HIGH-DIMENSION ONLINE CHANGEPOINT DETECTION

Candidate number: 8254R

Department of Applied Mathematic and Theoretical Physics, University of Cambridge

Supervisor: Prof. Richard Sainworth

Department od Pure Mathematics and Mathematical Statistics, University od Cambridge

Problem description

We assume that a sequence of observation $x_1, x_2, \dots x_T$ may be divided into non-overlapping product partitions [2]. The delineations between partitions we call the "changepoints". Our model also assume that each such partition: p, the data within it are i.i.d. from some probability distribution: $p(x_t \mid n_p)$. The parameter: $n, p = 1, 2$ we taken to be i.i.d. as well. We denote the contiguous set of observations between time a and b inclusive as $x_{a:b}$. (our primary concern is in the detection of the first change point, so the partition: p is only up to 2).

We extend our model to high-dimensional times-series with k-sparse changepoint: $x_{n,t}$. n is the index of the dimensions. At the changepoint: $t = A_1$. There exists $K \in \{1, \dots, n\}$ (typically k is much smaller than p), such that the compliment of K : K^C represents the indexes of the dimensions which experience no change at all (i.e. $n_1 = n_2$)

Our primary goal in this problem model, is to assume the changepoint represent a hazardous event that occurs in real time. The data: x_t will arrive one at a time. By observing the data point sequentially, we are to declare that a changepoint has happened as soon as possible subjected to false positive constrain. In order to stop the process responsible for generating the data.

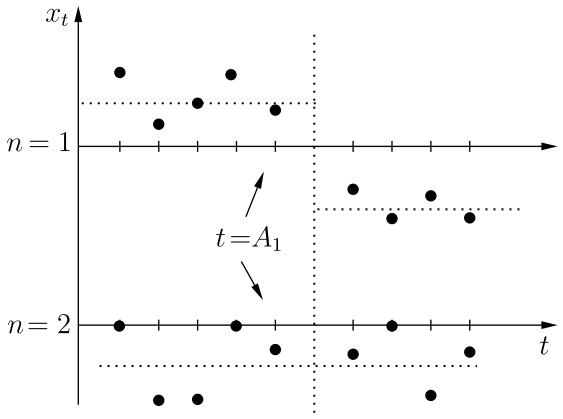


Figure 1: This figure illustrate how we describe a k-sparsed changepoint: $K = \{1\}$, $K^C = \{2\}$ for the 2D time-series

Our subsidiary goal is to locate the changepoint in time and to estimate the sparsity: K. As well as other distribution properties

Online/sequential/recursive learning

An online/sequential/recursive learning method is one in which data becomes available in sequential order and is used to update our best predictor for future data at each step. As oppose to the traditioanl/batch/retrospective learning methods which generate the best predictor by learning on the entire training data set at once.

It is used in area where: reaction speed is vital, computationally infeasible to train over the entire data set (the majority of retrospective learning algorithm have a total complexity of $O(n^2d^2+nd^3)$ (n is total points, d is dimensions). Area such as industrial quality control, robotics, brain-computer interfacing, navigation or finance. Sometimes the data itself is generated as a function of time.

[1] Shai (2007) describe a novel framework for the design and analysis of online learning algorithms. [4] Schuurmans and Greiner (1995) presents theoretical results that proves that the worst-lose expected sample complexity for online learning (specifically for PAC learning). Scales in essentially the same manner as retrospective learning in most situations and presented empirical evidence that shows that an online learner can require significantly fewer samples in practice; often by order of magnitude. Most formally-described retrospective learning algorithms first compute how many training samples are required to learn a target in the worst possible case before observing training data at once. In many situations, the computed worst-case bounds can lead to very inefficient use of training data, In comparison with online.

Bayesian Motivation

Consider a pair of random variables θ and z governed by a joint distribution $p(z, \theta)$. The conditional expectation of z given θ defines a deterministic function:

$$f(\theta) \equiv \mathbb{E}[z \mid \theta] = \int zp(z \mid \theta)dz \tag{1}$$

In Robbins and Manro, [5] (1951). An general online procedure for find the root θ^* in figure 2 is proposed.

If we have a large data set, we could use frequentist regression function model directly to obtain θ^*. However, suppose that we observe z one at a time. We wish to find a corresponding online scheme for θ^*. Given from [5], assume:

$$\mathbb{E}\big[(z - f)^2 \mid \theta\big] < \infty \tag{2}$$

without loss of generality assume also: $f(\theta) > 0$ if $\theta > \theta^*$ and $f(\theta) < 0$ for $\theta < \theta^*$. [5] defines:

$$\theta^{(N)} = \theta^{(N-1)} - a_{N-1}z(\theta^{(N-1)}) \tag{3}$$

where $z(\theta^{(N)})$ is an observed value of z when $\theta = \theta^{(N)}$. [5] shows that the sequence (3) converges to θ^*. If the coefficients a_n obey:

$$\lim_{N\to\infty} a_N = 0 \tag{4}$$

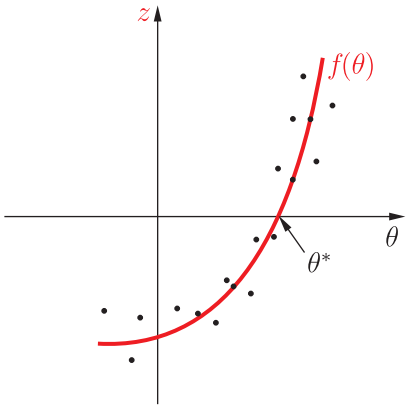


Figure 2: Two correlated randon variable z vs θ given by $\mathbb{E}[z \mid \theta]$ (1)

