Event Detection from Time Series Data

Valery Guralnik, Jaideep Srivastava
Department of Computer Science
University of Minnesota
{guralnik, srivasta}@cs.umn.edu

Abstract

In the past few years there has been increased interest in using data-mining techniques to extract interesting patterns from time series data generated by sensors monitoring temporally varying phenomenon. Most work has assumed that raw data is somehow processed to generate a sequence of *events*, which is then mined for interesting *episodes*. In some cases the rule for determining when a sensor reading should generate an event is well known. However, if the phenomenon is ill-understood, stating such a rule is difficult. Detection of events in such an environment is the focus of this paper. Consider a dynamic phenomenon whose behavior changes enough over time to be considered a qualitatively significant change. The problem we investigate is of identifying the time points at which the behavior change occurs. In the statistics literature this has been called the *change-point detection* problem. The standard approach has been to (a) *apriori* determine the number of change-points that are to be discovered, and (b) decide the function that will be used for curve fitting in the interval between successive change-points. In this paper we generalize along both these dimensions. We propose an iterative algorithm that fits a model to a time segment, and uses a likelihood criterion to determine if the segment should be partitioned further, i.e. if it contains a new change-point. In this paper we present algorithms for both the *batch* and *incremental* versions of the problem, and evaluate their behavior with synthetic and real data. Finally, we present initial results comparing the change-points detected by the batch algorithm with those detected by people using visual inspection.

1 Introduction

Sensor-based monitoring of any phenomenon creates time series data. The spacing between successive readings may be constant or varying, depending on whether the sampling is *fixed* or *adaptive*. The overall goal is to obtain an accurate picture of the phenomenon with minimum sampling effort. Examples of such observations include highway traffic monitoring, electro-cardiograms, and monitoring of oil refineries.

In the past few years there has been increased interest in using data mining techniques to extract interesting patterns from temporal sequences [SA95, MTV97, PT96]. A standard assumption has been that the raw data collected from sensors is somehow processed to generate a sequence of *events*, which is then mined for interesting *episodes* [MTV95, HKM+96]. While there is no strict definition of an episode, intuitively it is *a pattern of events occurring in some order, and close enough to each other in time.* Recent research has developed languages for specifying temporal patterns [MT96, PT96, GWS98], and algorithms have been proposed that takes advantage of the specified pattern to increase the computational efficiency of the mining process.

However, an issue that has received scant attention is of deriving an event sequence from raw sensor data. In some cases the rule for determining when a sensor reading should generate an event is well known, e.g. **if** *the temperature of a boiler goes above a certain threshold*, **then** *sound an alarm.* However, if the phenomenon is ill-understood or changes its behavior unpredictably, adapting the threshold such that event reporting is accurate becomes very difficult. Thus, a more systematic approach is required for processing the raw sensor data to generate an event sequence. This is the focus of our paper.

Consider a dynamic phenomenon whose behavior changes enough over time so as to be considered a qualitatively significant change. Each such change is an *event*. An example is the change of highway traffic from *light* to *heavy* to *congested*. Another example is the change of a boiler from *normal* to *super-heated*. The specific problem we address is of applying data mining techniques to identify the time points at which the changes, i.e. events, occur. In the statistics literature this has been called the *change point detection*

KDD-99 San Diego CA USA

problem. The standard approach has been to (a) *apriori* determine the number of change points to be discovered, and (b) decide the model to be used for fitting the subsequence between successive change points. Thus, the problem becomes one of finding the best set of the predetermined number of points that minimizes the error in fitting the pre-decided function [SO94, Hus93, Haw76, HM73, Gut74]. [KS97] addresses the problem of approximating a sequence of sensor readings by a set of k linear segments as a pre-processing step. This too can be considered a version of the change-point detection problem. In the proposed approach, we address both limitations of standard approaches. First, we place no constraint on the class of functions that will be fitted to the subsequences between successive change points. Second, the number of change points is not fixed *apriori*. Rather, the appropriate set is found by using *maximum likelihood methods* [Hud66].

In this paper we study two versions of the change point detection problem, namely the *batch* and the *incremental* versions. In the batch version the entire data set is available, as in the case of 24-hour data from traffic sensors, from which the best set of change points can be determined. In the incremental version, the algorithm receives new data points one at a time, and determines if the new observation causes a new change-point to be discovered. Our contributions include

- developing a general approach to change-point, i.e. event, detection that generalizes previous approaches,
- developing algorithms for both the batch and incremental versions of the change point detection problem,
- evaluating their behavior with synthetic and real data,
- and comparing the algorithms with visual change-point detection by humans.

This paper is organized as follows: In section 2 we formally describe the event detection problem. Section 3 presents the batch algorithm and Section 4 its performance. Section 5 describes the incremental algorithm, which is evaluated in the Section 6. Section 7 concludes the paper.

2 Event Detection as Change-Point Detection

In this paper we are interested in real-valued time series denoted by $y(t)$, $t = 1, 2, ..., n$, where t is a time variable. It is assumed that the time series can be modeled mathematically, where each model is characterized by a set of parameters. The problem of *event detection* becomes one of recognizing the change of parameters in the model, or perhaps even the change of the model itself, at unknown time(s).

