment. And the edge weight between segments y_a and y_b in G can be defined as $w(y_a, y_b) = \text{Dist}(p(\tilde{x}|y_a), p(\tilde{x}|y_b))$. We want that any distance metric $\text{Dist}(\cdot, \cdot)$ we use should satisfy the following property:

Property 1 *For any three consecutive segments u, v, t, and if v can be further divided into segments v_1 and v_2 (i.e. if $v = [c_i, c_j)$, $v_1 = [c_i, c_k)$, $v_2 = [c_k, c_j)$), then* $w(e(u, v)) + w(e(v, t)) \leq w(e(u, v_1)) + w(e(v_1, v_2)) + w(e(v_2, t))$.

Intuitively, this property makes the segmentation problem well defined in the sense that adding more cut-points always gives us more difference/pattern changes (measured by the sum of edge weights)—hence 'zooming-out' i.e. aggregation by looking at larger time-segments should only *decrease* the difference. Note that capturing more pattern

Algorithm 2 Pseudo-code of DAG-ALP

Input: a weighted DAG $\mathcal{G}$ (V, E, W), h, s, t
Output: Average longest path

```
Layer_0 = {s} // initialize the first layer
lp_0(s) = 0 // the longest path form s to s with length 0 is initialized as 0
for i = 1 to h do
    Layer_i = {nodes directly connected to any nodes in Layer_{i-1}}
    Calculate lp_i(·) for nodes in Layer_i using lp_{i-1}(·)
ALP = arg max(lp_i(t)/i)
```

changes does not always lead to a *better* segmentation: having a segmentation with many small changes may be less desirable than one which captures only a few *globally significant* changes at the right segment sizes. Hence how to find the best segmentation is a separate problem.

We use the popular Euclidean distance between distributions (like used in (Liu et al. 2012)) i.e. $w(e(y_a, y_b)) = D_{EU}(p(\tilde{x}|y_a), p(\tilde{x}|y_b))$, which satisfies property 1. The proof follows from the subadditivity (triangle inequality) of D_{EU}. In contrast, the well-known KL divergence does not satisfy this property in general.

4.3 Q2: Finding the best path

In the weighted $\mathcal{G}$, the problem of finding the optimal segmentation is now reduced to finding the 'best' path from the set of all valid paths $\mathcal{P}$ in $\mathcal{G}$.

We argue that a good segmentation should regularize the total segment difference with the number of segments: having many small changes is less desirable than capturing just the significant ones. Hence, we propose to solve the Average Longest Path problem (ALP) to find the best path.

Given: Segment-graph (DAG) $\mathcal{G}$ (V, E, W) with a start node s and end node t.

Find: Path S^* from s to t with maximum average weight: $S^* = \arg\max_{S \in \mathcal{P}} \frac{\sum_{\text{e in } S} w(e)}{|S|}$.

We present a novel ALP algorithm DAG-ALP with $O(h \cdot |E|)$ on general DAGs (h is the maximum path length in the DAG). Our idea is that the ALP from s to t must also be the *longest (most heavily weighted)* path among all paths with the same number of nodes. Hence, we calculate all the longest paths with *different* lengths (number of nodes) from s to t, and find the one giving the maximum average edge weight. More concretely, DAG-ALP uses a multi-layer structure, where the first layer L_0 contains only the beginning node s, and layer L_i contains the nodes which can be reached from s by i steps. When we iterate through layers, we maintain the weight ($lp_i(v)$) of the longest path from s to $v \in L_i$, and the parent node of v in L_i ($\pi(v, i)$) in the longest path. After all iterations, we get longest paths from s to t with different lengths, and we output the one with the largest average weight. Alg. 2 shows the brief pseudo-code. Due to the structure of our segment-graph, DAG-ALP finds the ALP in $\mathcal{G}$ in $O(E)$ time (proofs omitted due to space).

