Memory-Based Algorithms for Abrupt Change Detection in Sensor Data Streams

Daniel Nikovski, Ankur Jain

TR2007-024 June 2007

Abstract

This paper describes two novel learning algorithms for abrupt change detection in multivariate sensor data streams that can be applied when no explicit models of data distributions before and after the change are available. One of the algorithms, MB-GT, uses average Euclidean distances between pairs of data sets as the decision variable, and the other, MB-CUSUM, is a direct extension of the CUSUM algorithm to the case when the unknown probability density functions are estimated by means of kernel density estimates. The algorithms operate on a sliding memory buffer of the most recent N data readings, and consider all possible splits of that buffer into two contiguous windows before and after the change. Despite the apparent computational complexity of O(N^4) of this computation, our proposed algorithmic solutions exploit the structure present in their respective decision functions and exhibit computational complexity of only O(N^2) and memory requirement of O(N).

Memory-Based Algorithms for Abrupt Change Detection in Sensor Data Streams

Daniel Nikovski, *Member, IEEE* and Ankur Jain, *Member, IEEE*

***Abstract*— This paper describes two novel learning algorithms for abrupt change detection in multivariate sensor data streams that can be applied when no explicit models of data distributions before and after the change are available. One of the algorithms, $\mathsf{MB-GT}$, uses average Euclidean distances between pairs of data sets as the decision variable, and the other, $\mathsf{MB-CUSUM}$, is a direct extension of the CUSUM algorithm to the case when the unknown probability density functions are estimated by means of kernel density estimates. The algorithms operate on a sliding memory buffer of the most recent N data readings, and consider all possible splits of that buffer into two contiguous windows before and after the change. Despite the apparent computational complexity of $O(N^4)$ of this computation, our proposed algorithmic solutions exploit the structure present in their respective decision functions and exhibit computational complexity of only $O(N^2)$ and memory requirement of $O(N)$.**

I. Introduction

In the past, the detailed monitoring of industrial equipment in real-time has been economically feasible typically for large, expensive, and safety-critical installations such as nuclear power plants, spaceships, etc. However, the rapid progress of computer technology, and more specifically the advent of ubiquitous sensor networks, cheap wireless communications, and powerful embedded processors, has made it possible to implement equipment condition monitoring (ECM) technology for much cheaper devices, such as electrical motors, turbines, power switchgear, as well as for an ever expanding range of industrial processes, such as oil refining, food processing, etc.

The resulting explosive increase in the amount of sensor information that is constantly streaming from industrial sensor network installations would quickly overwhelm any human supervisor tasked with processing it. The only viable solution to the problem of processing this information quickly and accurately is to develop automated methods for monitoring of data streams. Whereas it is true that such automated methods are not likely to reach the competence and versatility of a well-trained human supervisor, they can still be very effective and accurate when designed to look for specific events in the sensor data stream.

One of the most important among these events is a sudden change in the data stream. Detecting such abrupt changes is not a trivial problem, because all but the simplest data streams vary even when no change in the process that generates the data has occurred. This might be caused either by the natural variability of the process, e.g. when the data come from a dynamical system, or to noise that is due to measurement errors, hidden variables, etc. In such cases, the detection of abrupt changes is done in the statistical sense, i.e., the problem reduces to detecting a difference between the probability distributions from which the data are sampled, before and after the change. In manufacturing, this task is often called Statistical Process Control (SPC), where the objective is to detect a departure from the in-control distribution of the data towards some other, out-of-control distribution.

D.Nikovski and A.Jain are with Mitsubishi Electric Research Laboratories, Cambridge, USA.

When the in-control and out-of-control distributions have known parametric forms and the respective parameters are known, it has been shown that the CUSUM algorithm originally due to Page [1] is optimal [2]. However, explicit modeling of the in-control and all possible out-of-control distributions is typically a laborious and expensive process, and might even be intractable. In contrast, what is needed are methods that can detect abrupt changes only by inspecting the data streams and reasoning about the probability distributions implied by the data readings themselves. One possible solution that we are proposing in this paper, employs memory-based machine learning methods that learn those distributions from data.

Section II reviews several existing methods for abrupt change detection (ACD) including the CUSUM algorithm and its variants, describes several recent algorithms based on machine learning, and discusses their shortcomings. Section III describes the two new memory-based ACD methods that we propose, and discusses computational schemes to reduce their computational complexity from $O(N^4)$ if implemented directly, to only $O(N^2)$. Section IV provides an experimental comparison of the two new algorithms with several alternative methods. Finally, section V discusses several ideas for even further reduction of the computational complexity of the methods, and possible increase in their accuracy.

