Online Discovery of Group Level Events in Time Series

Xi C. Chen *
chen@cs.umn.edu

Abdullah Mueen †
mueen@cs.unm.edu

Vijay K Narayanan ‡
vkn@microsoft.com

Nikos Karampatziakis ‡
nikosk@microsoft.com

Gagan Bansal‡
Gagan.Bansal@microsoft.com

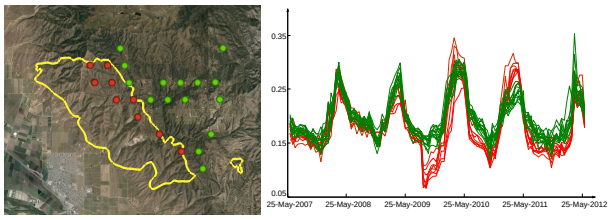
Vipin Kumar*
kumar@cs.umn.edu

Abstract

Recent advances in high throughput data collection and storage technologies have led to a dramatic increase in the availability of high-resolution time series data sets in various domains. These time series reflect the dynamics of the underlying physical processes in these domains. Detecting changes in a time series over time or changes in the relationships among the time series in a data set containing multiple contemporaneous time series can be useful to detect changes in these physical processes. Contextual events detection algorithms detect changes in the relationships between multiple related time series. In this work, we introduce a new type of contextual events, called group level contextual change events. In contrast to individual contextual change events that reflect the change in behavior of one target time series against a context, group level events reflect the change in behavior of a target group of time series relative to a context group of time series. We propose an online framework to detect two types of group level contextual change events: (i) group formation (i.e., detecting when a set of multiple unrelated timeseries or groups of time series with little prior relationship in their behavior forms a new group of related time series) and (ii) group disbanding (i.e., detecting when one stable set of related time series disbands into two or more subgroups with little relationship in their behavior). We demonstrate this framework using 2 real world datasets and show that the framework detects group level contextual change events that can be explained by plausible causes.

1 Introduction

Recent advances in high throughput data collection and storage technologies have led to a dramatic increase in the availability of high-resolution time series data sets in many areas including remote sensing, structural and functional brain data in magnetic resonance imaging, and in finance. Mining these time series provide important clues on the nature of the physical processes generating this data. For example, a change in the temporal pattern of precipitation may indicate a climate event (e.g., drought), while a change in the pattern of vegetation coverage over time might be an indicator of changes in the underlying land use (e.g., change in land cover type from forests to urban). As a result, change detection in time series is an active topic in data mining with applications in a variety of domains.



(a) **(b)**

Figure 1: A set of EVI time series which disbands in August 2009 because of forest fire. Such disbanding pattern is useful to detect events from time series datasets. Red and green points show the time series of points inside (marked as the red locations) and outside (marked as the green locations) the fire affected region, respectively.

The relationships among multiple contemporaneous time series in a dataset are guided by processes generating their corresponding signals. Time series generated by related physical processes often demonstrate high coherence while time series generated by independent processes are expected to show low coherence. Discovering *changes in the relationships* among the time series in a dataset containing multiple contemporaneous time series can be useful to detect events that change the underlying process(es) that generate these time series. For example, greenness of a patch of vegetation is determined by many factors including the type of vegetation, temperature, soil-moisture etc. The time series of greenness at geographically proximal locations will show highly coherent behavior if they are covered by the same type of vegetation. Fig. 1 shows a group of such locations and their corresponding time series from year 2007 to year 2009. Their greenness" time series (called

*University of Minnesota.
†University of New Mexico.
‡Microsoft Corporation

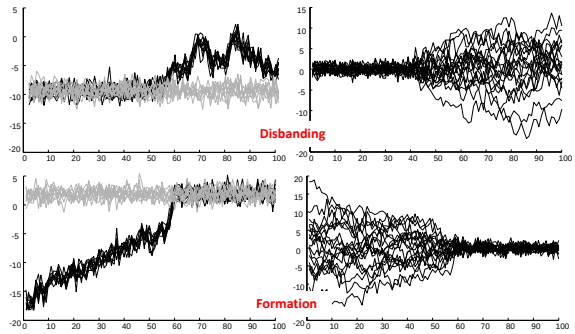


Figure 2: Illustration of two types of group level contextual changes. Top: Group disbanding events. Bottom: Group formation events.

as Enhanced vegetation index or EVI) are very similar to each other during normal years (year 2007 and year 2008). In year 2009, some of these locations (marked by red dots) were burned because of a natural forest fire, leading to a change in the relationship of EVI time series of some nearby locations. It is easy to see two distinct groups of time series (shown in red and green) after the forest fire. All the red time series (whose EVI values dipped in tandem as they are all affected by the fire) are located inside the burnt region while all the green time series (whose EVI values remain relatively unaffected by the fire) are located outside the region of the fire. Hence, by discovering this group level contextual change in the EVI spatio-temporal dataset, we are able to infer when (the time) and where (the geographical locations) the forest fire event occurred.

In this paper, we define group level contextual change detection which is a new type of change detection method for time series data. We focus on detecting two types of group level contextual change events, as shown in Fig. 2: (i) group formation, where a set of multiple unrelated time series (bottom right panel) or subgroups of time series with little correlated behavior (bottom left panel) merge to form a new group of time series with similar behavior at a later time, and (ii) group disbanding, where one set of time series with correlated behavior at some time disbands into two or more subgroups with little correlation (top left panel) or dissipates into uncorrelated individual components (top right panel) at a later time. We propose an online framework to detect these two types of group level contextual change events, present the results of applying the the group level contextual change detection framework on real datasets from 2 different domains that detect events with plausible real world causes, and propose two enhancements to speed up the group detection method in this framework.