Time and space complexity The pseudo-code of DASSA is shown in Alg. 1. With priority queue, reduction of unnecessary data description, and DAG-ALP, our final time complexity is $O((|X| - l^*)|X| \log |X| + |E|)$. To find the ALP we only need to store the previous layer in DAG-ALP, hence the overall space complexity of DASSA is O($|\mathbb{D}|$). In practice, for all datasets used in our experiments, DAG-ALP finishes within $40s$, and the complete algorithm takes $30m$ to run on average (including one with 2 million data observations), satisfying **P3**.

5 Related Work

We review the most closely related work here.

Event sequence mining. Related work include finding summaries of event sequences (Kiernan and Terzi 2009), developing streaming algorithms (Patnaik et al. ICDM 2012), pattern sets mining (Tatti and Vreeken 2012), episode mining (Wu et al. 2013), progression stage analysis (Yang et al. 2014). Their datasets can be understood as one-dimensional categorical data sequences. In contrast, we study a more general case, where the data can be multi-dimensional, and both real and categorical.

Time series analysis. There has been a lot of work on time series, such as modeling co-evolving time series using multi-level HMMs (Matsubara, Sakurai, and Faloutsos 2014), discovering patterns in data streams (Toyoda, Sakurai, and Ishikawa 2013; Rosman et al. 2014), developing online algorithms for frequent sequence mining (Mueen and Keogh 2010), time series segmentation (Samé and Govaert ; Li et al. 2009; Loglisci and Berardi 2006), change detection algorithms (Nguyen and Vreeken 2016; Chen et al. 2013), temporal clustering (Nguyen and Torre 2012). All these methods, while very valuable, work on single or multiple time series, but we focus on more general data sequences with multi-dimensional data points, and the data points can have arbitrary time stamps (certain time periods may have many more data points than others).

Others. Topic modeling (Smola and Narayanamurthy 2010; Blei, Carin, and Dunson 2010), biclustering (Madeira and Oliveira 2004), and co-clustering (Dhillon 2001) can be adapted to find relations between data values, as the IB-based clustering does in DASSA. However, the words/data found in the same topic/bicluster/co-cluster do not necessarily have similar temporal occurrence. Also, data values which occur together, may not form statistically significant topics/clusters. Hence these methods cannot be used to find temporally similar clusters. Further IB has an exact formal solution (Tishby, Pereira, and Bialek 1999). There are also some specialized algorithms e.g. (Amiri, Chen, and Prakash 2017) which deal with graph sequences. The MDL principle we used in this paper has also been used for extracting features for time series (Hu et al. 2011), and for speaker diarization (Vijayasenan, Valente, and Bourlard 2009). However, our MDL code is completely different from theirs, and to the best of our knowledge, we are the first to combine IB with MDL principle to temporally cluster data values for data sequence segmentation.

6 Experiments

Setup. Our experiments are conducted on a 4 Xeon E7-4850 CPU with 512GB of 1066Mhz main memory and DASSA

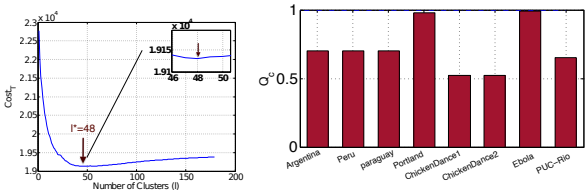
takes 30m to run on average for our datasets. For all the datasets, we set a discretization level $k = 10$ as it leads to a reasonable running time, and the performance is stable around 10 ($k = 5, 15$ gives similar results). When constructing the segment-graph in practice, we ignore segments with less than 5% of $|D|$ data values (which is a small fraction of all segments), as they have too few observations, and are not interesting for the final segmentation.

Datasets. DASSA works for general data sequences, hence we collected real world datasets from different domains to test. Tab. 1 shows the content of each data sequence. These sequences contain different data types like age, town id (categorical), sensor observations (real), etc., different time-units and some of them (like *Portland*, *Ebola*) have arbitrary time stamps (a data point can have any time stamp value, and as a result there may be different number of data points at each time stamp).