II. Methods for Abrupt Change Detection

If we denote with $\mathbf{x}_t$ the d-dimensional data vector from the sensor data stream at time instant t, the problem of abrupt change detection is to determine whether such a change has occurred *at or before* the current time t. An important assumption for this problem is that the change is assumed to be permanent, i.e. once it has occurred, *all* subsequent readings come from the new distribution. (This is the typical situation for industrial equipment when the change is destructive, e.g. burnout, motor failure, etc.) All samples before the change are assumed to be independent and identically distributed (i.i.d.) random variables sampled from the distribution $p_0(\mathbf{x})$; similarly, all samples after the change are assumed to be i.i.d. variables sampled from the distribution $p_1(\mathbf{x})$.

For cases when $p_0(\mathbf{x})$ and $p_1(\mathbf{x})$ are known, Page proposed the CUSUM (CUmulative SUM) method that accumulates the log-likelihood of the current reading with respect to the two distributions, and makes a decision based on an auxiliary variable $g_t = S_t - m_t$ for $m_t = \min_{1 \leq j \leq t} S_j$, $S_t = \sum_{i=1}^{t} s_i$, and $s_i = \log \frac{p_1(\mathbf{x}_t)}{p_0(\mathbf{x}_t)}$. A change is declared to have occurred if $g_t > h$ for a suitably chosen threshold h, and this decision can be shown to be optimal with respect to maximizing detection probability for a given false-positive rate [2].

However, the CUSUM algorithm has the significant disadvantage that both distributions $p_0(\mathbf{x})$ and $p_1(\mathbf{x})$ must be known. Specifying an accurate probability distribution $p_0(\mathbf{x})$ for the normal operation of industrial equipment is typically hard and laborious even for the very engineers who designed the equipment; specifying a possible distribution $p_1(\mathbf{x})$ for the out-of-control case might be outright impossible. What is more, even the correct parametric forms for these distributions might not be available.

These limitations of CUSUM have spurred extensive research on alternative methods for ACD that are more data driven and do not rely on pre-specified distributions. An important direction of research is to use non-parametric statistics such as rank statistics, Kolmogorov-Smirnov, etc. [3]. Another line, to which this paper also belongs, has focused on ACD methods based on machine learning. The fundamental idea is to learn (fit) two probability distributions from data samples before and after the hypothesized change point, and then test for differences between the two distributions, often using information-theoretic distance measures such as Kullback-Leibler divergence, Rényi divergence, etc. [4].

The first main problem when using such methods is how to learn the two distributions from data. When the two distributions are known to be Gaussian, the sample means and variances for the two windows can be computed and the two distributions compared using Student's t-statistic. The much more important case is when the two probability density functions (pdfs) of the distributions are not Gaussian — for example, when they are multi-modal due to the system jumping between several distinct modes. Some popular methods, such as Gaussian Mixture Models, that are otherwise an excellent choice for modeling multi-modal distributions, are fairly poor solutions for this particular problem, because they are parametric and their fitting requires multiple iterative adjustments of the respective parameters [5]. This is prohibitively expensive when considering many possible change points. A much better alternative is to use memory-based methods, such as Parzen's kernel density estimate, a.k.a. Nadaraya-Watson estimate of a probability density function [5]. In this method, the probability density $p(\mathbf{x})$ is represented as a normalized sum of kernel values:

$$p(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} w(\mathbf{x} - \mathbf{x}_i) \tag{1}$$

where w is a suitably chosen kernel, and $\mathbf{x}_i$, $i = 1, n$ is a sample of data points from the distribution to be modeled. Popular choices for the kernel are Gaussian, tricubic, etc. [5].

The second main problem is how to compare the two distributions, once they have been fit from data. Some popular methods employ information-theoretic distance measures, such as Kullback-Leibler (KL) divergence:

$$D_{KL}(p_0 \parallel p_1) = \int \int \dots \int_{\mathbf{x} \in \Omega} p_0(\mathbf{x}) \log \frac{p_0(\mathbf{x})}{p_1(\mathbf{x})} d\mathbf{x}$$

The main difficulty when using KL divergence is the need to integrate over the entire domain Ω of $\mathbf{x}$ — this can be prohibitively expensive even in one-dimensional domains, and might be impossible in multivariate cases. Using other popular information-theoretic distance measures, such as Rényi divergence, Jensen-Shannon distance, Bregman divergence, Hellinger-Matsushita-Bhattacharya distance, etc., leads to similar integration-related difficulties.

As a result, much research has focused on approximate computation of these distances. For example, Guha et al. [4] describe several polynomial-time approximation schemes (PTAS) that can compute approximations to most of the above distances in polynomial time. While valuable from a theoretical point of view, similar PTAS methods are not very likely to result in practical algorithms that can be used for monitoring of actual industrial equipment. In contrast, the algorithms proposed in the next section have very favorable low-order polynomial complexity without having to resort to approximations.