The structure of this paper is as follows. We give a formal definition of group level contextual changes in Section 2 and present the related work in Section 3. In Section 4, we propose an online framework to detect group level contextual change events. Besides, two technology is introduced to our framework in order speed up the proposed framework. Section 5 first compares the scalable algorithm with the original algorithm and then presents the results of our experiments using 2 real world datasets. Section 7 concludes the paper with a discussion of future work.

2 Definition and problem formulation

In this section, we formally define group level contextual changes. We first introduce the concept of a contextual change. A contextual change refers to a deviation in the behavior of one or a group of time series with respect to its context, where the context is a collection of time series that behaves similar to the target group of time series over a time window.

DEFINITION 2.1. (CONTEXTUAL CHANGES) *Let* $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$ *be a set of* n *time series and let* $\mathcal{Y}^1$ *and* $\mathcal{Y}^2$ *be two mutually disjoint subsets of* $\mathcal{X}$. $\mathcal{X}$ *changes contextually between* t_1 *and* t_2 *if*

- *The time series in* $\mathcal{Y}^1$ *and* $\mathcal{Y}^2$ *exhibit similar behavior at time* t_1 *or* t_2.
- $f_d(\mathcal{Y}^1, \mathcal{Y}^2, t_1) \neq f_d(\mathcal{Y}^1, \mathcal{Y}^2, t_2)$

where $f_d(\mathcal{Y}^1, \mathcal{Y}^2, t)$ *is a function that measures the similarity of two sets* $\mathcal{Y}^1$ *and* $\mathcal{Y}^2$ *at time* t.

In general, the function f_d is determined by two components: (i) a dissimilarity metric (d) to measure the difference between two time series and (ii) a function (f) to tell the difference between two sets of time series based on this dissimilarity metric.

The choice of dissimilarity metric $d(.)$ and function $f(.)$ is governed by the desired type of change detection. For example, we use Pearson's correlation coefficient as the dissimilarity metric in the event detection for finance data, and Euclidean distance in the event detection using remote sensing data. There are multiple choices for $f(.)$ as well. The commonly used ones are mean, median, maximum and minimum. We define a function called similarity-aware entropy (which will be introduced in Section 4.2) as our choice of $f(.)$.

Chen et al. [6] proposed a framework to detect one type of contextual change. We refer to that framework as individual contextual event detection because it involves only one target time series. Specifically, they detect contextual changes assuming that $\mathcal{Y}^1$ contains only one time series. In this paper, we relax this constraint and propose a new type of contextual change called GROUP LEVEL CONTEXTUAL CHANGES where the

target $\mathcal{Y}^1$ can be a set of time series. We define two types of Group level contextual change events called GROUP DISBANDING and GROUP FORMATION.

DEFINITION 2.2. (GROUP LEVEL CONTEXTUAL CHANGES) *Let $\mathcal{X}$ be a time series set that changes contextually between t_1 and t_2 and t be the time when all the individual time series elements in the set $\mathcal{X}$ are in the same cluster (i.e have a small distance between each other). We call the event as*

- GROUP DISBANDING *when* $t = t_1$
- GROUP FORMATION *when* $t = t_2$.

3 Related work

There has been a wide array of "change detection" algorithms in the literature, many of which are on univariate time series detecting break points where the predefined features (e.g., mean, variance and any statistics calculated from the given time series) change [1, 2, 5, 10, 11]. For example, CUSUM searches for break points when the means of the subsequences shift; BIFAST detects a change point when the coefficients of subsequence models change; GPchange [4] considers variation in auto-covariance of a time series as "change."

We can classify all the above work, in general, as traditional algorithms for time series change detection in that they detect changes in some characteristic of the same time series over time. Different from change detection algorithms that consider auto-correlation in the same time series, contextual change detection algorithms detect events that change cross-correlation. Contextual changes refers to a deviation in the behavior of *one or a group* of time series (called the target set of time series) with respect to its context, where the context is defined as another collection of time series that exhibit similar behavior to the target set of time series over a period of time. [6] proposed a framework to detect one type of contextual change which involves only one target time series. Therefore, we call the method in [6] as individual contextual change detection as opposed to the group level contextual changes discussed in this paper. Previous works have considered defining the context based on other source of information such as event log [9], geo-location, etc. Our method builds context entirely using the time series based on similarity between the time series.

Our algorithm uses dynamic clustering of the time series. We perform density based clustering as opposed to methods that detect only spherical clustering. We do not assume that the time series are either stationary, periodic etc. [14]. We re-cluster the set of time series at every time instance and therefore, our method can detect both insertion (i.e. formation) into clusters, as well as deletion (i.e. disbanding) of clusters. Methods that maintain clusters only after insertions [3] are not suitable for our purpose. Finally, we cluster the time series using both correlation and Euclidean distances without normalization for different datasets [12].

4 Online Framework for Group Level Contextual Change Detection

Fig. 3 shows the proposed online framework to detect group level contextual events. As new observations are collected, the method performs three steps; (i) group time series according to their similarity (Section 4.1), (ii) calculate event scores at the current time based on the updated groups (Section 4.2), and (iii) report group level events using the event scores obtained (Section 4.3).