This problem is widely known as the *change-point detection* problem in the field of statistics. A number of approaches have been proposed to solve the change-point detection problem [SO94, Hus93, Haw76, HM73, Gut74]. The standard assumption is that the phenomenon can be approximated by a known, stationary (usually linear) model. However, this assumption may not be true in some domains, creating the need for an approach that works without this assumption. In this paper we propose an approach that simultaneously addresses the issue of model selection and change-point detection.

2.1 Formal Statement of the Problem

Consider a time series denoted by $y(t)$, $t = 1, 2, ..., n$, where t is a time variable.

We would like to find a piecewise segmented model M, given by

$$\begin{aligned} Y &= f_1(t, \mathbf{w}_1) + e_1(t), \ (1 < t \leq \theta_1), \\ &= f_2(t, \mathbf{w}_2) + e_2(t), \ (\theta_1 < t \leq \theta_2), \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ &= f_k(t, \mathbf{w}_k) + e_k(t), \ (\theta_{k-1} < t \leq t(n)). \end{aligned}$$

An $f_i(t, \mathbf{w}_i)$ is the function (with its vector of parameters $\mathbf{w}_i$) that is fit in segment i. The θ_i's are the change points between successive segments, and $e_i(t)$'s are error terms. At this point we put no constraints on the nature of $f_i(t, \mathbf{w}_i)$'s.

2.2 Maximum Likelihood Estimation

If all change points are specified *a priori*, and models with parameters $\mathbf{w}_i$'s and estimated standard deviations σ_i's found for each segment, then the statistical likelihood L, of the change points is proportional to

$$L = \begin{cases} \prod_{i=1}^{k} \sigma_i^{-m_i} & - \textit{heteroscedastic error} \\ \left[\sum_{i=1}^{k} m_i \sigma_i^2\right]^{-n/2} & - \textit{homoscedastic error} \end{cases}$$

Here k is the number of change-points, m_i is the number of time points in segment i, and n is the total number of time points[1].

If the change points are not known, the maximum likelihood estimate (MLE) of the θ_i's can be found by maximizing the likelihood L over all possible sets of θ_i's, or equivalently, by minimizing $-2 \log L$. This function is equivalent to,

$$-2logL = \begin{cases} \sum_{i=1}^{k} m_i \log \sigma_i^2 & - \textit{heteroscedastic error} \\ n \log(\sum_{i=i}^{k} m_i \sigma_i^2) & - \textit{homoscedastic error} \end{cases}$$

[1]The homoscedastic error model specifies that $\sigma_1 = \sigma_2 = \ldots = \sigma_k$. Heteroscedastic error model doesn't impose this constraint.

In this paper, the term *likelihood criteria* will refer to function $-2\log L$, and will be denoted as $\mathcal{L}$. Because log is a monotonically increasing function, for the homoscedastic error case we use the equivalent likelihood criteria of minimizing the function $\sum_{i=1}^{k} m_i\sigma_i^2$.

2.3 Model Selection

For each segment i, model estimation is the problem of finding the function $\hat{f}_i(t, \mathbf{w}_i)$ that best approximates the data. The quality of an approximation produced by the learning system is measured by the loss function $Loss(y(t), \hat{f}_i(t, \mathbf{w}_i))$, where $\theta_{i-1} < t \leq \theta_i$. The expected value of loss is called risk functional $R_i(\mathbf{w}_i) = E[Loss(y(t), \hat{f}_i(t, \mathbf{w}_i)]$. Therefore, for each segment the learning system has to find a $\hat{f}_i(t, \mathbf{w}_i)$ that minimizes $R(\mathbf{w}_i)$.

Let us now consider the nature of $\hat{f}_i(t, \mathbf{w}_i)$'s. Most past work has assumed that the nature of these functions is known, or can be somehow determined from domain knowledge. However, in general this cannot be done, and thus our approach allows the possibility of arbitrary functions. To provide a handle on the problem, however, we use the key result of universal approximation theory, which states, that any continuous function can be approximated by another function from a given class [CM98]. The latter class can be considered as a basis class. An example of such a basis class is the set of algebraic polynomials $\{ t^0, t^1, t^2, \ldots \}$[2] [KC96].

For each of the segments, the learning machine should select a model that best describes the data. Various model selection methods have been proposed, e.g. analytical model selection via penalization and model selection via re-sampling [CM98]. The re-sampling approach has an advantage of making no assumptions on the statistics of the data or the type of target function being estimated. However, its main disadvantage is high computational effort. With linear regression it is possible to compute the leave-one-out cross-validation estimate of expected risk analytically [CM98]. This has computational advantages over the re-sampling approach, since repeated parameter estimation is not required. This is the approach used in the paper. Finally, the change-point likelihood also depends on the error model used. Unless there is a known fact that the error model is heteroscedastic, it is reasonable to assume the homoscedastic error model [Kue94], which is what we do.

[2]For practical reasons, there must be an upper bound on the degree of the polynomials in the basis class, say p-1. In general it is possible to use other basis classes, e.g. radial, wavelet, Fourier, etc. The choice of which basis class to use is itself an interesting problem, but outside our present scope. Note that the proposed approach can work with any of these basis classes.

3 Batch Algorithm

In this section we assume that the entire data set is collected before the analysis begins. In section 5 we consider the incremental case where change-point detection proceeds concurrently with data collection.