The third main problem in algorithms that are based on machine learning is determining the size of the two windows from which the two pdfs are estimated. On one hand, the larger the window size, the better the asymptotic fit to the true pdf from which the data were sampled. On the other hand, when an actual change in distribution occurs, if the size of the window is large, the new data starts to affect the post-change distribution very slowly, resulting in longer detection times.

One possible way to deal with this contradiction is to consider windows of varying sizes, both before and after the change point, such that the sum of the two window sizes does not exceed the length N of a sufficiently large buffer of stream data kept in memory. This would allow both fast reaction to more drastic changes, for post-change windows of length as small as one data reading, as well as sensitivity to more subtle changes, if they can only be detected from distributions learned from larger windows. While a direct implementation of this idea typically results in increase of computational complexity to $O(N^4)$, we will demonstrate efficient implementations of the algorithms proposed below that eliminate this increase in complexity.

III. Memory-Based Abrupt Change Detection

In this section, we propose two algorithms for abrupt change detection: one is based on Euclidean distances between two sets of samples, while the other one is an extension of the CUSUM algorithm to the case when both the pre-change and post-change distributions are modeled via Parzen kernel density estimates. Both algorithms work on variable sizes of the two windows, considering all possible partitions of a memory buffer of size N of past data readings. While vastly different in their motivation, derivation, and probabilistic assumptions, the two algorithms do share a common computational structure that can be exploited to significantly reduce the computational

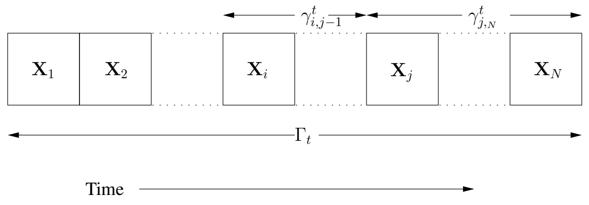

Fig. 1. A snapshot of a sliding window and its probable sub-windows.

cost of considering all possible partitions of the memory buffer.

Both algorithms work on a sliding memory buffer Γ^t which contains the last N readings, always renumbered for convenience from $\mathbf{x}_1$ to $\mathbf{x}_N$, such that $\mathbf{x}_N$ is the latest recorded reading. A change-detection algorithm α has to compute a quantitative figure of merit Υ^α_t, which is in proportion to the possibility that a change has occurred within the span of the current buffer in memory. In particular, Υ^α_t can be, but is not limited to, a distance measure between two *sub-windows* of the current memory buffer.

In contrast to existing algorithms that simply split the buffer into two equal parts and test for difference between them, the two algorithms described below consider all possible pairs of indices (i, j), such that $1 \le i < j \le N$, that split Γ^t into two adjacent *sub-windows*, $\gamma^t_{i,j-1}$ and $\gamma^t_{j,N}$ (as shown in Figure 1). Here $\gamma^t_{p,q} = \{\mathbf{x}_p, \ldots, \mathbf{x}_q\}$.

The index i defines the beginning of the first window, and the index j defines the beginning of the second window (and, respectively, the hypothesized change point). The end of the first window has index $j - 1$, ensuring contiguity of the two windows, whereas the end of the second window is always the latest reading, $\mathbf{x}_N$. Under this scheme, the two windows might have unequal lengths, and might even consist of a single data sample. Under this setting, if a separate figure of merit $\Upsilon^\alpha_t(i, j)$ is computed for each possible pair subject to the listed constraints, the overall figure of merit is the maximum over all splits: $\Upsilon^\alpha_t = \max_{1\le i<j\le N} \Upsilon^\alpha_t(i, j)$. The modeling challenge, then, is which figure of merit Υ^α_t to use, while the computational challenge is how to compute it efficiently for each incoming data sample $\mathbf{x}_t$.

A. A Memory-Based Graph Theoretic Algorithm

One very straightforward solution to the problem of computing the distance between two distributions indirectly specified by means of two sample sets is to compute the average distance between the samples themselves. Since each sample is a point in a multi-dimensional Euclidean space, a natural distance measure between pairs of points $\mathbf{x}_k$ and $\mathbf{x}_l$ is their Euclidean distance $d_{k,l} \doteq \|\mathbf{x}_k - \mathbf{x}_l\|$. For a particular split defined by the index pair (i, j) specified as above, we can compute the average distance between the two as

$$C_{i,j} = \frac{\sum_{k=i}^{j-1} \sum_{l=j}^{N} d_{k,l}}{(j-i)(N-j+1)}$$

We will call the corresponding change detection algorithm $\mathsf{MB-GT}$ (Memory-Based Graph Theoretic). Its overall figure of merit $\Upsilon^{\mathsf{MB-GT}}_t$ can be computed as described above: $\Upsilon^{\mathsf{MB-GT}}_t = \max_{1\le i<j\le N} C_{i,j}$. Since computing each $C_{i,j}$ is of complexity of $O(N^2)$, and there are $O(N^2)$ such terms to be considered, the overall complexity of computing $\Upsilon^{\mathsf{MB-GT}}_t$ seems to be $O(N^4)$, if implemented directly. This complexity is unacceptable for practical applications.