Baselines. To the best of our knowledge, there is no algorithm that finds segmentations for general data sequences as we do. Hence, we first adapt a time series algorithm *Dynammo* (Li et al. 2009) (also used in (Matsubara, Sakurai, and Faloutsos 2014)) as our baseline. Additionally, we compare with three variations of DASSA (*EMP*, *TopicM*, *LP* in Tab. 2). Note that unlike DASSA which detects the no. of cut points *automatically*, *Dynammo* needs this as an input. We set this value from the ground truth when one is available, otherwise we set it as the number detected by DASSA.

6.1 Results

Testing each component of DASSA: We check the number of clusters found by MDL, Fig. 3(a) shows that the MDL-curve is indeed near-convex, and it suggests an optimal number of clusters; We examine the quality of the detected clusters by designing a Silhouette score Q_c to measure the 'temporal similarity' of data values in the clusters, the Silhouette score (Fig. 3(b)) shows that the data values in the clusters we found truly appear close in time (all $Q_c > 0.5$); We also compare our ALP path optimization with the longest path (LP) optimization, which finds the path with the maximum sum of edge weights. Our ALP path optimization outperforms the LP optimization in all of the datasets with ground truth segmentations (see Tab. 3).



(a) MDL curve for *ChickenDance* 1 (b) Q_c scores

Figure 3: *(a) MDL curves of ChickenDance 1: C_T vs number of clusters l. (b) Q_c scores. Note $Q_c > 0.5$ for all datasets—indicates high quality clusters.*

Quality of segmentations: We measure our final segmentation output here. We show the **F_1 score** for datasets with ground truth segmentation (*Portland*, *ChickenDance*, *PUC-Rio*), and our case study results for *Twitter* and *Ebola*.

Quantitative evaluation In short, DASSA gives much better F_1 scores.

Portland: DASSA finds the exact ground truth ($F_1 = 1$ in Tab. 3), and *EMP* has a much lower score ($\sim$0.6). *TopicM* and *Dynammo* also gets $F_1 = 1$ in this dataset, but in all other datasets, DASSA outperforms both of them. We show the most frequent values in the two segments of the segmentation found by DASSA in Fig. 1(a), It shows that elderly people, with higher incomes, larger number of workers in family, and more vehicles are infected first. Then younger people with lower incomes, fewer vehicles get infected. It illustrates that DASSA is capable of detecting the pattern of disease propagation. And the results are easily interpretable.

ChickenDance: We find the exact ground truth ($F_1 = 1$). As shown in Fig. 4, DASSA discovers all the distinct chicken dance motions precisely. In contrast, the cut points detected by *EMP*, *TopicM* and *Dynammo* do not correctly find the time when a different motion takes place: they either miss the correct cut points, or have unnecessary additional ones.

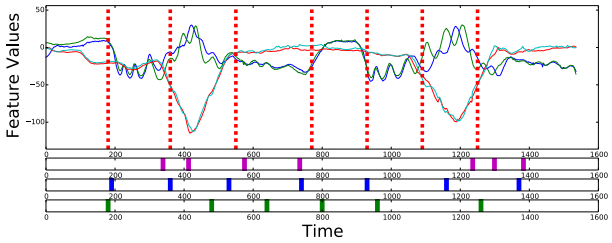


Figure 4: *DASSA segmentation results for ChickenDance. The cut points of* Dynammo *(purple in the 1st row),* TopicM *(blue in 2nd row), and* EMP *(green in 3rd row) are shown below the DASSA.*

PUC-Rio: This dataset was originally collected for classification tasks. Finding the difference between actions is itself a non-trivial task. Interestingly, DASSA is powerful enough to capture some meaningful segments. We see that in Tab. 3, DASSA reaches a F_1 score of around 0.66, which again outperforms both *EMP* and *TopicM*.

Case studies DASSA gives meaningful segments, compared to baselines.