$$\sum_{N=1}^{\infty} a_N = \infty \tag{5}$$

$$\sum_{N=1}^{\infty} a_N^2 < \infty \tag{6}$$

(4) ensure successive correction decrease in magnitude so that the sequence (3) have a limiting convergence (5) ensure that the sequence does not converge unless at the root. (6) is needed so that the accumulated noise has finite variance and hence doesn't disrupt convergence. By definition, max likelihood solution θ_{ML} is the stationary point of the negative log likelihood function:

$$\begin{aligned} \frac{\partial}{\partial \theta}\left\{-\frac{1}{N}\sum_{1}^{N}\ln p(x_n \mid \theta)\right\}\Bigg|_{\theta_{ML}} &= 0 \\ = \mathbb{E}_x\left[-\frac{\partial}{\partial \theta}\ln p(x \mid \theta)\right] & \end{aligned} \tag{7}$$

which is in the form of (1). We therefore apply (3), which takes the form:

$$\theta^{(N)} = \theta^{(N-1)} - a_{N-1}\frac{\partial}{\partial \theta^{(N-1)}}\Big[-\ln p\big(x_N \mid \theta^{(N-1)}\big)\Big] \tag{8}$$

where $\dfrac{\partial}{\partial \theta^{N-1}}[-\ln p\big(x_N \mid \theta^{(N-1)}\big)]$.

In the specific example of sequential estimation of mean of a gaussian distribution: $f(x) \equiv \mathbb{E}[\mu_{ML} \mid x]$

$$z = \frac{\partial}{\partial \mu_{ML}}\Big[-\ln p\big(x \mid \mu_{ML}, \sigma^2\big)\Big] = -\frac{1}{\sigma^2}(x - \mu_{ML}) \tag{9}$$

provided we choose our a_N to have the form $a_N = \sigma^2/N$:

$$\begin{aligned} \mu_{ML}^{(N)} &= \mu_{ML}^{(N-1)} + \frac{1}{N}\Big(x_N - \mu_{ML}^{(N-1)}\Big) \\ &= \frac{1}{N}x_N + \frac{N-1}{N}\mu_{ML}^{(N-1)} \\ &= \frac{1}{N}x_N + \frac{N-1}{N}\sum_{n=1}^{N-1} x_n \end{aligned}$$

$$= \frac{1}{N} x_N \sum_{1}^{N} x_n$$

which is a well-known result for off-line estimation.

Bayesian computation

Suppose we have a probabilistic model $p(y \mid w)$, parameterised by w. On which we have prior distribution $p(w)$. We have observed a set of data D consisting of N i.i.d observation $\{y_n\}_{n=1}^{N}$. Bayes' theorem say: (posterior α likelihood x prior)

$$p(w \mid D) = \frac{p(D \mid w)p(w)}{p(D)} \tag{10}$$

after which the predictions can then be made averaging over parameter space:

$$p(y \mid D) = \int dw\, p(y \mid w)p(w \mid D) \tag{11}$$

The numerical computation for (11) is very expensive in memory. Even with substantial quantization, especially if the distribution is not closed.

Often, a close approximation of the true distribution $p(w)$ with the "conjugacy properties" is used.

In many cases, however, we may have little idea od what form the form the distribution should take. [6] Snelson and Ghahramani (2005) provided a general framework for learning a non-informative prior. The framework is based on minimizing the KL divergence between the predictive distribution and the approximating distribution. The outcomes are parameters that represent compactly and sufficiently accurately of the predictive distribution. These explicit parametric representation is cheap in memory and significantly reduce prediction time.

For more general distribution shape, other nonparametric methods are being developed. (Walker et al., 1999; Neal, 2000; Teh et al., 2006) (10). Notice that if you reduce the data set D to a singular data point. We see that bayesian paradigm leads very naturally to an online view of the inference problem.

Kullback-Leibler divergence

KL divergence (Kullback and Leibler, 1951) or relative entropy is given by:

$$KL(p\|q) = -\int p(x) \ln \left\{ \frac{q(x)}{p(x)} \right\} dx. \tag{12}$$

Bishop (2007) [8], provided a proof that $KL(p\|q) \geq 0$ with equality if, and only if, $p(x) = q(x)$. Therefore we can interpret the KL divergence as a measure of the 2 distribution $p(x)$ and $q(q)$.

Note that $KL(p\|q) \not\equiv KL(q\|p)$, there 3 to say that it is not a symmetrical quantity. Mckay (2003) [7] interpreted KL divergence: $KL(p\|q)$ as the average addition information in (bit or nats (base on the bases of the log)) that is required to communicate the value of x as a result of basing your loading sheme on $q(x)$ is the true distribution is $p(x)$. $p(x)$ is the distribution that you know less about i.e. the distribution of the signal after the change point should be interpreted as $p(x)$. This is vital, in order to use KL-divergence correctly as I shall illustrate later.

ID online change-point detection

Every iteration during a bayesian computation condenses all the observation so far into a finite set of summarizing values. I shall explain how this comes to be and how to track and extrapolate these values in order to solve the change point soonest declaration problem.