However, the computation of individual $C_{i,j}$ terms has certain redundancy and repetitive structure that can be exploited to bring the computational complexity of $\mathsf{MB-GT}$ back down to $O(N^2)$. If we define

$$C'_{i,j} \doteq \sum_{k=i}^{j-1} \sum_{l=j}^{N} d_{k,l} \qquad \beta_{i,j} \doteq \sum_{l=j}^{N} d_{i,l},$$

one can verify that the following recurrent relationships hold: $\beta_{i,j-1} = \beta_{i,j} + d_{i,j-1}$, with $\beta_{i,N+1} = 0$, and $C'_{i-1,j} = C'_{i,j} + \beta_{i-1,j}$, with $C'_{j,j} = 0$ for all $1 \le j \le N$. These recurrences suggest the following efficient computational algorithm. If the values $C'_{i,j}$ are placed in a tableau that is conceptually similar to a matrix, this matrix would be upper triangular due to the constraint $i < j$. Computation can start with the bottom row of this matrix that has a single element $C'_{N,N}$ which is zero by definition. For each row $1 \le i < N$ above the last one, proceeding from bottom to top, two steps are performed:

1) All values $\beta_{i,j}$ are computed recurrently from their immediate neighbor to the right, proceeding right to left, and using the recurrence $\beta_{i,j-1} = \beta_{i,j} + d_{i,j-1}$.
2) All values $C'_{i,j}$ are computed from the respective values $\beta_{i,j}$ and the values $C'_{i+1,j}$ in the row immediately below the current one, using the recurrence $C'_{i,j} = C'_{i+1,j} + \beta_{i,j}$.

Computing $\Upsilon^{\mathsf{MB-GT}}_t$ can be done simultaneously with the computation of the individual terms $C'_{i,j}$, since it involves only normalization and maximization. For this reason, it is *not* necessary to keep all values $\beta_{i,j}$ and $C'_{i,j}$ in memory; it suffices to keep a buffer of size N elements for the current row i of values $\beta_{i,j}$, and two buffers of the same size for $C'_{i,j}$ and $C'_{i+1,j}$. Thus, the memory requirement of this algorithm is only $O(N)$. As for its computational complexity, the computation of $\beta_{i,j}$ and $C'_{i,j}$ for each row i is only of $O(N)$, and since there are N such rows, the overall complexity is only $O(N^2)$, as opposed to $O(N^4)$ for a naïve implementation.

B. A Memory-Based CUSUM Algorithm

Unlike $\mathsf{MB-GT}$, the second algorithm we propose, $\mathsf{MB-CUSUM}$, has probabilistic foundations identical to that of the original CUSUM algorithm, which potentially allows it to achieve optimal change detection under certain modeling conditions. At the same time, despite its very different theoretical foundation, $\mathsf{MB-CUSUM}$ has similar computational structure to $\mathsf{MB-GT}$, and we will demonstrate how this structure can be leveraged to achieve the same significant improvements in computational complexity.

Following the derivation of CUSUM in [2], we consider the following hypotheses about a possible change within the current buffer of N readings kept in memory:

$$\mathbf{H}_{i,0}\colon \mathbf{x}_k \sim p_0 \quad \text{for } i \le k \le N \tag{2}$$

$$1 \le i < j \le N, \mathbf{H}_{ij} : \mathbf{x}_k \sim p_0 \quad \text{for } i \le k \le j-1 \tag{3}$$

$$\mathbf{x}_l \sim p_1 \quad \text{for } j \le l \le N \tag{4}$$

Here we are considering the null hypotheses $\mathbf{H}_{i,0}$ that no change has occurred while the latest $N-i+1$ sample were collected, vs. multiple hypotheses $\mathbf{H}_{i,j}$ that such a change has occurred. Compared to standard CUSUM, by introducing the starting index i, we are expanding the set of hypotheses to be tested to those that do not necessarily use all N samples in the window. According to the Neyman-Pearson lemma, the most powerful test that we can perform when testing each particular hypothesis $\mathbf{H}_{i,j}$ vs. $\mathbf{H}_{i,0}$ (i.e., the test that has the highest probability of rejecting a false null hypothesis), is the likelihood ratio

$$\Lambda_{ij} = \frac{\prod_{k=i}^{j-1} p_0(\mathbf{x}_k) \cdot \prod_{l=j}^{N} p_1(\mathbf{x}_l)}{\prod_{k=i}^{N} p_0(\mathbf{x}_k)}$$