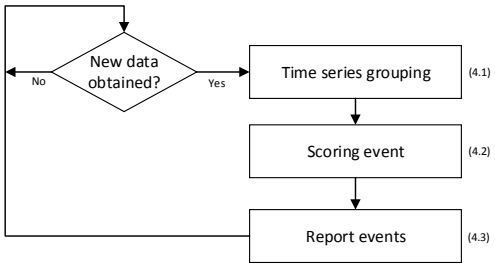


Figure 3: Flowchart illustrating the key steps in the proposed group level contextual change event detection method: (i) Grouping time series by similarity (ii) Scoring events and (iii) Reporting events based on the event scores

4.1 Grouping time series: AutoDBSCAN In this section, we propose a method called AutoDBSCAN for automatically clustering subsequences from each time series (over the same temporal window) into groups based on their (domain dependent) similarity functions. Since we use a financial dataset and a remote sensing dataset in our experimental results, we demonstrate the method using two similarity functions viz., Pearson's correlation (for the currency exchange rate dataset) and Euclidean distance (for the remote sensing data). Our framework can be easily extended to work with other similarity functions.

4.1.1 AutoDBSCAN AutoDBSCAN (shown in Algorithm 1) is an online method to detect group level contextual changes in a large dataset. It starts by detecting clusters using DBSCAN with a maximum value of neighborhood size (ϵ_2) and iteratively searches for higher density clusters, progressively reducing the neighborhood size by δ at each iteration, till either a minimum neighborhood size is reached, or there are no more higher density clusters detected by DBSCAN. Its advantages in contextual group level change detection

Algorithm 1: AutoDBSCAN

Input: data, ϵ_1, ϵ_2, minPts, δ
Output: Clusters

```
Clusters ← DBSCAN(data,ε2 ( > ε1), minPts);
if ε2 > ε1 and |Clusters| > 1 then
    for each non-noise cluster c in Clusters do
        dataT ← time series in c;
        output c;
        AutoDBSCAN(dataT, ε1,ε2 − δ, minPts) ;
    end
end
```

are as below.

1. It can be used even when several time series do not belong to any group, which is a common occurrence in real applications.
2. It returns all possible groups with different densities (above a threshold)
3. It is computationally efficient.

In real datasets, many time series do not fall into any cluster. Hence, we need to form an "unclustered-set" that includes all these unclustered time series. AutoDBSCAN utilizes DBSCAN [8] to form the contexts or groups and hence is a density based grouping method. By definition, it reports points in low-density regions as noise. These time series points form the "unclustered-set".

By Definition 2.2, if a set of objects forms a group or a group of objects splits into multiple groups, any valid algorithm should report it as a group level contextual event. Hence, we need to extract groups with different density. DBSCAN clusters objects with fixed density that is given by two parameters (i) the size of the neighborhood (ϵ) of each point, and (ii) the minimum number of points within the $\epsilon-$neighborhood of a point required to form a cluster ($minPts$). A naive method to modify DBSCAN for contextual event detection is to run it using multiple thresholds. We call this method as MultiDBSCAN. The proposed AutoDBSCAN is equivalent to MultiDBSCAN but computationally more efficient. Next, we first describe a containment property of DBSCAN that shows the equivalence of AutoDBSCAN and MultiDBSCAN. Then, we compare their computational cost.

PROPERTY 4.1. *(Containment Property) Let $\mathcal{A}_1, \mathcal{A}_2, \cdots, \mathcal{A}_m$ and $\mathcal{B}_1, \mathcal{B}_2, \cdots, \mathcal{B}_n$ be clusters discovered by DBSCAN using neighborhood size ϵ_a and ϵ_b, respectively (with $\epsilon_a < \epsilon_b$). Then for any $i \in [1, \cdots, m]$, there is a $j \in [1, \cdots, n]$ such that $\mathcal{A}_i \subset \mathcal{B}_j$.*

Property 4.1 means that any cluster $\mathcal{A}$ detected using a smaller ϵ is a subset of a cluster $\mathcal{B}$ detected using a larger ϵ. Therefore, after obtaining $\mathcal{B}$ using a higher ϵ, $\mathcal{A}$ can be obtained by searching within $\mathcal{B}$. Hence, MultiDBSCAN and AutoDBSCAN provide same outputs.

The computational cost of DBSCAN is $O(n^2w)$ where n is the number of time series and w is the window size used to compute the dissimilarity between subsequences. MultiDBSCAN executes DBSCAN for $J = \lfloor \frac{\epsilon_2 - \epsilon_1}{\delta} \rfloor$ times and its cost would be $O(n^2wJ)$. AutoDBSCAN uses Property 4.1 to reduce this cost. For example, the computational cost at the second iteration (i.e. for $\epsilon_2 - \delta$) is $O(\sum_{i=1}^{k} n_i^2 \times w)$, where k is the number of clusters detected by DBSCAN in the first iteration and n_i is the number of time series in the ith cluster. Since $\sum_{i=1}^{k} n_i^2 < (\sum_{i=1}^{k} n_i)^2$ and $\sum_{i=1}^{k} n_i << n$ in many applications due to the large number of "unclustered" time series. Hence, the computation cost of AutoDBSCAN is significantly lower than MultiDBSCAN.