Adam and Mackay (2006) [3] proposed a unique formulation of the changepoint problem: instead of being concerned with retrospectively placing changepoints, the algorithms is only interested in the most recent change. The data from the present back to the most recent changepoint is called a run. The run length: r, is the number of data point in this run. At any time t, there are t possible runs that can be associated with the most recent datum. The clever part of this model, is that there are only 2 possibilities that can happen to the run length at each datum. Either the new datum joins the current run (i.e. r increase by 1) or the new datum starts a new run (i.e. r return to 0).

So, the posterior distribution is: $P(r_t \mid x_{1:t})$

$$P\big(r_t \mid x_{1:t}\big) = \frac{P(r_t, x_{1:t})}{P(x_{1:t})} \tag{13}$$

where:

$$\begin{aligned} P(r_t, x_{1:t}) &= \sum_{r_{t-1}} P\big(r_t, r_{t-1}, x_{1:t}\big) \\ &= \sum_{r_{t-1}} P\big(r_t, x_t \mid r_{t-1}, x_{1:t-1}\big) P\big(r_{t-1}, x_{1:t-1}\big) \\ &= \sum_{r_{t-1}} p(r_t \mid r_{t-1}) \underbrace{P\big(x_t \mid r_{t-1}, x_t^{(r)}\big)}_{\pi_t^{(r)}} \underbrace{p\big(r_{t-1}, x_{1:t-1}\big)}_{\text{recursive prior}} \end{aligned} \tag{14}$$

where

$$P\big(x_t \mid r_{t-1}, x_t^{(r)}\big) \equiv \pi_t^{(r)} \tag{15}$$

can be considered as te likelihood of the datum.

$P(r_{t-1}, x_{1:t-1})$ can be considered as our recursive prior. And we can set

$$P(r_{t-1}, x_{1:t-1}) = 1 \quad \text{at } t = 1, \tag{16}$$

as initializing condition. Since the probability of the run length being zero is 1 before the data is being generated.

$P(r_t \mid r_{t-1})$ can be considered as the prior likelihood for change to occur:

$$P(r_t \mid r_{t-1}) = \begin{cases} H(r_{t-1}+1) & \text{if} \quad r_t = 0 \\ 1 - H(r_{t-1}+1) & \text{if} \quad r_t = r_{t-1}+1 \\ 0 & \text{if} \quad \text{othervise} \end{cases} \tag{17}$$

$H(r)$ is the hazard function [9]. In this case, it can be interpreted as the probability for a changepoint to occur provided that no changepoint has occur already. The hazard function [9]:

$$H(r) = \frac{P(r)}{S(r)} \tag{18}$$

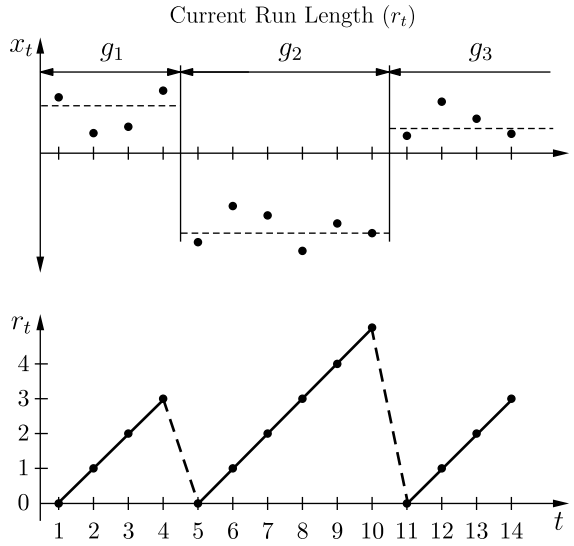


Figure 3: Illustration of the run length formulation of the changepoint problem

where $P(r)$ is the density function [9] abs $S(r)$ is the survival function these can be estimated offline using recent subset of the data, using survival analysis technique, such as Kaplan-Meier and Nekon-Aaka Estimators [9].

Alternatively, Kelsey Craing Anderson [10] (2008) attempted to expand on A&M's algorithm by implementing an extra layer of online inference of the hazard function in parallel with the online change point detection. This hugely increase the time complexity. It is only suitable for a specific class of multiple change point problem and the increase on performance is only significant in the much later changepoints. And it is parametric.

Alternatively, and in this essay, we can assume that the data generating process is memoryless and the hazard function is constant at $H(r) = \frac{1}{\lambda}$ for all r. λ is the timescale of the expected survival. And we can study empirically the affect it has on the performance on the soonest declare problem.

One easy and affective compromise would be to use Kaplan-Meier Estimator [9] on a recent subset of data to obtain $S(r)$ and then replace the initializing prior (16) to be $P(r_0) = S(r)$. And then set λ to be equal to the average survival time (i.e. run length).

Conjugate-Exponential models

Finally, we must infer the likelihood of the datum online: $\pi_t^{(r)}$. The properties of conjugacy: the posterior and the multiple of the likelihood and prior must share the same form. Such that the resultant distribution share the same set of sufficient statistics [3] which can be calculated incrementally as data arrives. Adam and MacKay [3] suggested exponential-family-likelihoods which ha the form:

$$P(x \mid \eta) = h(x) \exp\left(\eta^T U(x) - A(\eta)\right) \tag{19}$$

where

$$A(\eta) = \log \int dx\, h(x) \exp\left(\eta^T U(x)\right) \tag{20}$$

In this representation both the prior and posterior take the form of an exponential-family distribution over η that can be summarized by succinct hyperparameters ν and χ.

$$P(\eta \mid \chi, \nu) = \tilde{h}(\eta) \exp\left(\eta^T \chi - \nu A(\eta) - \tilde{A}(\chi, \nu)\right).$$