For convenience, the log-likelihood ratio $S_{ij} = \log(\Lambda_{ij})$ is commonly used. In our algorithm, we replace the true pdfs p_0 and p_1 with their kernel density estimates, as described by Equation 1, to result in

$$\mathcal{S}_{i,j} = \sum_{l=j}^{N} \log \frac{\frac{1}{N-j+1} \sum_{k=j}^{N} w_{l,k}}{\frac{1}{j-i} \sum_{k=i}^{j-1} w_{l,k}}, \quad w_{l,k} \doteq w(\mathbf{x}_l - \mathbf{x}_k) \tag{5}$$

Here $w_{l,k}$ is a kernel weight for the pair of samples $(\mathbf{x}_l, \mathbf{x}_k)$, and it also holds that $w_{l,k} = w(d_{l,k})$. By using the maximum likelihood principle, the figure of merit for this algorithm will be $\Upsilon_t^{\mathsf{MB-CUSUM}} = \max_{1 \le i < j \le N} S_{i,j}$. Again, a direct computation of $\Upsilon_t^{\mathsf{MB-CUSUM}}$ would have computational complexity of $O(N^4)$.

However, $\Upsilon_t^{\mathsf{MB-CUSUM}}$ has a similar structure to $\Upsilon_t^{\mathsf{MB-GT}}$ that can again be exploited to reduce its computational complexity. Again, we can conceptually organize the values of $S_{i,j}$ in a tableau, and define the following auxiliary variables:

$$\mu_j^l \doteq \sum_{k=j}^{N} w_{l,k} \qquad \nu_{i,j}^l \doteq \sum_{k=i}^{j-1} w_{l,k}. \tag{6}$$

Although there appear to be $O(N^3)$ $\nu_{i,j}^l$ terms to be computed, the recurrent re-formulation of Equation 5

$$S_{i,j} = S_{i,j+1} + \log \mu_j^j - \log \nu_{i,j}^j + \log(j-i) - \log(N-j+1) \tag{7}$$

can convince us that not all of them are needed. By further defining $\mu_j' \doteq \mu_j^j$, and $\nu_{i,j}' \doteq \nu_{i,j}^j$, we can use the following equations as a basis for an efficient algorithm:

$$\mu_j' = \sum_{k=j}^{N} w_{j,k} \qquad \nu_{i,j}' = \nu_{i+1,j}' + w_{j,i}. \tag{8}$$

Note that only the one for $\nu_{i,j}'$ happens to be recurrent; the other, for μ_j', is computed directly. These equations suggest the following efficient algorithm:

```
S1:   Compute $\mu_j'$ for $j = 1, N$ directly, per Equation 8.
      This computation takes $O(N^2)$, but the results can
      be stored in $O(N)$ space.
S2:   For each row $i = N, 1$ of the matrix $S_{i,j}$, starting
      from the bottom row $i = N$ and moving upwards to
      the first row $i = 1$, perform the following two steps:
    S2.1: For each value of $j$ between $i+1$ and $N$,
          compute $\nu_{i,j}'$ from the corresponding $\nu_{i+1,j}'$
          in the row below, and $w_{j,i}$, per Equation 8.
    S2.2: For each value of $j$ between $N$ and $i+1$,
          compute $S_{i,j}$ from the value $S_{i,j+1}$ immediately
          to the right, using the equation
          $S_{i,j} = S_{i,j+1} + \log \mu_j' - \log \nu_{i,j}' + \log(j-i) - \log(N-j+1)$,
          starting with $S_{i,N+1} = 0$ for all $i = 1, N$.
          The computation in this step proceeds strictly
          right to left ($j = N, i+1$).
```

IV. Experimental verification of MB − GT and MB − CUSUM

We now present an experimental comparison of the two algorithms proposed above with several existing methods. In all experiments, we generate a sequence of experimental data of length W, such that the first $W/2$ data points come from a distribution p_0, and the second $W/2$ come from a distribution p_1. All change detection algorithms process the data points on-line, one by one as they arrive, and their objective is to decide whether a change has occurred at or before the current point in time. If a change has occurred, they output a decision variable $\hat{y}_t = 1$; if not, $\hat{y}_t = 0$. All algorithms assume that *only one* change will occur in the data stream, and this change is persistent. Under this assumption, which we call the single-change hypothesis, a positive decision persists for all future time moments, and the final sequence of decisions is a step function $\hat{y}_t = 0$ for $1 \le t < t_{det}$, $\hat{y}_t = 1$ for $t_{det} \le t \le W$, where t_{det} is the first time when a change was detected. The ground truth y_t for the decision variable is also a step function $y_t = 0$ for $1 \le t \le W/2$, $y_t = 1$ for $W/2 < t \le W$, where the step occurs precisely at the time of change $t = W/2 + 1$. Detection accuracy is determined by how close the produced decisions and the ground truth coincide.