4.1.2 Scalable Region Query for AutoDBSCAN

A straightforward implementation of AutoDBSCAN in C++ language took more than 20 hours to process a patch of 5,000 EVI time series over a mere 200 time steps (at this geographical resolution, the entire earth generates over 150 millions of EVI time series). AutoDBSCAN iteratively runs DBSCAN multiple times in different subsets of the dataset and the computational cost is dominated by searching for $\epsilon-$neighbors (See [7] for detailed explanations). We use two techniques to speed up $\epsilon-$neighbors searching function (the regionQuery algorithm described in Algorithm 2), that makes AutoDBSCAN fast enough to be used in online applications. In the following section, we denote $\mathbf{D}$ as the dataset and $\mathbf{D}(\mathbf{i}, :)$ as its ith object.

Indexing-technique Indexing-technique builds an index to speed up the algorithm for searching $\epsilon-$neighbors. This method can be used on all types of datasets for which indices can be built, including time series datasets. A few preprocessing steps are needed to create an index. These steps are: (1) Select one time series ($\mathbf{r}$, say the first time series) from the dataset $\mathbf{D}$. (2) Calculate the distance $d(.,.)$ between $\mathbf{r}$ and all the other time series. This is an $O(nw)$ operation. (3) Reorder all time series based on the distance calculated from step 2. This is an $O(n \log n)$ operation.

We refer to this preprocessed dataset as a reordered dataset. This preprocessing adds two properties to the reordered dataset, that are useful for speeding up the algorithm.

PROPERTY 4.2. *For any i and j such that $i > j$, in the reordered dataset, $d(i, r) > d(j, r)$.*

PROPERTY 4.3. *For any positive integer a, the re-*

ordered dataset satisfies the following two properties.

- *If* $d(i,r) - d(i-a,r) > \epsilon$, *then* $d(i,j) > \epsilon$ *for any* $j \leq i-a$.
- *If* $d(i+a,r) - d(i,r) > \epsilon$, *then* $d(i,j) > \epsilon$ *for any* $j \geq i+a$.

We use the reordered dataset as follows. Let us assume that the rank of the given time series in the reordered dataset is k. We test against its neighbors in the reordered dataset with ranks $k-1, k-2, \ldots,$ until $d(k,r) - d(k-a,r) > \epsilon$ is satisfied. Similarly, we test against its neighbors in the reordered dataset with ranks $k+1, k+2, \ldots$ until $d(k+a,r) - d(k,r) > \epsilon$ is satisfied. The Pseudocode for Indexing DBSCAN is given in Algorithm 2

Algorithm 2: regionQuery function Used in Indexing DBScan

```
   Input: k, ε, D(:, r)
   Output: NeighborPts
 1 i = k-1;
 2 while D(k,r) − D(i,r) < ε do
 3 |   Calculate D(i,k) ;
 4 |   if d < ε then
 5 |   |   Add the i^th time series into NeighborPts;
 6 |   end
 7 |   i = i − 1;
 8 end
 9 i=k+1;
10 while D(i,r) − D(k,r) < ε do
11 |   Calculate D(i,k) ;
12 |   if d < ε then
13 |   |   Add the i^th time series into NeighborPts;
14 |   end
15 |   i = i+1;
16 end
17 return NeighborPts;
```

Iterative-technique In Algorithm 2, the distance computation in preprocessing step 2 is $O(nw)$. Using Iterative-technique, we iteratively compute the current distance value using the distance calculated from the previous time step.

Assume that two time series $\mathbf{x(t)}$ and $\mathbf{y(t)}$ are collected from time step t to time step $t+w$. Let $\mu_x(t) = \sum_{i=t}^{t+w} x_i$, $\mu_y(t) = \sum_{i=t}^{t+w} y_i$, $S_x(t) = \sum_{i=t}^{t+w} x_i^2$, $S_y(t) = \sum_{i=t}^{t+w} y_i^2$ and $P_{x,y}(t) = \sum_{i=t}^{t+w} x_i y_i$. The Pearson's correlation value between $\mathbf{x(t)}$ and $\mathbf{y(t)}$ ($corr(\mathbf{x(t)}, \mathbf{y(t)})$) can be calculated using the following formula.

$$corr(\mathbf{x(t)}, \mathbf{y(t)}) = \frac{P_{x,y}(t) - \mu_x(t)\mu_y(t)/w}{\sqrt{S_x(t) - \mu_x(t)^2/w}\sqrt{S_y(t) - \mu_y(t)^2/w}}$$

Similarly, Euclidean distance can be calculated as

$$d^2(\mathbf{x(t)}, \mathbf{y(t)}) = S_x(t) + S_y(t) - 2P_{x,y}(t)$$

To compute correlation between $\mathbf{x(t)}$ and $\mathbf{y(t)}$ in an online manner, we update $\mu_x(t)$, $\mu_y(t)$, $S_x(t)$, $S_y(t)$ and $P_{x,y}(t)$ using values $\mu_x(t-1)$, $\mu_y(t-1)$, $S_x(t-1)$, $S_y(t-1)$ and $P_{x,y}(t-1)$. For example, $\mu_x(t)$ can be maintained iteratively (see [13][14] for details).

4.2 Scoring events using similarity-aware entropy The second step in the framework assigns an event score to each time series group ($\mathcal{X}$) discovered by AutoDBSCAN at time t. According to Definition 2.2, a function $f_d(.)$ that measures the similarity of two time series groups is required in group level contextual change detection. As mentioned in Section 2, the choice of f_d itself is determined by two components (i) dissimilarity metric (d) that measures the difference between two time series and (ii) a function (f) that tells the dissimilarity between two sets when d is given. The choice of d is generally guided by the application domain (e.g., we use Pearson's correlation coefficient is commonly in the finance dataset and Euclidean distance when analyzing the remote sensing dataset below). In this section, we will introduce a new event scoring method called similarity-aware entropy score. It is generalized from traditional entropy and has the following properties that make it useful in group level contextual change detection.