We wish to infer the parameter vector η associated with the data from this current run length r_t. We denote this run-specific parameter as $\eta_t^{(r)}$. After finding the posterior distribution $P(\eta_t^{(r)} \mid r_t, x_t^{(r)})$, we can marginalize through the parameters to find the predictive distribution, conditional on the length of current run

$$P\big(x_{t+1} \mid r_t\big) = \int d\eta \, P\big(x_{tt1} \mid \eta\big) P\Big(\eta_t^{(r)} = \eta \mid r_t, x_t^{(r)}\Big)$$

Complexity

Looking at figure 4, the trellis demonstrate clearly that at each iteration only the columns of probability values from the previous timestep is required, each element in the columns require exactly 2 computation to advance the algorithm. Since the size of the columns grow linearly with time. The space and time complexity per timestep are linear in the number of data points so far observed. We can modify the algorithm such that the complexity is reduced further by methods:

1) We can discard the normalized run length probability estimates if it is less than a certain threshold, say 10^{-4}.
2) After a certain run length, discard the least probable element at the end of each time step. This will yield constant complexity per time-step.

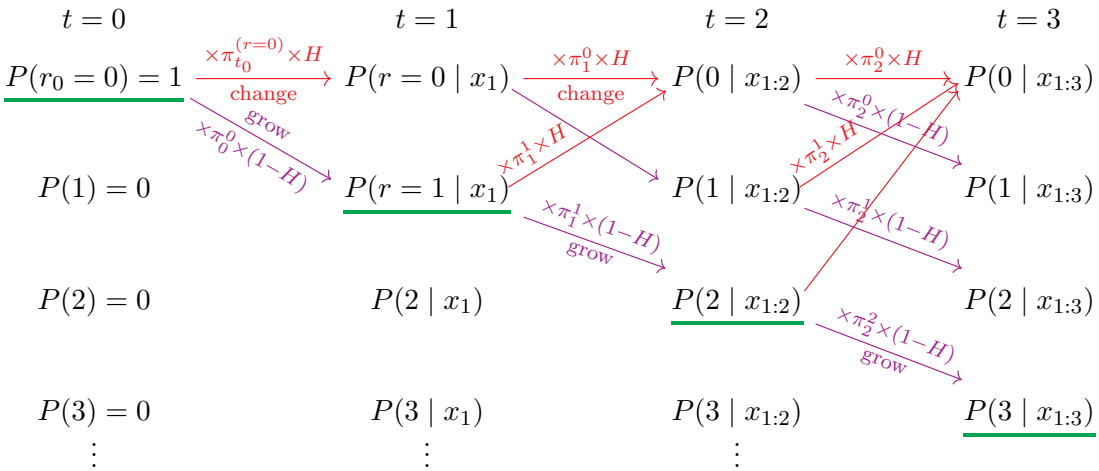


Figure 4: A complete illustration of all the calculation involve regarding the run length posterior for the first few iteration

A&M used 1), I decide to try 2), and the empirical result show almost no difference, especially in the singular change point case (which I am focusing on), where almost all the probability mass is concentrated to the bottom side of trellis (figure 4).

Change-point declaration

Notice the diagonal elements underline in green in figure 4. Those element represent the probability that no-change point has occurred (here $r_t = t$). $1 - P(r_t = t)$ therefore equal the probability that a change-point has occurred. From now on, I shall define $1 - P(r_t = t) = C_t$. And if C_t reaches 0.9, I would declare that a change point most likely has happened, and stop the process. Figure 5f will show a plot of C_t. The delay from real changepoint time, to declare time, will be the key performance we study.

Change-point location

Upon stoping at $C_t \geq 0.9$, the latest column from the last time step gives the best estimation of the run length distribution, which can be use to deduce the estimation of the changepoint location.

Normal-gamma prior

We will suppose that both the mean and the precision $\lambda = \sigma^{-2}$ are unknown.

Likelihood can be written in this form [11]

$$\begin{aligned} P(D \mid \mu_s \lambda) &= \frac{1}{(2\pi)^{\frac{n}{2}}} \lambda^{\frac{n}{2}} \exp\left(-\frac{\lambda}{2} \sum_{i=1}^{n} (x_i - \mu)^2\right) \\ &= \frac{1}{2\pi^{\frac{n}{2}}} \lambda^{\frac{n}{2}} \exp\left(-\frac{\lambda}{2}\left[n(\mu - \bar{x})^2 + \sum_{i=1}^{n} (x_i - \mu)^2\right]\right). \end{aligned}$$

Prior can be written:

$$NG\big(\mu, \lambda \mid \mu_0, K_0, \alpha_0, \beta_0\big) \stackrel{\text{def}}{=} \mathcal{N}\Big(\mu \mid \mu_0, (K_0\lambda)^{-1}\Big) Ga\big(\lambda \mid a_0, \text{rate} = \beta_0\big)$$

posterior:

$$P\big(\mu, \lambda \mid D\big) \alpha NG\big(\mu, \lambda \mid \mu_0, K_0, \alpha = \beta_0\big) P(D \mid \mu, \lambda).$$

Using derivation from [11]:

$$P(\mu, \lambda \mid D) = NG\big(\mu, \lambda \mid \mu_n, K_n, \alpha_n, \beta_b\big)$$

where:

$$\mu_n = \frac{K_0\mu_0 + n\bar{x}}{K_0 + n} \tag{21}$$

$$K_n = K_0 + n \tag{22}$$

$$\alpha_n = \alpha_0 + \frac{n}{2} \tag{23}$$

$$\beta_n = \beta_0 = \frac{K_0 n(\bar{x} - \mu_0)^2}{2(K_0 + n)} \tag{24}$$

The parameters for the likelihood:

$$\mu = \mu_n \tag{25}$$

$$\sigma = \frac{\beta K + 1}{\alpha K} \tag{26}$$

In the case of online learning, n can be set to 1, $\bar{x}$ is simply the datum value.