Following the current standards for experimental evaluation of classifiers in the machine learning community, we report the entire receiver operating characteristic (ROC) curves for our algorithms [6]. Since the horizontal axis of an ROC curve corresponds to the false positive rate, and the vertical axis corresponds to the true positive rate of detection, the ROC curves measure directly the statistical power of the respective algorithms for all possible decision thresholds. In order to eliminate statistical variability and obtain asymptotically accurate ROC curves, we average them over 100 identical runs for each experiment reported below.

The two new algorithms MB − GT and MB − CUSUM were compared against several known change detection algorithms. Since the actual distributions before and after the change are known, we can employ the classical CUSUM algorithm as the gold standard for optimal detection. An algorithm that uses the t-statistic between the two windows as its figure of merit is denoted by $\mathsf{MB-TSTAT}$. Similarly, an algorithm that uses the Kolmogorov-Smirnov statistic between the two windows as its figure of merit is denoted by $\mathsf{MB-KS}$. The $\mathsf{MB-KS}$ algorithm is also very expensive ($O(N^3 \log N)$), due to the need to sort the data in the buffer. Apart from comparing the relative performance of these algorithms with respect to CUSUM, we also investigated several additional questions:

- How does the magnitude of the difference between the distributions p_0 and p_1 affect performance?
- How does the shape of the distribution affect the relative performance of the algorithms, especially of those that assume a specific parametric form?
- Does using all possible splits (i, j) of the memory buffer result in better performance with respect to equal sizes $(j - i = N - j + 1)$? (The former case is denoted with a suffix -UNEQ, and the latter case with suffix -EQ in the graphs below.)
- Is there a benefit to limiting the size of the smallest window that the memory-based algorithms would consider? Intuitively, considering kernel density estimates from very few samples (e.g. one or two) would result in imprecise density models.

A. *Gaussian distributions*

The first set of experiments used Gaussian distributions before and after the change. The size of the memory buffer was $N = 100$. In all cases, the mean before the change was 0, while the mean after the change took on values in $\{0.5, 1.0, 1.5, 2.5\}$. The standard deviation before and after the change was equal, and in experiments took on values in $\{0.2, 0.4, 0.8, 1.0\}$. As expected, for large ratios between the mean change and the standard deviation of the distributions, all algorithms perform well; practically all of them achieve 100% detection rate for zero false positive rate. Conversely, the hardest case, with the worst performance for the algorithms, is when this ratio is minimal — in this experiment, this happens when the jump in means is only 0.5 and the standard deviation is 1.0. This case is shown in Figure 2. It can be seen that even the omniscient CUSUM algorithm that knows the exact distributions that generated the data cannot achieve 100% accuracy in all cases. From the learning algorithms, for low false positive rates, the $\mathsf{MB-KS}$ algorithm performs best; for higher false positive rates, $\mathsf{MB-CUSUM}$ is the best, and it even approaches the accuracy of omniscient CUSUM. $\mathsf{MB-CUSUM}$ also clearly dominates $\mathsf{MB-GT}$ and $\mathsf{MB-TSTAT}$ in all cases. Comparing the versions that use equal and unequal windows sizes, we can see that the latter is always better; this difference is most pronounced for the $\mathsf{MB-CUSUM}$ algorithm. Inspection of the remaining 15 experimental graphs (not shown here for lack of space) confirms these conclusions.

B. *Effect of the size of memory buffer and data dimensionality*

In the next set of experiments, we varied the size of the memory buffer $N \in \{50, 100, 200, 250\}$. We found only negligible change in performance when the size of the buffer

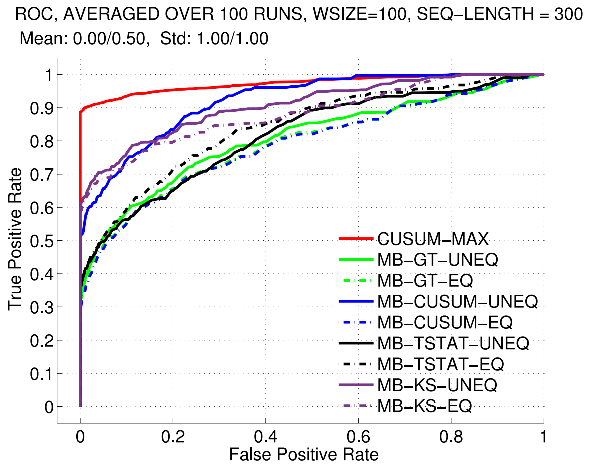

Fig. 2. The most difficult test case for Gaussian distributions: the change in means is equal to half of the standard deviation.

was reduced; this is probably due to the fact that all of these values are sufficient to estimate well a single one dimensional Gaussian pdf. (The only more noticeable degradation occurred for $N = 50$, and only for $\mathsf{MB-CUSUM}$.)