3.1 Algorithm Description

Change-point detection algorithms have been studied in the statistics literature [Haw76, HM73, Gut74]. They have worked under the assumptions that

(a) a stationary known model can be used to describe the phenomenon, and

(b) the number of change points is known apriori.

Our approach was to start from the algorithm described in [Haw76], and remove these assumptions.

Assume that the best model that maintains time points t_i, t_{i+1}, ..., t_j as a single segment has been selected. Let S be the residual sum of squares for this model. The number of points in this segment is $m = j - i + 1$. Let $\mathcal{L}(i,j) = m\log(S/m)$ if a heteroscedastic error model is used, and $\mathcal{L}(i,j) = S$ if the error model is homoscedastic.

The key idea behind the proposed algorithm is that at every iteration, each segment is examined to see whether it can be split into two significantly different segments. The splitting procedure can be illustrated by a consideration of the first stage, since all subsequent stages consist of equivalent scaled-down problems.

Let the data set cover the time points t_1, t_2, ..., t_n. The change point in the first stage is the j minimizing $\mathcal{L}(1,j) + \mathcal{L}(j+1,n)$, say j^*. Here j^* is defined as

$$\mathcal{L}(1,j^*) + \mathcal{L}(j^*+1,n) = \min_{p \leq j \leq n-p} \{\mathcal{L}(1,j) + \mathcal{L}(j+1,n)\}.$$

The range of j depends on the fact that at least p points are needed for model fitting in each segment. Further, the model fitted in each segment is the best possible from the space described by the basis functions, according to the model selection method used.

At the second stage, each of the two segments is analyzed as above and the best candidate change-points c_1 and c_2 of each are located. The better of these candidates is then selected, yielding a division of the original sequence into three segments. Without loss of generality let's assume that point c_1 is chosen: Now, the likelihood criteria of the model becomes

$$\mathcal{L} = (\mathcal{L}(1,c_1) + \mathcal{L}(c_1+1,j^*) + \mathcal{L}(j^*+1,n)) < (\mathcal{L}(1,j^*) + \mathcal{L}(j^*+1,c_2) + \mathcal{L}(c_2+1,n)).$$

The above procedure is repeated until a stopping criterion (described in section 3.2) is reached. Figures 1, 2, 3 provide the details of the algorithm.

The algorithm takes the set of approximating basis functions $MSet$ and the time series T

```
1.  new_change_point = find_candidate(T, MSet)
2.  Change_Points = ∅
3.  Candidates = ∅
4.  T1, T2 = get_new_time_ranges(T, Change_Points, new_change_point)
5.  while(stopping criteria is not met) do begin
6.      c1 = find_candidate(T1, MSet)
7.      c2 = find_candidate(T2, MSet)
8.      Candidates = Candidates ∪ c1
9.      Candidates = Candidates ∪ c2
10.     new_change_point = c ∈ Candidates|Q(Change_Points, c) = min
11.     Candidates = Candidates \ new_change_point
12.     T1, T2 = get_new_time_ranges(T, Change_Points, new_change_point)
13.     Change_Points = Change_Points ∪ new_change_point
14. end
```

Figure 1: Hierarchical Procedure To Detect Change Points

```
1.  optimal_likelihood_criteria = ∞
2.  for(i = p to |T| − p − 1) do begin
3.      likelihood_criteria = Find_Likelihood_Criteria(T[1, i], MSet) +
4.                            Find_Likelihood_Criteria(T[i + 1, |T|], MSet)
5.      if (likelihood_criteria < optimal_likelihood_criteria)
6.          split = T(i)
7.          optimal_likelihood_criteria = likelihood_criteria
8.      endif
9.  endfor
10. return split
```

Figure 2: Find_Candidate Algorithm

```
1.  minimum_risk = ∞
2.  for (each model M ∈ MSet) do begin
3.      model_risk = Risk(T, M)
4.      if(model_risk < minimum_risk)
5.          minimum_risk = model_risk
6.          likelihood_criteria = Fit(T, M)
7.      endif
8.  endfor
9.  return likelihood_criteria
```

Figure 3: Find_Likelihood_Criteria Algorithm

It should be noted that there have been other algorithms [HM73, Gut74] proposed to solve the change-point problem. We chose to modify a hierarchical solution because it is computationally more efficient.

3.2 Stopping Criteria

Since the number of change points is not known apriori, a stopping criterion must be used by the algorithm. In practice one would expect that once the algorithm has detected all "real" change-points, adding any more change points would not change the likelihood significantly. In fact, upon the addition of a sufficient number of spurious change-points, the overall likelihood value can increase, as illustrated in Figure 4. In successive iterations of the algorithm, at first the likelihood criteria decreases dramatically until it becomes stable, and then starts to increase slowly as spurious change-points are found. Therefore, the algorithm should stop when the likelihood criteria becomes stable or starts to increase. Formally, if in iterations k and $k+1$ the respective likelihood criteria values are $\mathcal{L}_k$ and $\mathcal{L}_{k+1}$, the algorithm should stop if

$$(\mathcal{L}_k - \mathcal{L}_{k+1})/\mathcal{L}_k < s,$$

where s is a user-defined stability threshold. When stability threshold s is set to 0%, the algorithm stops only when the likelihood criteria starts increasing.