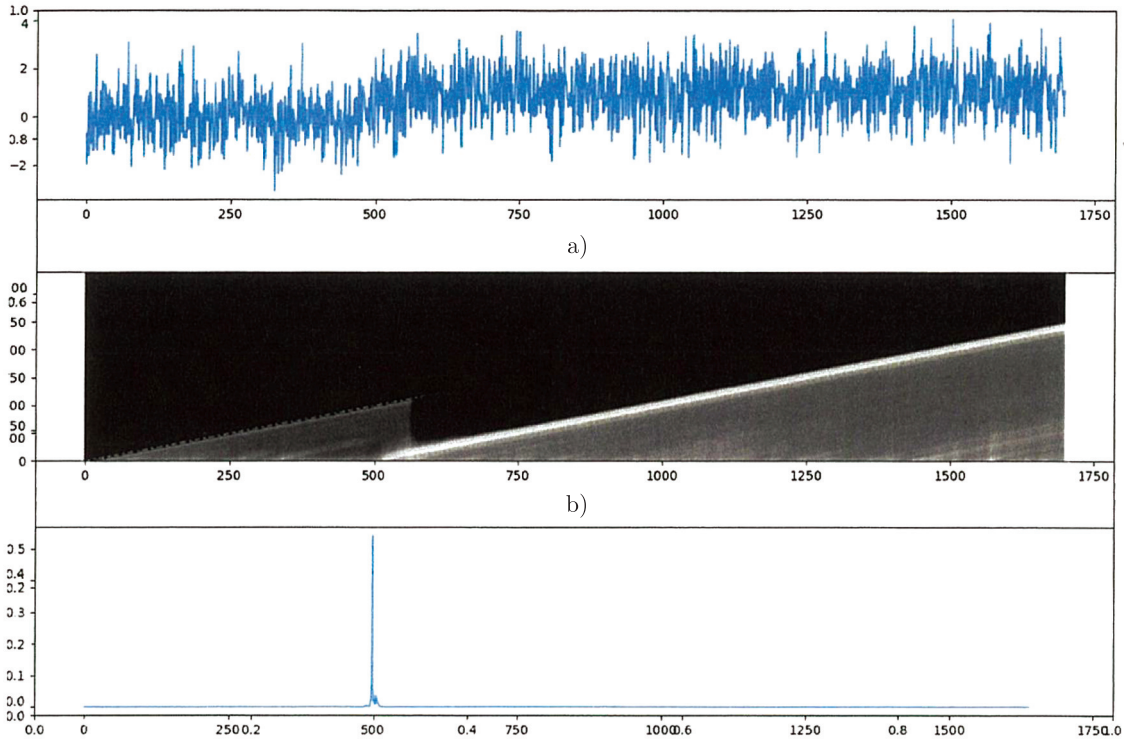
In light of (21)-(26), we can write fully a pseudo algorithm.

Algorithm 1: Algorithm for declaring changepoint and locating change point

1. Initialize $P(r_0 = 0) = 1$
   $\alpha_1^{(0)} = \alpha_0$
   $\beta_1^{(0)} = \beta_0$
   $K_1^{(0} = K_0 \qquad \mu_1^{(0)} = \mu_0$
2. Observe Neis Datum X_t
3. Evaluate likelihood
$$\pi_t^{(r)} = P\Big(x_t \mid \alpha_t^{(r)}, \beta_t(r), K_t(r), \mu_t(r)\Big)$$
4. Calculate growth probability
$$P\big(r_t = r_{t-1} \mid x_{1:t}\big) = P\big(r_{t-1}, x_{1:t-1}\big)\pi_t^{(r)}(1-H)$$
5. Calculate changepoint probability
$$P\big(r_t = 0, x_{1:t}\big) = \sum_{r_{t-1}} P\big(r_{t-1}, x_{1:t-1}\big)\pi_t^{(r)} H$$
6. Normalize:
$$P(r_t \mid x_{1:t}) = \frac{P(r_t, x_{1:t})}{P(x_{1:t})}$$
7. Update posterior parameters:
$$\mu_{t+1}^{(r+1)} = \frac{K_t^{(r)}\mu_t^{(r)} + x_t}{K_t^{(r)} + 1}$$
$$K_{t+1}^{(r+1)} = K_t^{(r)} + 1$$
$$\alpha_{t+1}^{(r+1)} = \alpha_t^{(r)} + \frac{1}{2}$$
$$\beta_{t+1}^{(r+1)} = \beta_t^{(r)} + \frac{K_t^{(r)}(x_t - \mu_t^{(r)})^2}{2(K_t^{(r)} + 1)}$$
8. Check if $P(r_t = t \mid x_{1:t}) < 0.1$:
   if yes go to 9
   if not return to 2
9. Output: t, $P(r_t \mid x_{1:t})$, μ, K, α, β