Another set of experiments tested whether the algorithms can handle multivariate readings well. $\mathsf{MB-GT}$ and $\mathsf{MB-CUSUM}$ do not need any modifications. $\mathsf{MB-GT}$ is based on Euclidean distances, which are defined identically for any number of dimensions, whereas $\mathsf{MB-CUSUM}$ simply has to use multi-dimensional kernels — for the case of Gaussian kernels, the extension to the multivariate case is straightforward. In contrast, extending the t-test to multivariate data is not trivial. One possible generalization is Hotelling's T^2 statistic; another one is to compute individual t-values along each dimension, and take their norm as a figure of merit. The latter approach was used in our experiments (using L_2 norm), since it avoids the computation of a pooled covariance matrix for each pair of sub-windows. Finally, extending the $\mathsf{MB-KS}$ algorithm to arbitrary multivariate data is not possible, because the Kolmogorov-Smirnov statistic is defined only for univariate data, and only partial extensions to two-dimensional data exist.

As a result, the four algorithms CUSUM, $\mathsf{MB-GT}$, $\mathsf{MB-CUSUM}$, and $\mathsf{MB-TSTAT}$ were compared in a series of experiments where the dimensionality of the data was chosen from the set $\{2, 4, 8, 16\}$, and the ratio of mean change to standard variance was either 0.5 or 1.0. Although the specific performance in each case was different, the same overall behavior and relative ranking of the algorithms was observed: $\mathsf{MB-TSTAT}$ had the worse overall performance; $\mathsf{MB-GT}$ had a slightly better performance than $\mathsf{MB-TSTAT}$ in seven out of eight cases; $\mathsf{MB-CUSUM}$ had a significantly better performance than both $\mathsf{MB-TSTAT}$ and $\mathsf{MB-GT}$ in all cases, but still not as good as the omniscient CUSUM. A representative hard case of sixteen-dimensional data and a change in means equal to half a standard deviation is shown in Figure 3: the ROC curve for $\mathsf{MB-CUSUM}$ lies approximately halfway between that of omniscient CUSUM, on one hand, and $\mathsf{MB-TSTAT}$ and $\mathsf{MB-GT}$, on the other.

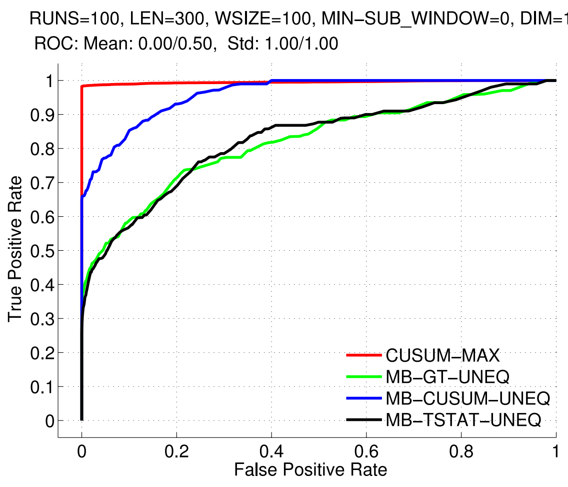

Fig. 3. For 16-dimensional data, $\mathsf{MB-CUSUM}$ substantially outperforms $\mathsf{MB-TSTAT}$ and $\mathsf{MB-GT}$.

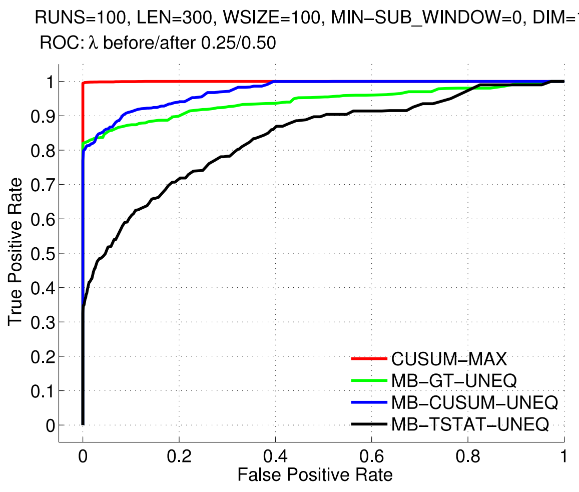

Fig. 4. For exponential distributions, the relative performance of the algorithms remains the same.