Twitter (Peru, Paraguay, Argentina): To explore the segmentation found by DASSA, we look at users' tweets in each segment and count the number of each word to draw word clouds. The size of a word in the word cloud is proportional to the frequency of its usage in the segment. As shown in Fig. 1(b), DASSA finds three segments. We observe that the sizes/frequencies of infection-related words like 'headache', 'tired', 'fever' are decreasing from segment to segment. On the other hand, the frequency of word 'remedy' gradually increases. This matches what we expect from a typical infection cycle: from getting exposed, to getting sick, and finally to be cured. Recent work (Chen et al. 2014) also matches what we found in the word cloud (unlike us they use complex temporal graphical models to figure out

Dataset	Domain	s_{min}	s_{max}	Data sequence $\mathbb{D}$	Ground truth
Portland	Epidemiology	0.2	1.0	$\{[age, y, x, income, size, \#workers, \#cars]_i, t_i\}_{i=1}^N$	✓
ChickenDance	Motion Seq.	10s	300s	$\{[Sensor_1, Sensor_2, Sensor_3, Sensor_4]_i, t_i\}_{i=1}^N$	✓
Twitter	Social Media	10d	100d	$\{[\#(Word_1), \#(Word_2), \ldots, \#(Word_{12})]_i, t_i\}_{i=1}^N$	-
Ebola	Epidemiology	4d	48d	$\{[Infection\ Status, Town\ ID]_i, t_i\}_{i=1}^N$	-
PUC-Rio	Motion Seq.	150s	600s	$\{[6$ demographical, 12 sensor features$]_i, t_i\}_{i=1}^N$	✓

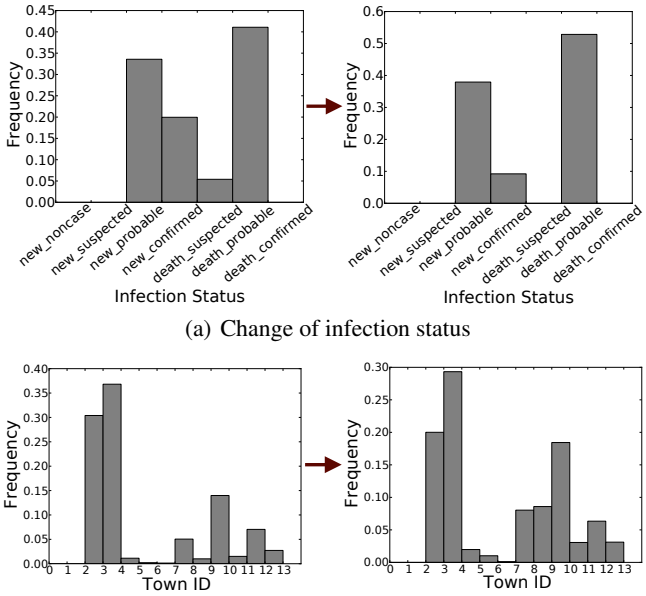
Table 1: *Summary of Datasets*

Baseline	Description
EMP	Defines the distance between segments based on the empirical data distribution $p(\mathbf{x}_j\|y)$ instead of $p(\tilde{x}_i\|y)$.
TopicM	Finds clusters of values using topic modeling instead of our IB-based data clustering.
LP	Finds the longest path instead of the ALP as the optimal segmentation.
Dynammo	Averaging data points in a sliding window to construct multi-dimensional time series, then feed the time series and the no. of cut points to *Dynammo*.

Table 2: *Baselines description.*

Dataset	**DASSA**	*TopicM*	*EMP*	*LP*	*Dynammo*
ChickenDance 1	**1**	0.85	0.76	0.63	0.57
ChickenDance 2	**1**	0.6	0.90	0.54	0.71
Portland	**1**	1	0.66	0	**1**
PUC-Rio	**0.66**	0.46	0.25	0.44	0.25

Table 3: F_1 score of DASSA, *TopicM*, *EMP*, *LP* and *Dynammo* on different datasets with ground-truth segmentation: DASSA gets perfect cuts in most of the datasets.



(a) Change of infection status

(b) Change of infection towns

Figure 5: *DASSA results for Ebola. (a) Distribution of infection status for the two segments detected. (b) Distribution of infection towns for the two segments detected.*

similar word clouds). In contrast, we find that *Dynammo* and *TopicM* fail to capture meaningful word transitions.