Numerical studies

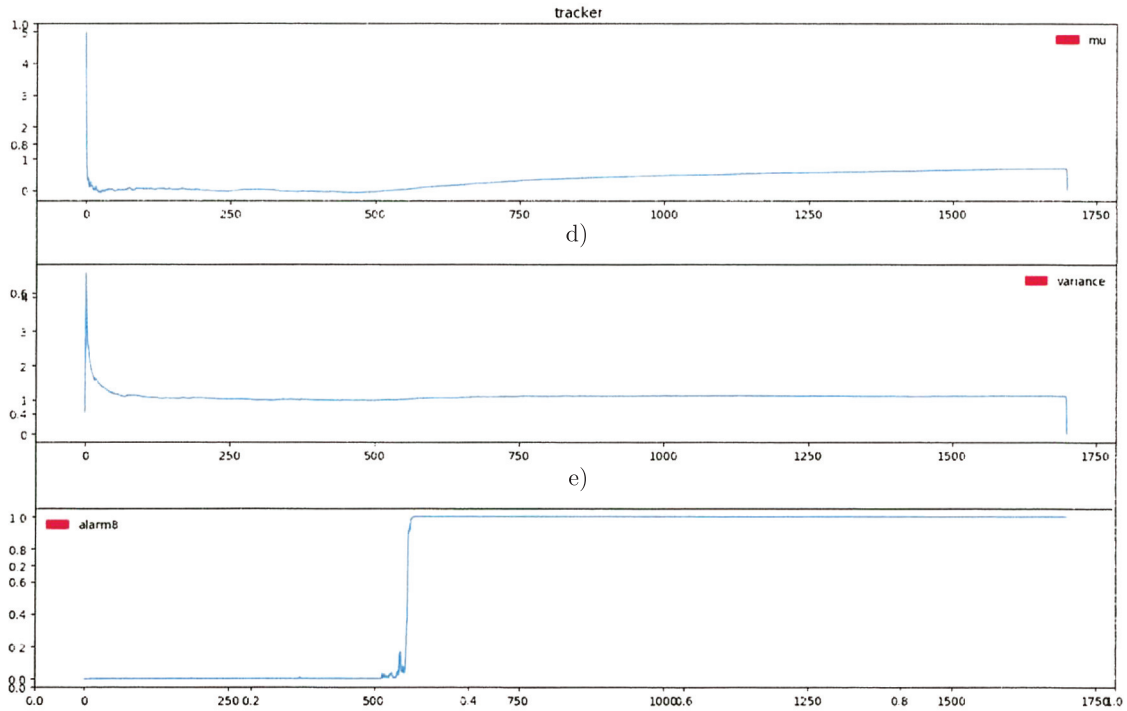
I have implemented the algorithm I described fully using python 2.7. Figure 5 and 7 and 8 illustrate some key insight as to how the algorithm works.



a)

b)

c)



d) e) f)

```
  -np.log(R[2:-1:sparsity, 2:-1:sparsity]),
all need to log parameters:  0 1 1 1 500
change declare at: [559]
change point location(most likely):   499
KL:  0.5
KL (2):  0.5
Lamda_gap 500
```

Figure 5

Artificial data generation

To test the performance of the algorithm, I concatenated 2 time series, each datum from each time series is generated i.i.d based on a normal distribution with a pair of mean and variance assigned to each time series.

In figure 5a, shows the raw signal generated at mean zero, standard derivation 1, changed at $t = 500$, into mean $= 1$, $\sigma = 1$.

Figure 5b shows the trellis from figure 4, with the higher probability density given a brighter colour.

Figure 5c show the probability distribution of the location that is inferred using the technique describe in the last chapter Using only the final column of the rum length distribution before a change is declared, which in this case was $t = 551$.

Figure 5d shows the online inference of the mean of the student-T distribution. Even though the prior initialize the mean at 5, instead of the true mean: zero, the updated mean quickly converge to the true mean within the first few time-step and stabilize, after passing the change-point at $t = 50$, it then start slowly converging toward the mean of the second time series but much more slowly. In reality the convergence can be more much faster by segmenting the time series, upon declaring a change point. And resetting K and α to be K_0 and α_0, see figure 6.

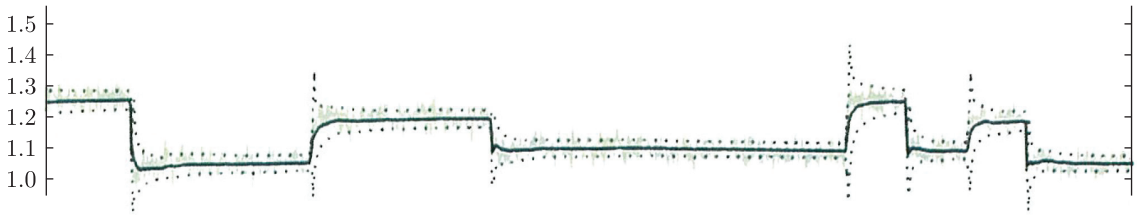


Figure 6

Figure 5e shows the same thing, except for variance. The variance also initialized at $\sigma_0 = 5$ Except there is no change in variance after the change point as expected.