4 Experimental Evaluation of the Batch Algorithm

We evaluated the behavior of our change-point detection algorithm on synthetic as well as real data from highway traffic sensors. In this section we present the

h	$s = 0\%$	$s = 5\%$
60	[1, 9], [10, 19], [20, 30], [31, 39]	[1, 9], [10, 19], [20, 30], [31,39]
30	[1, 9], [10, 19], [20, 29], [30, 39]	[1, 9], [10, 19], [20, 29], [30, 39]
20	[1, 9], [10, 20], [21, 29], [30, 39]	[1, 9], [10, 20], [21, 29], [30, 39]
15	[1, 9], [10, 20], [21, 30], [31, 39]	[1, 9], [10, 20], [21, 30], [31, 39]
10	[1, 9], [10, 19], [20, 29], [30, 39]	[1, 9], [10, 19], [20, 29], [30, 39]
8	[1, 11], [12, 20], [21, 30], [31, 39]	[1, 11], [12, 20], [21, 30], [31, 39]
5	[1, 11], [12, 20], [21, 29], [30, 35], [36, 39]	[1, 11], [12, 20], [21, 29], [30, 39]
2	[1, 11], [12, 15], [16, 25], [26, 31], [32, 39]	[1, 15], [16, 25], [26, 31], [32, 39]

Table 1: Experimental Results for Synthetic Data Sets

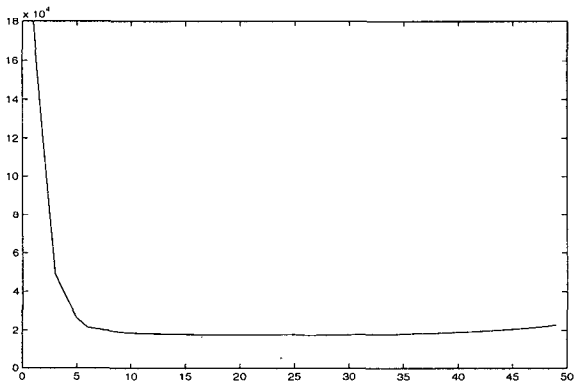

Figure 4: Likelihood criteria as a function of change-points

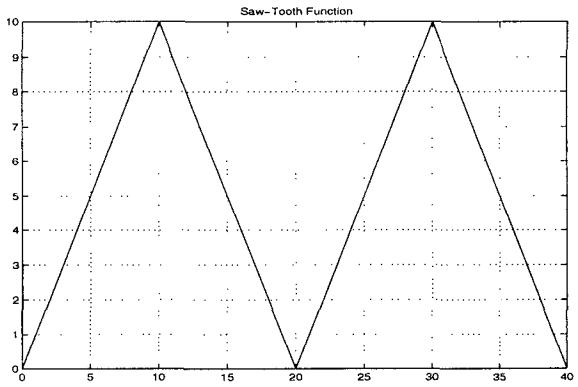

Figure 5: Saw-Tooth Function

results of these evaluations. In each case we measure the effectiveness of the algorithm, i.e. the quality of the change-points detected. For experimental purposes, the basis functions we selected were 1, t, t^2 and t^3. Note that our approach is general and can work with any class of basis functions.

4.1 Experiment with Synthetic Data

The data set consisted of 40 data points and was generated using the following saw-tooth function

$$f(t) = \begin{cases} t * h/10 + \epsilon & : \quad t \in [1, 9] \\ (20 - t) * h/10 + \epsilon & : \quad t \in [10, 19] \\ (t - 20) * h/10 + \epsilon & : \quad t \in [20, 29] \\ (40 - t) * h/10 + \epsilon & : \quad t \in [30, 39] \end{cases}$$

The noise ϵ is Gaussian with zero mean and unit variance. The height of the function h controls the signal-to-noise ratio. The larger the value of h, the greater the signal-to-noise ratio. An example of such a function (without noise) is depicted in Figure 5.

If the proposed algorithm is able to correctly identify all change-points, it should detect the following intervals: [1, 9], [10, 19], [20, 29], [30, 39]. However, due to the continuity of the saw-tooth function $f(t)$ at the change-points, a different set of change-points can also be detected. For example, the set [1, 10], [11, 19], [20, 29], [30, 39] is also a correct set of intervals. This is because $t = 10$ can be interpreted as the end of the current trend or the beginning of a new one. Similarly for $t = 20$ and $t = 30$.

The experiment was aimed at finding whether the method is able to correctly identify all change-points, and the sensitivity of the technique to the noise level. The results of the experiment are summarized in Table 1. As the signal-to-noise ratio decreases, the algorithm starts to give less accurate results. In this particular case the algorithm breaks at height $h = 2$. However, the algorithm works well for larger values of h. For $h \geq 8$, the algorithm identifies all change points without introducing false positives and false negatives.

The stability threshold, s, of the stopping criterion doesn't affect the results when the data set does not have a lot of noise. However, when the noise in the data set is increased, higher values of s prevent the algorithm from identifying false change-points. At height $h = 5$, when we increased the stability threshold from 0% to 5%, the algorithm was able to stop before falsely splitting the region [30, 39] into two regions [30, 35] and [36, 39].