1. It converges to traditional entropy when the distance between groups is very large.
2. It is sensitive to the distance between different clusters and hence highlights events that contain significantly different subgroups.
3. It is calculated directly from pair-wise distances between all subsequences, whereas discrete group membership (or cluster ID) of each time series is required when calculating traditional entropy.

In the remainder of this section, we will describe similarity-aware entropy, demonstrate its properties that are useful for detecting group level changes, and also describe how to estimate the similarity-aware entropy score from clusters of time series.

The outputs of AutoDBSCAN are groups of subsequences within a certain time window, where the subsequences within a group are similar to each other according to a dissimilarity metric. Let us assume a set of time series $\mathcal{X}$ are discovered as a cluster within a time window $\mathbf{t} = \{t, t+1, \cdots, t+b\}$. A valid scoring method should score high if time series in $\mathcal{X}$ cannot be considered as a cluster (or a group) before t (for group formation score) or after $t+b$ (for disbanding score).

Entropy is a plausible candidate scoring function. Let us denote by $\mathbf{t}$ the time window when group $\mathcal{X}$ is discovered and use $\mathbf{t}'$ as the time period to check whether or not a group level event happens.

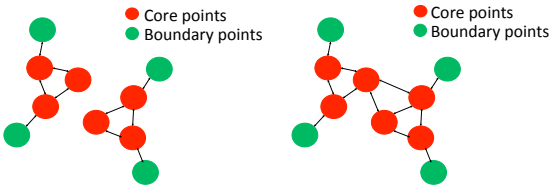


Figure 4: Sensitivity of the clustering results to the score based on entropy. (a) The 10 points are clustered into 2 groups and hence the entropy for these points is 1. (b) All the 10 points are clustered together in 1 group and hence the entropy for these points is 0.

Let $\mathcal{Y}^1, \mathcal{Y}^2, \cdots, \mathcal{Y}^k$ be mutually disjoint subsets of $\mathcal{X}$ such that within $\mathbf{t}'$ time series in the same subset are "similar" to each other while time series belonging to different subsets are dissimilar. Let us assume that the set $\mathcal{X}$ has n time series and $\mathcal{Y}^i$ contains n_i time series for any $i \in \{1, 2, , \cdots, k\}$. Then, the entropy of $\mathcal{X}$ in time period $\mathbf{t}'$ is given as

$$H(\mathcal{X}|\mathcal{Y}^1, \mathcal{Y}^2, \cdots, \mathcal{Y}^k) = -\sum_{i=1}^{k} p_i \log p_i$$

where $p_i = n_i/n$. However, there are two challenges when using entropy as the event score.

First of all, entropy requires the group membership (or cluster ID) of each time series during the given time period. Practically, clustering algorithm is used to obtain such information. Hence, scores based on entropy are sensitive to not only which clustering method is chosen but also the parameters selected for the clustering method. For example, let us assume that DBSCAN method is used to identify cluster IDs for all subsequences. Fig. 4 shows the clustering results of 10 points (subsequences) that were in the same group in an earlier time window, for two different values of ϵ (and keeping $minPts = 3$ in both clustering). Two equal sized clusters are discovered for a smaller value of ϵ (Fig. 4a), while all 10 points are clustered together for a larger value of ϵ (Fig. 4b) Hence, the entropy of these points is 1 for the smaller ϵ, which indicates that the cluster has split into 2 distinct clusters, while the entropy is 0 for the larger ϵ, which indicates that there is no group level event. Note that similar problems will be faced for most of clustering methods.

Second, the entropy score only considers cluster membership but is not aware of the distance among clusters. Fig. 5 shows the current observations of two sets of points. Each of the points is in a dense cluster before their current observations. Assuming that points in both figures are clustered into two groups based on some algorithm as shown in Fig. 5, identical entropy scores are provided for both of the two scenarios. However, we can observe that the groups in Fig. 5b are split more significantly compared with the groups in Fig. 5a.

We now propose a new event score called similarity-aware entropy, that is inspired by entropy but does not suffer from the above two limitations.

Definition 4.1. (Similarity-aware entropy) *Let $\mathcal{Y}^1, \mathcal{Y}^2, \cdots, \mathcal{Y}^k$ be mutually disjoint subsets of $\mathcal{X}$ such that during time period $\mathbf{t}'$, time series in the same subset are "similar" to each other and also time series from different subsets are different from each other. Assume that the set $\mathcal{X}$ has n time series and $\mathcal{Y}^i$ contains n_i time series for any $i \in \{1, 2, , \cdots, k\}$ and let $p_i = n_i/n$. The similarity-aware entropy of $\mathcal{X}$ during $\mathbf{t}'$ is defined as*

$$E(\mathcal{X}) = -\sum_{j=1}^{k} p_j \log \sum_{i=1}^{k} p_i e^{-d(\mathcal{Y}^i, \mathcal{Y}^j)}$$

where $d(\mathcal{Y}^i, \mathcal{Y}^j)$ measures the distance between two well separated clusters.

We present two properties of similarity-aware entropy. Property 4.4 illustrates the relationship between similarity-aware entropy and traditional entropy. Property 4.5 provides an opportunity to design a score that can overcome the limitations of using traditional entropy in scoring group level events.