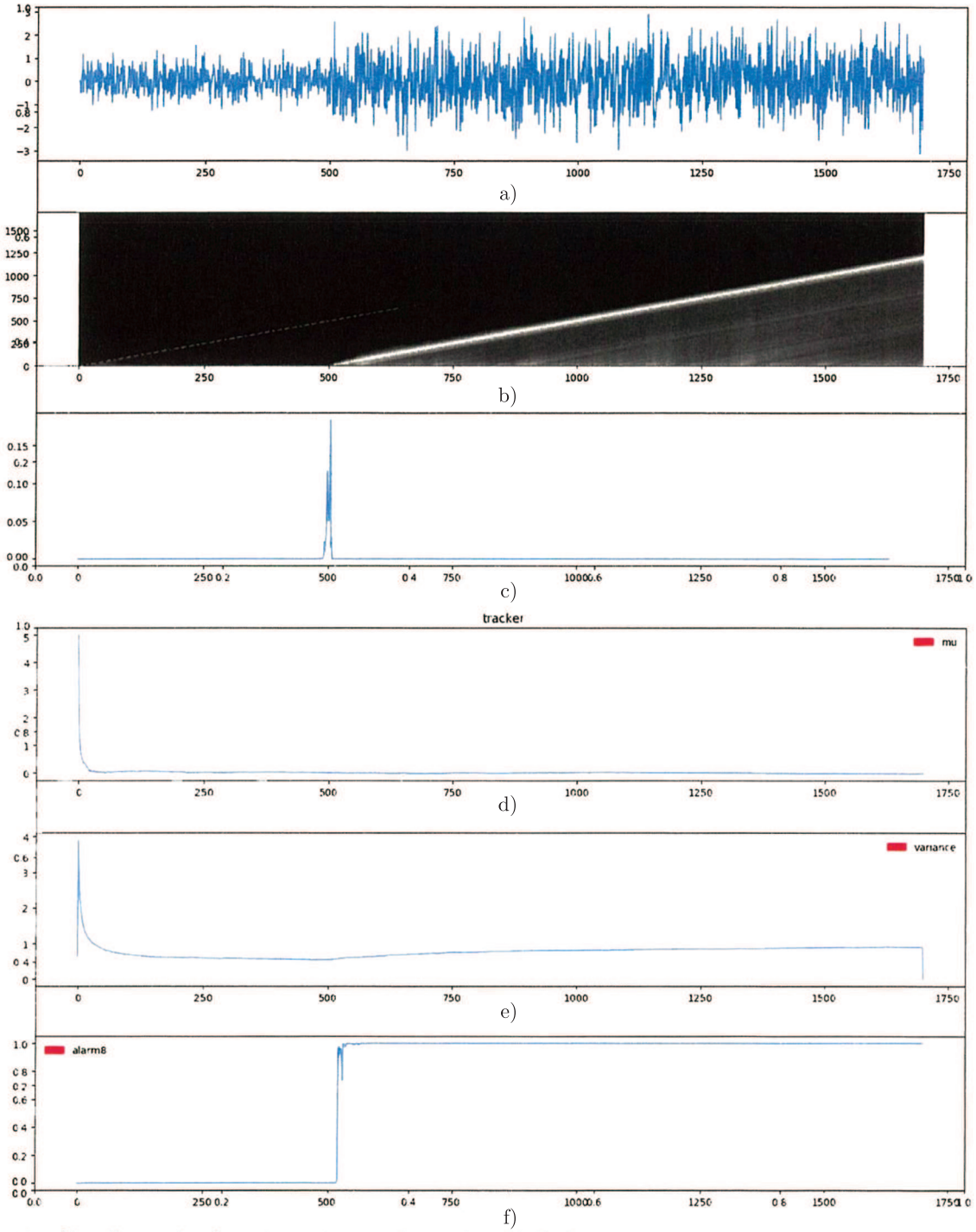
Figure 5f shows the key performance plot of C_t. Which is the confidence that we have that the change point happened. When it reaches 0.9, we declare that we found a change and record the time and stop the process. The delay between the actual change-point and declare would be our key performance indicator.

Figure 7 shows essentially the same thing, except here there is no mean changes this time. Only variance changed from 0.5 to 1 at $t = 500$. We see that this algorithm can detect variance changes just as well as mean changes. This properties cannot be found in many of the CUSUM based change point detection algorithm.

I written automated script in order to repeatedly run the detection large amount of time. Table 1 shows the portion of the test that I ran.

$m1:$ mean before change
$m2:$ mean after change
$o1:$ set. dev. before
$o2:$ set. dev. after
$t1:$ actual change point

} these are the input to the data generation




```
1D_online_changepoint_detection.py:227: RuntimeWarning: divide by
zero encountered in log
  -np.log(R[2:-1:sparsity, 2:-1:sparsity]),
all need to log parameters:  0 0 0.5 1 500
change declare at: [521]
change point location(most likely):   517
KL:  0.31814718056
KL (2):  0.80685281944
```

Figure 7

l_gap :	λ as in the $H_{GS} = \frac{1}{r}$	these are the "tuning" parameter
pre_Ma :	this is the initialization mean prior	
pre_V :	this is the initialization variance prior	
M_a :	the declare time	main results
$r_m$se_a :	the root mean square of the delay time	
M_Z :	estimated change point location	

Tally: how many time did I repeat with the same set of input and "tuning" parameter.

false_tally: tally of the "false positive", when the declare is before the known change point.

Figure 8 is the C_t plot of 7 different detection on 7 different change point plot. 8a has $0 \to 1$ mean step size. 8b has $0 \to 0.5$. 8c has $0 \to 0.4$. (While keeping variance constant).

8d has $0.5 \to 1$ set. dev. 8c has $0.7 \to 1$, 8f has $0.75 \to 1$ (all while keeping mean constant), as the 2 distribution get more and more similar, the C_t plot takes longer and longer to reach 1. Until when there is no difference, then the C_t plot is flat.