Ebola: We explore the feature values in the two segments detected by DASSA. In Fig. 5(a) we see that the death and newly confirmed cases reduce significantly from segment 1 to segment 2, which shows a sign of increased caution for the disease. We also notice from the change of distribution of towns (Fig. 5(b)) that at first the infection mostly occurs in town 2 and 3 which are "Kono" and "Kambia" in Sierra-Leone. Then it spreads to other towns (e.g. town 9 which is "Bo" in sierra-leone). DASSA *automatically* finds a segmentation that captures this disease propagation pattern; giving a better understanding of the situation.

7 Discussion

A segmentation algorithm implicitly contains its own distance measurements between time-segments. In this paper, we define the distance as a carefully designed metric between the associated 'co-occurrence' cluster distributions in the segments ($p(\tilde{\mathbf{x}}|y)$). One might naturally think to define the segment distance as the distance between the clusters in segments themselves; but doing so has multiple issues. As an example, we ran one classic subspace clustering algorithm (*Fires* (Kriegel et al. 2005)) on our datasets. We observe that *Fires* simply does not output any clusters for many segments (for example the last two segments in the optimal segmentation of *Argentina*, *Paraguay*, *Peru*), and it cannot detect the same good segmentations as DASSA does. We believe similar problems would happen to other traditional clustering algorithms as well. The cluster-based distance measurements intrinsically do not handle well datasets where there is no clustering. In addition, using the clusters themselves to represent the dataset will lose information as many data points are not in any of the clusters.

8 Conclusions

We introduce DASSA, a novel, general, self-guided and efficient algorithm, to automatically segment data sequences. We construct a segment-graph to efficiently represent and search among all possible segmentations. Then we propose an IB-MDL-based clustering algorithm to capture temporal similarities between data values. Finally, a novel DAG-ALP algorithm is present to automatically find the segmentation. DASSA has good performance on all datasets we col-

lect: discovering ground truth, finding high quality segmentations, and providing interpretable real patterns.

Our framework is general for segmentation problems and extending it for more complex sequences (such as image sequences) can be interesting future work. Future work can also look into a parallelized or online version of DASSA.

9 Acknowledgements

This paper is based on work partially supported by the NSF (IIS-1353346), the NEH (HG-229283- 15), ORNL (Order 4000143330) and from the Maryland Procurement Office (H98230-14-C-0127), and a Facebook faculty gift.

References

Amiri, S. E.; Chen, L.; and Prakash, B. A. 2017. Snapnets: Automatic segmentation of network sequences with node labels. In *AAAI*, 3–9.

Blei, D.; Carin, L.; and Dunson, D. 2010. Probabilistic Topic Models. *Signal Processing Magazine, IEEE* 27(6):55–65.

Chen, X. C.; Steinhaeuser, K.; Boriah, S.; Chatterjee, S.; and Kumar, V. 2013. Contextual time series change detection. In *SDM*.

Chen, L.; Hossain, K. S. M. T.; Butler, P.; Ramakrishnan, N.; and Prakash, B. A. 2014. Flu gone viral: Syndromic surveillance of flu on twitter using temporal topic models. ICDM.

Dhillon, I. S. 2001. Co-clustering documents and words using bipartite spectral graph partitioning. In *ACM SIGKDD*.

Grünwald, P. D. 2007. *The Minimum Description Length Principle (Adaptive Computation and Machine Learning)*. The MIT Press.

Hu, B.; Rakthanmanon, T.; Hao, Y.; Evans, S.; Lonardi, S.; and Keogh, E. 2011. Discovering the intrinsic cardinality and dimensionality of time series using mdl. In *ICDM*, 1086–1091. IEEE.

Kiernan, J., and Terzi, E. 2009. Constructing comprehensive summaries of large event sequences. *ACM Trans. Knowl. Discov. Data* 3(4).