4.2 Experiment with Traffic Data

The data used in our experiments was taken from highway traffic sensors, called *loop detectors*, in the

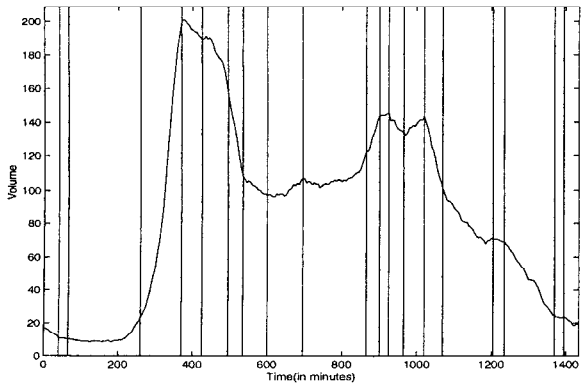

Figure 6: Data Set: V274

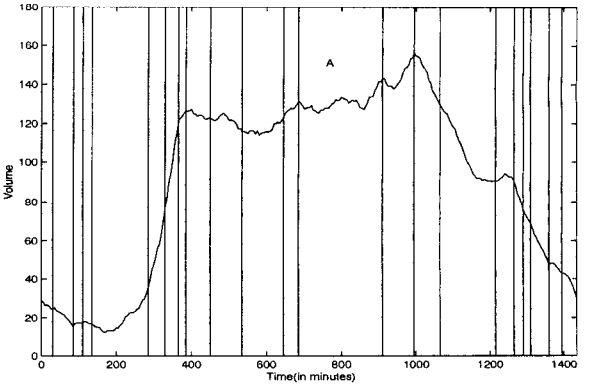

Figure 7: Data Set: V287

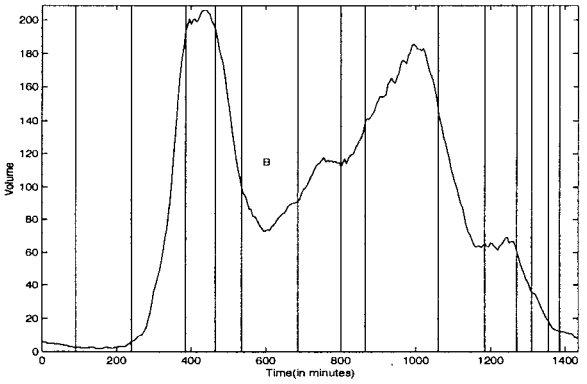

Figure 8: Data Set: V1101

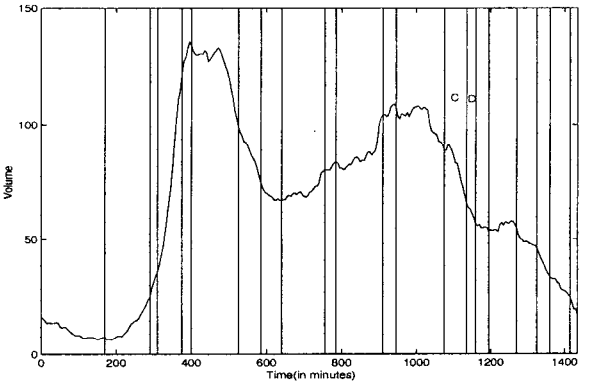

Figure 9: Data Set: V315

Minneapolis-St. Paul metro area. A loop detector is a sensor, embedded in the road, with an electro-magnetic field around it, which is broken when a vehicle goes over it. Each such breaking, and the subsequent re-establishment, of the field is recorded as a count. *Traffic volume* is defined as the vehicle count per unit time. In our data set the volume data was sampled at 5 minute intervals, i.e. the vehicle count was recorded at the end of a 5 minute interval and the counter was reset to 0. Each data set is a time sequence collected over a 24-hour time period, i.e. consisting of 288 samples.

We were interested in how our change point detection algorithm performed compared to a person doing the same task through visual inspection. The original data was very noisy, and thus in some cases it was difficult to visually detect the actual change points. Essentially, the data had a lot of small variations, which can potentially cause a human to observe microscopic trends that are not actually present. Based on our discussions with traffic engineers from the Minnesota Department of Transportation, i.e. the domain experts, we smoothed the data using a moving averages approach for visual inspection based change point detection by the human observer. Our algorithm was fed the original data set, i.e. without smoothing.

The proposed algorithm's behavior was evaluated on four different data sets, the results of which are shown in Figures 6, 7, 8, and 9. Each change point detected by the algorithm is based on the criteria defined in Section 3, i.e. the stability threshold of 0% is met for each of the points. However, some interesting observations can be made from these graphs. Segment **A** of Figure 7 is reported as one segment by the algorithm, whereas based on visual inspection one could argue that there are one or more change points in it. However, the likelihood calculations of the algorithm show that the variations being observed are not statistically significant and probably attributable to noise. A similar situation occurs in segment **B** of Figure 8, which contains a seemingly significant local minima. The converse appears in Figure 9, where **C** and **D** are reported as two separate segments, even though they visually appear to be a single segment. A reason is that we often tend to focus on straight-line segments in visual examinations [Att54]. Figure 6 represents a case where all the change points detected by the algorithm seem to agree with our intuitive notion of change-points.