C. Exponential distributions

In a final set of experiments, the effect of the parametric shape of the distribution was verified. The distribution before the change was exponential with mean λ_0, varied so that $\lambda_0 \in \{0.25, 0.75\}$, while the distribution after the change was also exponential, but with mean λ_1, varied so that $\lambda_1 \in \{0.25, 0.5, 0.75, 1.0, 1.25\}$. (Cases when $\lambda_0 = \lambda_1$ were not tested.) One hard representative case, when the two means are fairly close ($\lambda_0 = 0.25$, $\lambda_1 = 0.5$), is shown in Figure 4. It can be seen that detecting the change is trivial for the omniscient CUSUM algorithm, which is supplied with the exact distributions p_0 and p_1. The two algorithms $\mathsf{MB-CUSUM}$ and $\mathsf{MB-GT}$ perform relatively well, while the Gaussian assumption built into the $\mathsf{MB-TSTAT}$ algorithm results in much worse performance.

V. Conclusion

Two novel fast memory-based algorithms for abrupt change detection were proposed, and their performance versus known methods for change detection was tested. Experimental verification under various conditions such as type of distribution, magnitude of change, data dimensionality, and memory buffer size confirmed their good performance. Between the two, the memory-based extension to CUSUM, $\mathsf{MB-CUSUM}$, outperformed $\mathsf{MB-GT}$ in practically all cases, which can be attributed to its probabilistic foundation and the optimality guarantees for its non-learning predecessor, the original CUSUM algorithm. The only algorithm that outperformed $\mathsf{MB-CUSUM}$ was the one using the Kolmogorov-Smirnov statistic; however, this algorithm is very expensive computationally ($O(N^3 \log N)$), and inherently limited to one- or at most two-dimensional data. In contrast, $\mathsf{MB-CUSUM}$ is fast ($O(N^2)$), and easily handles multivariate data; its accuracy even increases on higher-dimensional data. In addition, $\mathsf{MB-CUSUM}$ is not too sensitive to the size of memory-buffer. In addition, $\mathsf{MB-CUSUM}$ and $\mathsf{MB-GT}$ are non-parametric memory-based algorithms that can work on all kinds of distributions, whereas using t-tests is limited to those that do not depart too much from Gaussian.

The proposed algorithmic solutions reduce the complexity of computing their respective figures of merit from $O(N^4)$ to $O(N^2)$; the usefulness of these solutions is much reinforced by the experimental verification that considering all possible splits of the memory buffer does make a big difference when compared to the case when only equal splits are used. This is a substantial improvement over the current practice in the field, where typically only one single split into two windows of size $N/2$ is considered.

As to whether further improvements in computational complexity are possible, one possibility is to amortize computation across time periods. Currently, computation for each time period t starts from scratch, and one might imagine an alternative scheme where the statistics $C_{i,j}$, respectively $S_{i,j}$, are computed from their counterpart values in previous time slices. However, since these statistics are placed in a tableau of size $O(N^2)$, retaining these tableaux in their entirety would destroy the $O(N)$ memory property of the algorithms.

Another direction to consider is the use of more advanced memory-based learning algorithms, for example ones that rely on data structures such as kd-trees, adaptive rectangular trees, ball-trees, etc. [7]. These methods have been used very effectively in memory-based learning, especially for lower-dimensional data, and are suitable candidates for future use in abrupt change detection algorithms.

References

[1] E.S. Page, Continuous inspection schemes, in *Biometrika*, vol.41, pp.100–115, 1954.
[2] M. Basseville and I. V. Nikiforov, *Detection of Abrupt Changes: Theory and Application*, Englewood Cliffs, NJ: Prentice Hall, 1993.
[3] B. E. Brodsky and B. S. Darkhovsky, *Nonparametric Methods in Change-Point Problems*, Kluwer, 1991.
[4] S. Guha, A. McGregor, and S. Venkatasubramanian, Streaming and sublinear approximation of entropy and information distances, in *Proceedings of SODA'06*, pp. 733–742, ACM Press, 2006.
[5] T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning*, Springer, 2001.
[6] F. J. Provost and T. Fawcett, Analysis and visualization of classifier performance: comparison under imprecise class and cost distributions, in *Proceedings of KDD'97*, pp.43–48, AAAI Press, 1997.
[7] C. G. Atkeson, A. W. Moore, and S. Schaal, Locally weighted learning, in *Artificial Intelligence Review*, vol.11, no.1-5, pp.11–73, 1997.