Kriegel, H.-P.; Kroger, P.; Renz, M.; and Wurst, S. 2005. A generic framework for efficient subspace clustering of high-dimensional data. ICDM.

Li, L.; McCann, J.; Pollard, N. S.; and Faloutsos, C. 2009. Dynammo: Mining and summarization of coevolving sequences with missing values. In *KDD*.

Liu, L.; Tang, J.; Han, J.; and Yang, S. 2012. Learning influence from heterogeneous social networks. *Data Mining and Knowledge Discovery* 25(3):511–544.

Loglisci, C., and Berardi, M. 2006. Segmentation of evolving complex data and generation of models. In *ICDM Workshop*, 269–273. IEEE.

Madeira, S. C., and Oliveira, A. L. 2004. Biclustering algorithms for biological data analysis: a survey. *Computational Biology and Bioinformatics, IEEE/ACM Transactions on* 1(1):24–45.

Matsubara, Y.; Sakurai, Y.; and Faloutsos, C. 2014. Autoplait: Automatic mining of co-evolving time sequences. SIGMOD '14, 193–204.

Mueen, A., and Keogh, E. 2010. Online discovery and maintenance of time series motifs. KDD '10, 1089–1098.

Nguyen, M. H., and Torre, F. 2012. Maximum margin temporal clustering. In *International Conference on Artificial Intelligence and Statistics*, 520–528.

Nguyen, H.-V., and Vreeken, J. 2016. Linear-time detection of non-linear changes in massively high dimensional time series. In *SDM*. SIAM.

Patnaik, D.; Laxman, S.; Chandramouli, B.; and Ramakrishnan, N. ICDM '2012. Efficient episode mining of dynamic event streams.

Rosman, G.; Volkov, M.; Feldman, D.; Fisher III, J. W.; and Rus, D. 2014. Coresets for k-segmentation of streaming data. In *Advances in Neural Information Processing Systems*, 559–567.

Samé, A., and Govaert, G. Online Time Series Segmentation Using Temporal Mixture Models and Bayesian Model Selection. *ICMLA '12* 1:602–605.

Shokoohi-Yekta, M.; Chen, Y.; Campana, B.; Hu, B.; Zakaria, J.; and Keogh, E. Discovery of meaningful rules in time series. In *KDD'15*, 1085–1094.

Slonim, N., and Tishby, N. 2000. Document clustering using word clusters via the information bottleneck method. SIGIR.

Smola, A., and Narayanamurthy, S. 2010. An architecture for parallel topic models. *Proceedings of the VLDB Endowment* 3(1-2):703–710.

Tatti, N., and Vreeken, J. 2012. The long and the short of it: Summarising event sequences with serial episodes. KDD.

Thompson, W. W.; Comanor, L.; and Shay, D. K. 2006. Epidemiology of seasonal influenza: use of surveillance data and statistical models to estimate the burden of disease. *Journal of Infectious Diseases* 194(Supplement 2):S82–S91.

Tishby, N.; Pereira, F. C.; and Bialek, W. 1999. The information bottleneck method. In *Proceedings of the 37th Annual Allerton Conference on Communication, Control and Computing*, 368–377.

Toyoda, M.; Sakurai, Y.; and Ishikawa, Y. 2013. Pattern discovery in data streams under the time warping distance. *The VLDB Journal* 22(3):295–318.

Vijayasenan, D.; Valente, F.; and Bourlard, H. 2009. An information theoretic approach to speaker diarization of meeting data. *IEEE Transactions on Audio, Speech, and Language Processing* 17(7):1382–1393.

Waggoner, J.; Wang, S.; Salvi, D.; and Zhou, J. 2013. Handwritten text segmentation using average longest path algorithm. WACV.

Wu, C.-W.; Lin, Y.-F.; Yu, P. S.; and Tseng, V. S. 2013. Mining high utility episodes in complex event sequences. KDD '13, 536–544.

Yang, J.; McAuley, J. J.; Leskovec, J.; LePendu, P.; and Shah, N. 2014. Finding progression stages in time-evolving event sequences. In *WWW '14*.