4.3 Comparison With Visual Change Point Detection

A crucial issue in evaluating the behavior of a change

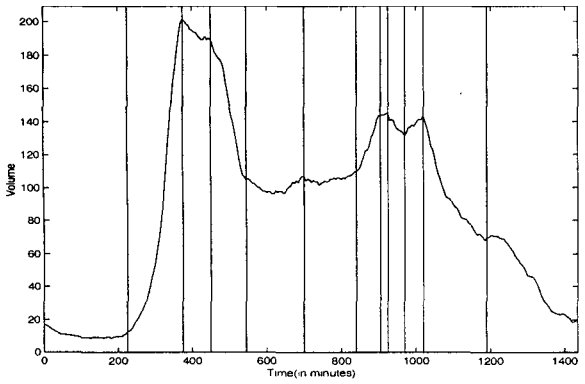

Figure 10: Subject S_1

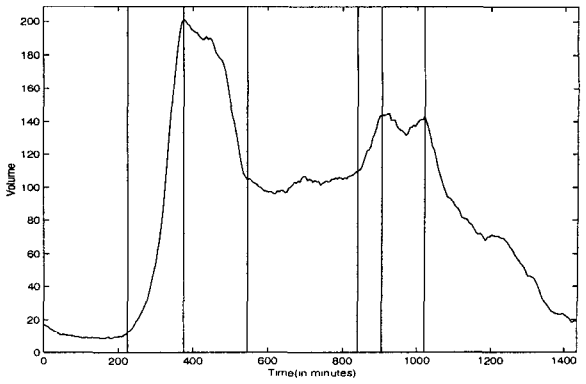

Figure 11: Subject S_2

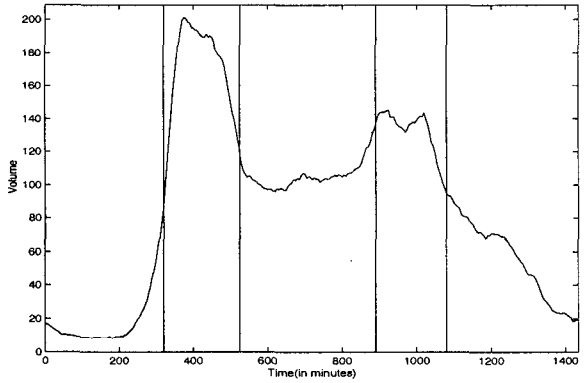

Figure 12: Subject S_3

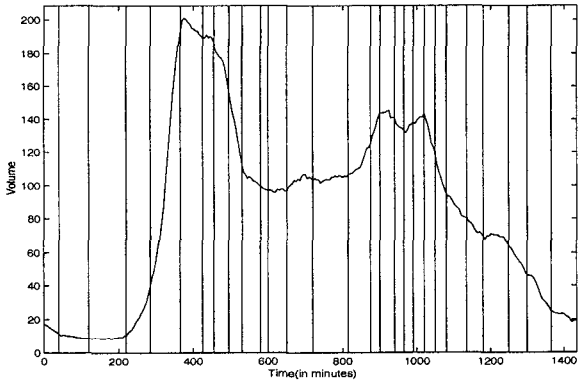

Figure 13: Subject S_4

Benchmark	Algorithm	Subject S_1	Subject S_2	Subject S_3	Subject S_4
V274	1.0	1.79	2.04	2.58	1.77

Table 2: Comparison of likelihood estimates for Algorithmic and Visual Approaches

point detection algorithm is to determine if the change points detected by it are indeed *true change points.* However, this raises the issue of first determining what the true change points of a function are. This is a difficult question to answer, because it in turn depends on the method employed to determine the true change points. Our approach was to examine the techniques used in the traffic domain, from which the data was taken. Traffic engineers use visual inspection for detecting change points in traffic data. Hence, we selected the data set of Figure 6 and asked four human subjects to detect change points[3] in it by visual inspection. Subject S_1 and S_2 were given smoothed representation of the time sequence, while subjects S_3 and S_4 received the original data set.

Figures 10 through 13 show the change points reported by subjects S_1, S_2, S_3, and S_4, respectively. The change points detected by subject S_1, Figure 10, seem to be the most similar to those detected by our algorithm. Subject S_2, Figure 11, seems to be using a quadratic model for segmentation, while subject S_3, Figure 12, seems to be using a cubic model. Subject S_4, Figure 13, seems to be using a linear segmentation model.

One thing that became clear from this experiment was that determining the *true change points* of a function is not at all straightforward, and human observers can have significant disagreements. Thus, a technique based on detecting change points based on some quantitative measure of likelihood is perhaps more robust than any of these.