Property 4.4. *Let $\mathcal{Y}^1, \mathcal{Y}^2, \cdots, \mathcal{Y}^k$ be mutually disjoint subsets of $\mathcal{X}$ such that for any $i, j \in [1, \cdots, k]$, $d(\mathcal{Y}^i, \mathcal{Y}^j) = \infty$ when $i \neq j$, and $d(\mathcal{Y}^i, \mathcal{Y}^j) = 0$ when $i = j$). Entropy of $\mathcal{X}$ based on $\mathcal{Y}^1, \mathcal{Y}^2, \cdots, \mathcal{Y}^k$ is identical to its similarity-aware entropy. That is,*

$$E(\mathcal{X}) = H(\mathcal{X}|\mathcal{Y}^1, \mathcal{Y}^2, \cdots, \mathcal{Y}^k)$$

Property 4.4 shows the relationship between traditional entropy (H) and similarity-aware entropy (E). Intuitively, if a set of objects $\mathcal{X}$ can be divided into k dense groups $\mathcal{Y}^1, \mathcal{Y}^2, \cdots, \mathcal{Y}^k$ such that $\mathcal{Y}^i$ and $\mathcal{Y}^j$ are far away

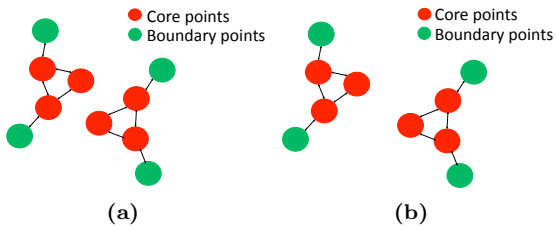


Figure 5: Both panels show 10 points that belong to the same cluster before the current observation. Assuming that a clustering method discovers 2 clusters in both, event scores based on entropy are identical in the two scenario. However, compared with the event shown in panel (a), the event in panel (b) is more significant.

from each other when $i \neq j$, then its similarity-aware entropy is same as its traditional entropy.

PROPERTY 4.5. *Assume that $\mathcal{X}$ can be partitioned into $\mathcal{A}^1, \mathcal{A}^2, \cdots, \mathcal{A}^m$ or $\mathcal{B}^1$, $\mathcal{B}^2$, $\cdots$, $\mathcal{B}^n$. As long as all these subsets are pure (i.e., the entropy of $\mathcal{A}^i$ and $\mathcal{B}^j$ for $\forall i \in [1, m]$ and $\forall j \in [1, n]$ is zero, similarity-aware entropy of such two partitions are same. That is,*

$$E(\mathcal{X}|\mathcal{A}^1, \mathcal{A}^2, \cdots, \mathcal{A}^m) = E(\mathcal{X}|\mathcal{B}^1, \mathcal{B}^2, \cdots, \mathcal{B}^n)$$

According to Property 4.5, we consider each time series in the group as its own cluster and estimate the similarity-aware entropy score as follows.

DEFINITION 4.2. (SIMILARITY-AWARE ENTROPY SCORE) *Consider a set $\mathcal{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_m\}$ that contains m time series. Then, its similarity-aware entropy score S during time period $\mathbf{t}$ is defined as*

$$S(\mathcal{X}, \mathbf{t}) = -\sum_{j=1}^{m} \frac{1}{m} \log(\sum_{i=1}^{m} exp(-d(\mathbf{x}_i, \mathbf{x}_j, \mathbf{t}))\frac{1}{m})$$

where, $d(\mathbf{x}_i, \mathbf{x}_j, \mathbf{t})$ is the dissimilarity metric between time series $\mathbf{x}_i$ and $\mathbf{x}_j$ during time $\mathbf{t}$.

4.3 Reporting events We use the proposed similarity-aware entropy score for group level contextual change detection. In detail, let us assume that a set of time series $\mathcal{X}$ is detected as one group during time period $\mathbf{t} = \{t, t+1, \cdots, t+b\}$ and also define time period $\mathbf{t}' = \{t-b, t-b+1, \cdots, t\}$. By Definition 2.2 and Definition 4.2, we say $\mathcal{X}$ formed at time t if

$$S(\mathcal{X}, \mathbf{t}')/S(\mathcal{X}, \mathbf{t}) > th$$

Similarly, if $\mathbf{t}' = \{t+b, t+b+1, \cdots, t+2b\}$, we say that a group $\mathcal{X}$ is disbanding at $t+b$ if

$$S(\mathcal{X}, \mathbf{t}')/S(\mathcal{X}, \mathbf{t}) > th$$

5 Experimental results

In this section, we first compare the running time of the original DBSCAN with both the indexing technique and the index + iterative distance computation. Then, we describe two case studies in finance and earth science to demonstrate the importance of group level contextual changes. Due to the page limitation, we show one event for each case study that our algorithm detects and explain the events based on our investigation from other source of information. More systermatic validation results are included in the extended version [7].

5.1 Scalability Experiment We implemented the two techniques in C++ and run the code on a desktop computer with an Intel Xeon 3.10GHz processor and 8GB of RAM. We compared the running time of the original DBSCAN implementation with both the indexing technique and the indexing + iterative distance computations.