To quantify the quality of change-points identified by the subjects, we calculated the likelihood estimates for each of the models and compared them with the likelihood criteria of the model identified by our algorithm. The resulting ratios are shown in Table 2. The results show that statistically speaking the

[3]Specific instruction given was to identify points at which the phenomenon changed significantly. Subjects were *not* given any instructions on *how* to do this, to eliminate bias.

```
1.   T = ∅
2.   while(true)
3.       T = T ∪ new_data_point
4.       split_likelihood_criteria = Find_Split_Likelihood_Criteria(T, MSet)
5.       no_split_likelihood_criteria = Find_Likelihood_Criteria(T, MSet)
6.       if ((no_split_likelihood_criteria − split_likelihood_criteria) > δ) then
7.           Report Change Of Pattern
8.           T = ∅.
9.       endif
10.  endwhile
```

Figure 14: Trend-Change Monitoring Algorithm

```
1.   optimal_likelihood_criteria = ∞
2.   for(i = p to |T| − p − 1) do begin
3.       likelihood_criteria = Find_Likelihood_Criteria(T[1, i], MSet) +
4.                             Find_Likelihood_Criteria(T[i + 1, |T|], MSet)
5.       if (likelihood_criteria < optimal_likelihood_criteria)
6.           optimal_likelihood_criteria = likelihood_criteria
7.       endif
8.   endfor
9.   return optimal_likelihood_criteria
```

Figure 15: Find_Split_Likelihood_Criteria Algorithm

algorithm performed better then any of the four subjects.

5 The Incremental Algorithm

The batch algorithm is useful only when data collection precedes analysis. In some cases, change-point detection must proceed concurrently with data collection, e.g. dynamic control of highway ramp metering lights. Towards this we developed an incremental algorithm. The key idea is that if the next data point collected by the sensor reflects a significant change in phenomenon, then its likelihood criteria of being a change-point is going be smaller then the likelihood criteria that it is not. However, if the difference in likelihoods is small, we cannot definitively conclude that a change did occur, since it may be the artifact of a large amount of noise in the data. Therefore, we use the criteria that a change-point has been detected if and only if

$$(\mathcal{L}_{no\_change} - \mathcal{L}_{change})/\mathcal{L}_{no\_change} > \delta,$$

where δ is a user-defined likelihood increase threshold.

Suppose that the last change-point was detected at time t_{k-1}. At time t_k the algorithm starts by collecting enough data to fit the regression model. Suppose at time t_j a new data point is collected. The candidate change point is found by determining t_i, with likelihood criterion $\mathcal{L}_{min}(k,j)$, such that

$$\mathcal{L}_{min}(k,j) = \min_{k<i\leq j}\{\mathcal{L}(k,i) + \mathcal{L}(i+1,j)\}.$$

If this minimum is significantly smaller than $\mathcal{L}(k,j)$, i.e. the likelihood criteria of no change-points from t_k to t_j, then t_i is a change-point. Otherwise, the process should continue with the next point, i.e. t_{j+1}. The algorithm is shown in Figures 14 and 15.

In the incremental algorithm, execution time is a significant consideration. If enough information is stored, some of the calculations can be avoided. Thus, at time t_{j+1} to find likelihood criteria

$$\mathcal{L}_{min}(k,j+1) = \min_{k<i\leq j}\{\mathcal{L}(k,i) + \mathcal{L}(i+1,j+1)\}$$

it is only necessary to calculate $\mathcal{L}(i+1,j+1)$, since $\mathcal{L}(k,i)$ was calculated in the previous iteration.

It should be noted that if a change-point is not detected for a long time, the successive computations become increasingly expensive. A possible solution is to consider a sliding window of only the last w points.

6 Performance Evaluation of the Incremental Algorithm

To study the performance of the incremental algorithm, we used data set generated by the following function

$$f(t) = \begin{cases} t * h/40 + \epsilon & : \quad t \in [1,39] \\ (80-t) * h/40 + \epsilon & : \quad t \in [40,80] \end{cases}$$

where the noise ϵ is Gaussian with zero mean and unit variance.

The goal of this experiment was to observe if the algorithm is able to accurately recognize the change-points. Accuracy is measured both by how close the identified change-point is to the point where the actual change occurred, and by how long it takes the algorithm to recognize the change.

	Incremental ($\delta = 35\%$)		Incremental ($\delta = 45\%$)		Batch ($s = 5\%$)
h	**change point detected**	**detection time**	**change point detected**	**detection time**	**change point detected**
70	38	42	38	43	40
60	39	42	39	43	39
50	40	43	37	44	40
40	38	42	38	43	40
30	37	44	37	44	41
20	36 48 58	44 53 63	32 50	46 53	41
15	5 44 56	9 49 59	41 54	50 57	40
10	9 40	12 74	9	12	40

Table 3: Performance of Incremental and Batch Algorithms; the actual change-point is 40.

The results of the experiment are shown in Table 3. The algorithm performs well for data sets with high signal-to-noise ratio. In addition, the time it takes to realize that the change occurred is small. However, for data sets with $h \leq 20$, the algorithm starts to break. The change-point estimates become increasingly inaccurate. Moreover, the latency of recognizing that change has occurred increases. In addition, for likelihood increase threshold $\delta = 35\%$, the algorithm identifies spurious change-points. Increasing the threshold to 45% does not eliminate spurious change-points, but eliminates a true change-point when $h = 10$.

The last column in Table 3 represents results obtained by running the batch algorithm on the same data sets with stability threshold $s = 5\%$. Note that the batch algorithm identifies change-points with very high accuracy, showing it to be much more tolerant of noise than the incremental algorithm. This is because the batch algorithm tries to achieve a global optimization of the likelihood metric, while the incremental algorithm seeks only local optimization due to unavailability of data about the future.