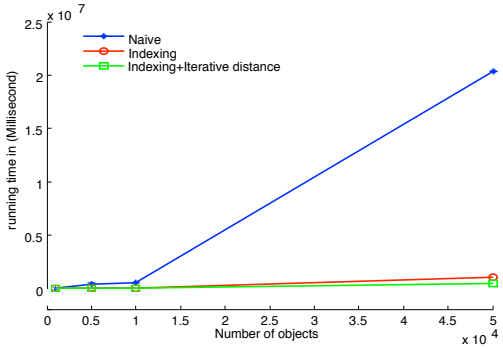


Figure 6: Running time of the original DBSCan implementation, the indexing and indexing + iterative distance computation implementations

Figure 6 shows the running time of the original DBSCAN implementation and the two optimized implementations. We see a speedup of up to 57× over the original implementation, which enabled us to search over more land area as well as larger correlation window in the experiments using EVI data.

5.2 Vegetation Index Data Group level contextual change detection can be useful for detecting various types land-cover changes. In this case study, we show an example that uses the proposed method to detect sudden change in land-cover by forest fire based on Enhanced Vegetation Index (EVI) time series.

In general, EVI is an indicator of "greenness" of the earth's surface, which can be used for forest fire detection (See [7] for detailed descriptions of the dataset). The idea is a fire would change the greenness of an area drastically and thus, EVI would drop significantly. A major unsolved challenge is to detect fire in non-forest vegetation, such as Shrubland and Grassland. The greenness of these vegetation highly depends on the condition of mesoscale and microscale meteorology (e.g., temperature and rainfall), which makes their EVI time series varying a lot. In addition, these land cover type recovers as fast as in 6 weeks. Hence, the drop in EVI due to fire can be insignificant compared with their nature variability.

Group level contextual change detection is a useful technique to distinguish forest fires from events due to mesoscale and microscale meteorology because in such events EVI time series of proximal patches show the same decrease in EVI (and are hence contextually unchanged), while forest fires affect limited geographical

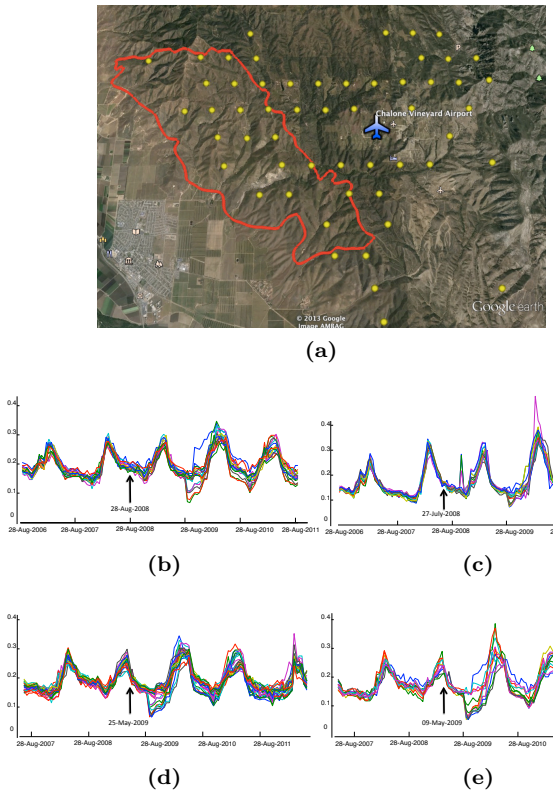


Figure 7: Four sets of EVI time series which disbands in August 2009 because of forest fire. Such disbanding pattern is useful to detect events from time series datasets.

regions and the EVI time series exhibit contextual changes with respect to the unaffected regions.

In this case study, we test the performance of our proposed framework in a region in California that is bounded by 36.5°N, 35.9°N, 121.2°W and 122°W and contains 3345 objects, each of which covers approximately 1 km^2 area. Twenty-five group disbanding events were detected for the period Aug 2008-May 2011 (See [7] for detailed analysis of the results).

Fig. 7 shows four of these group disbanding events that correspond to the same fire event. These four events are detected at different times. Fig. 7 (b) - (e) show EVI time series corresponding to these events. Event (b) and Event (c) indicated that two split events happened between Aug. 2008 and Aug. 2010 and Event (d) and Event (e) indicated that two split events happened between May 2009 and May 2011. In each of these events, EVI time series that show similar patterns over two years are grouped together and then they split into roughly two groups around Aug 2009, when the patch bounded within the red line was burned. We notice that all the time series that have low values around 28 Aug. 2009 are located within the burned patch.

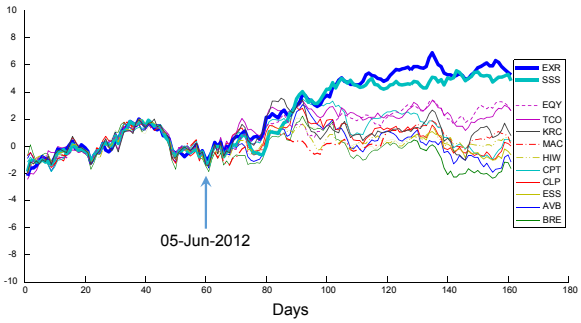


Figure 8: An event in the stock price data showing a change in the grouping of REITs. The two Self-service Storage companies forked out from the general REIT context.

Note that, the left side of the red polygon in Fig. 7 is farmland which has different greenness cycles than the natural vegetation. Our method does not include any location from the farmland into any of the four events, while it automatically includes the locations from the right side of the burned patch because they have vegetation similar to the one that was burned. (This demonstrates the efficacy of context building in our algorithm.) The airport shown in the figure is the Chalone Vineyard Airport region, which is different land-cover type than all the locations marked by yellow dots and is, again, not included in any of the groups for the same reason.