7 Conclusion and Future Work

In this paper, we presented an approach for event detection from time series data. The approach allows us to detect a change-point by detecting the change of model (or parameters of the model) that describe the underlying data. We use a combination of change-point detection and model selection techniques. The proposed approach does not assume the availability of a model describing the data, or the number of deviation points in the time series. In addition, the technique is independent of regression and model selection methods.

Our experimental results suggest that both algorithms are able to correctly identify change-points in cases where signal-to-noise ratio is not too low. In addition, the proposed approach is more robust than using visual inspection by humans, at least by the likelihood measure used here. First, it is not subject to human tendency to segment smooth curves into piecewise straight lines. Second, while human beings find it hard to work with data that contains a lot of noise, the algorithms are able to handle such data sets (as long as the noise level doesn't dominate the signal). The batch algorithm is more robust than the incremental one, since it works with the entire data set and can perform global optimization.

As discussed in [Raf93], applicable Bayesian approaches have been found to produce results more easily than non-Bayesian ones, especially for change point detection in one dimensional stochastic processes. However, a significant hurdle is the existence of a prior model that is both sophisticated enough to model the application, and computationally tractable for deriving the posterior model. In general, to make the computation tractable often simplifying assumptions are made [CGS92]. Previous work [CGS92] has shown that iterative techniques such as Monte-Carlo methods can be used to compute the marginal posterior densities. Our approach is non-Bayesian, and hence doesn't require a prior model. It would be an interesting future research to see how our approach compares with a Bayesian one for the problem of event detection.

8 Acknowledgments

The research reported herein has been supported in part by NSF grant no. EHR-9554517 and ARL contract no. DAKF11-98-P-0359.

References

[Att54] F. Attneave. Some informational aspects of visual perception. *Psychol. Rev.*, 61:183–193, 1954.

[CGS92] B.P. Carlin, A.E. Gelfand, and A.F. Smith. Hierarchical bayesian analysis of change-point problems. *Journal of Applied Statistics*, 41(2):389–405, 1992.

[CM98] Vladimir Cherkassky and Filip Mulier. *Learning from Data.* Wiley-Interscience, New York, N.Y., 1998.

[Gut74] S.B. Guthery. Partition regression. *J. Amer. Statist. Ass.*, 69:945–947, 1974.

[GWS98] Valery Guralnik, Duminda Wijesekera, and Jaideep Srivastava. Pattern directed mining of sequence data. In *The Fourth International Coference on Knowledge Discovery and Data Mining*, 1998.

[Haw76] Douglas M. Hawkins. Point estimation of parameters of piecewise regression models. *The Journal of the Royal Statistical Society Series C (Applied Statistics)*, 25(1):51–57, 1976.

[HKM+96] K. Hatonen, M. Klemettineen, H. Mannila, P. Ronkainen, and H. Toivon en. Knowledge discovery from telecommunication network alarm databases. In *Proc. of the 12th Int'l Conf. on Data Eng.*, pages 115–122, Kyoto, Japan, 1996.

[HM73] D.M. Hawkins and D.F. Merriam. Optimal zonation of digitized sequential data. *Mathematical Geology*, 5(4):389–395, 1973.

[Hud66] D.J. Hudson. Fitting segmented curves whose joint points have to be estimated. *J. Amer. Statist. Ass.*, 61:1097–1125, 1966.

[Hus93] Marie Huskova. Nonparametric procedures for detecting a change in simple linear regression models. In *Applied Change Point Problems in Statistics*, 1993.

[KC96] David Kincaid and Ward Cheney. *Numerical Analysis.* Brooks/Cole Publishing Company, Pacific Grove, CA, 1996.

[KS97] Eamonn Keogh and Padhraic Smyth. A probabilistic approach to fast pattern matching in time series databases. In *Third International Conference on Knowledge Discovery and Data Mining*, 1997.

[Kue94] Robert O. Kuehl. *Statistical Principles of Research Design and Analysis.* Wadsworth Publishing Company, Belmont, California, 1994.

[MT96] H. Mannila and H. Toivonen. Discovering generalized episodes using minimal occurrences. In *Proc. of 2nd Int'l Conference on Knowledge Discovery and Data Mining*, pages 146–151, Portland, Oregon, 1996.

[MTV95] H. Mannila, H. Toivonen, and A. I. Verkamo. Discovering frequent episodes in sequences. In *Proc. of the First Int'l Conference on Knowledge Discovery and Data Mining*, pages 210–215, Montreal, Quebec, 1995.

[MTV97] H. Mannila, H. Toivonen, and A.I. Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discover*, 1(3):259–289, November 1997.

[PT96] B. Padmanabham and A. Tuzhilin. Pattern discovery in temporal databases: A temporal logic approach. In *Proc. of 2nd Int'l Conference on Knowledge Discovery and Data Mining*, pages 351–354, 1996.

[Raf93] Adrian E. Raftery. Change point and change curve modeling in stochastic processes and spatial statistics. Technical Report 23, University of Washington, 1993.

[SA95] R. Srikant and R. Agrawal. Mining generalized association rules. In *Proc. of the 21th VLDB Conference*, pages 407–419, Zurich, Switzerland, 1995.

[SO94] N. Sugiura and Todd Ogden. Testing change–points with linear trend. *Communications in Statistics B:Simulation and Computation*, 23:287–322, 1994.