5.3 Stock price data We use a dataset of 5825 time series of the daily closing price of different stock tickers starting from April, 1996 till July, 2013 in the NYSE (See [7] for detailed descriptions of the dataset). We have run our algorithm for a window of 60 business days on this dataset to find historical contextual changes of stock tickers. One of the interesting change point we find is in June 2012 as shown in Figure 8. All the tickers are REITs (Real Estate Investment Trust). Two of the tickers significantly rise after June 2012 while, others remain within the context and show stability for more than six months after the event.

We investigate the fork and find the two rising time series are stock tickers from two self-service storage companies (EXR and SSS). The others are real estate companies in different parts of US and none of them does self-service storage business as per google finance. The event started at June 2012 that is the usual time of the year for publishing the quarterly/annual financial reports.

6 Conclusion and Future work

We have proposed a framework for detecting group level contextual changes in a dataset of multiple related time series. The framework uses 3 components viz.

(a) detect groups of time series (b) score events at each time instant and (c) report detected events. We also proposed 2 modifications to speed up the core AutoDBSCAN algorithm used in this framework. We demonstrated the results of applying this framework on 2 real world datasets, which detected plausible change events.

We identify the following four major items for future exploration. First, since AutoDBSCAN is based on DBSCAN, it can detect arbitrary-shaped clusters in which two members of the same cluster can be very different from each other. In the 2 applications presented in this paper, we did not encounter this problem. However, it is a potential risk when using this framework in other applications. A potential work-around is to add a constraint on the largest pair-wise distance within a cluster. A future extension of this work is to design a clustering method that returns all possible time series groups in spherical clusters. Second, the framework as presented here has 5 user defined parameters viz. (a) w: window size used in constructing time series groups, (b) $[\epsilon_1, \epsilon_2]$: the range of the size of the $\epsilon-$neighborhood in the AutoDBSCAN method to detect groups of time series, (c) δ: step size in ϵ used in AutoDBSCAN, (d) b: window size of event scoring interval, and (e) the threshold used to report events. (Of these (b) and (c) are specific to the AutoDBSCAN clustering algorithm chosen for illustration). In this paper, all these parameters are empirically chosen based on domain knowledge. Techniques to automatically determine (or at least estimate reasonable values) of these parameters will be helpful in applying this framework to new domains. Third, although the framework assigns a higher similarity aware score for more significant splits in group disbanding or tighter clusters in group formation, it would be useful to estimate a confidence value for the detected change event (for example, the confidence value for a event detected from a random walk dataset should be much lower than a event detected from a normal time series dataset). Finally, we could extend this framework to detect other interesting contextual events like *loops*, i.e., when one set of time series first disbands and then merges back. Such contextual events could provide interesting insights in fields of Ecology and Meteorology.

Acknowledgement

Part of this work was done when the first author was an intern in the Cloud & Information Services Lab in Microsoft. It was supported in part by the National Science Foundation under Grants IIS-1029711 and IIS-0905581, as well as the Planetary Skin Institute. Access to computing facilities was provided by the University of Minnesota Supercomputing Institute.

References

[1] M. Basseville and I. V. Nikiforov. *Detection of Abrupt Changes: Theory and Application*. Prentice Hall, 1993.

[2] S. Boriah. *Time Series Change Detection: Algorithms for Land Cover Change*. PhD thesis, University of Minnesota, 2010.

[3] F. Cao, M. Ester, W. Qian, and A. Zhou. Density-based clustering over an evolving data stream with noise. In *SDM*, 2006.

[4] V. Chandola and R. R. Vatsavai. A gaussian process based online change detection algorithm for monitoring periodic time series. *SIAM: SIAM International Conference on Data Mining (SDM11)*, 2011.

[5] J. Chen and A. Gupta. On change point detection and estimation. *Communications in Statistics: Simulation & Computation*, 30(3):665–697, 2001.

[6] X. C. Chen, K. Steinhaeuser, S. Boriah, S. Chatterjee, and V. Kumar. Contextual time series change detection. In *Proc. SIAM Intl. Conf. on Data Mining, 2013*, 2013.

[7] X. C. Chen, A. Mueen, V. K. Narayanan, N. Karampatziakis, G. Bansal, and V. Kumar. Online discovery of group level events in time series. Technical Report 14-004, Computer Science, University of Minnesota, 2014.

[8] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, pages 226–231, 1996.

[9] M. Gupta, A. B. Sharma, H. Chen, and G. Jiang. Context-aware time series anomaly detection for complex systems. In *SDM Workshop on Data Mining for Service and Maintenance (2013)*, SDM'13, 2013.

[10] V. Guralnik and J. Srivastava. Event detection from time series data. In *KDD '99: Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 33–42, 1999.

[11] T. L. Lai. Sequential changepoint detection in quality control and dynamical systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, 57(4): 613–658, 1995.

[12] Z. Li, B. Ding, J. Han, and R. Kays. Swarm: mining relaxed temporal moving object clusters. *Proc. VLDB Endow.*, 3(1-2):723–734, 2010.

[13] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. KDD '12, pages 262–270, 2012.

[14] P. Rodrigues, J. Gama, and J. Pedroso. Hierarchical clustering of time-series data streams. *Knowledge and Data Engineering, IEEE Transactions on*, 20(5):615–627, 2008.