I discovered that the most predictive way to summarize the combine power of mean step size and variance change, is the $KL(p\|q)$, the divergence between the distribution before and after the change.

Let $p(x) = N(\mu_1, \sigma_1)$ and $q(x) = N(\mu_2, \sigma_2)$.

$$KL(p\|q) = -\int p(x)\log q(x)\,dx + \int p(x)\log p(x)\,dx$$

where integration is done overall real line,

$$\int p(x)\log p(x)\,dx = \frac{1}{2}(1+\log 2\pi\sigma_1^2) \qquad \text{LHS}$$

$$-\int p(x)\log\frac{1}{(2\pi\sigma_2^2)^{\frac{1}{2}}}e^{\frac{-(x-\mu_2)^2}{2\sigma_2^2}}\,dx, \qquad \text{RHS}$$

which I can separate into

$$= \frac{1}{2}\log(2\pi\sigma_2^2) - \int p(x)\log e^{\frac{-(x-\mu_2)^2}{2\sigma_2^2}}\,dx$$

$$= \frac{1}{2}\log(2\pi\sigma_2^2) + \frac{\int p(x)x^2dx - \int p(x)2x\mu dx + \int p(x)\mu^2 dx}{2\sigma_2^2}$$

$$= \frac{1}{2}\log(2\pi\sigma_2^2) + \frac{\sigma_1^2+\mu_1^2-2\mu_1\mu_2+\mu_2^2}{2\sigma_2^2}$$

$$= \frac{1}{2}\log(2\pi\sigma_2^2) + \frac{\sigma_1^2(\mu_1-\mu_2)^2}{2\sigma_2^2}$$

LHS+RHS:

$$\begin{aligned} KL(p,q) &= -\int p(x)\log q(x)dx + \int p(x)\log p(x)dx \\ &= \frac{1}{2}\log(2\pi\sigma_2^2) + \frac{\sigma_1^2+(\mu_1-\mu_2)^2}{2\sigma_2^2} - \frac{1}{2}(1+\log 2\pi\sigma_1^2) \\ &= \log\frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2+(\mu_1-\mu_2)^2}{2\sigma_2^2} - \frac{1}{2} \end{aligned} \tag{27}$$

Table 1

m1	m2	o1	o2	t1	l_gap	kl	kl2	rmse_a	rmse_z	m_a	m_z	pre_Ma	pre_V	tally	falsc_tally
0	1	1	1	500	50	0.5	0.5	9.3	110.13	517.25	489.16	0	10.1	12	8
0	0.8	1	1	500	500	0.32	0.32	16.5	5.96	547.35	501.05	0	1010	20	0
0	0.5	1	1	500	500	0.12	0.12	44.52	20.82	574.421053	509.78	0	10.1	19	1
0	0	0.65	1	500	500	0.14	0.25	41.47	11.40	571.75	507.1	0	1010	20	0
0	0	0.75	1	500	500	0.06	0.10	241.96	23.24	759.95	497.9	0	1010	20	0
0	1	1	1	100	200	0.5	0.5	8.55	226.38	126.75	154.7	0	10.1	20	0
0	1	1	1	1000	10000	0.5	0.5	7.58	2.86	1027	1001.9	0	10.1	20	0
0	1	1	1	500	500000	0.5	0.5	14.20	3.76	545.5	501.45	0	10.1	20	0
0	1	1	1	1000	100	0.5	0.5	7.75	2.72	1020.09	1001.27	0	10.1	11	9
0		0.7	1	500	500	0.10	0.16	69.50	154.84	583.12	532.97	0	1010	40	0
0	1	1	1	250	1000	0.5	0.5	10.03	178.51	275.38	280	0	10.1	39	1
0	1	1	1	150	1000	0.5	0.5	9.9	10.70	175.4	147.9	0	10.1	20	0
0	1	1	1	500	5000	0.5	0.5	8.05	2.44	524.7	501.75	0	10.1	20	0
0		0.5	1	500	500	0.31	0.80	7.61	89.29	513.21	524.84	0	10.1	19	1
0	1	1	1	100	100	0.5	0.5	12.97	22.77	122.66	98.94	0	10.1	18	2
0	0.9	1	1	500	500	0.405	0.405	19.42	8.50	543.3	499.95	0	1010	20	0
0	1	1	1	10	1000	0.5	0.5	416.42	2.79	1039.6	7.05	0	10.1	20	0
0	1	1	1	1000	500	0.5	0.5	10.19	7.32	1024.42	1002.26	0	10.1	38	2
0	0.4	1	1	500	500	0.08	0.08	160.80	40.52	652.63	516.36	0	10.1	19	1
0	1	1	1	1000	1000	0.5	0.5	8.73	120.68	1021.85	1028.55	0	10.1	20	0
0	1	1	1	500	500	0.5	0.5	9.38	102.85	524.16	512.89	0	10.1	78	2
0		0.55	1	500	500	0.24	0.55	14.44	7.47	528.05	503.85	0	1010	20	0
0	0.35	1	1	500	500	0.06	0.06	195.29	122.67	766.9	485.05	0	10.1	20	0
0	1	1	1	500	100	0.5	0.5	61.26	232.26	1686.3	150.2	20	10.1	20	0
0	1	1	1	100	1000	0.5	0.5	34.88	2.81	138.5	100.6	0	10.1	20	0
0	0.7	1	1	500	500	0.245	0.245	28.77	20.87	556.25	504.62	0	10.1	40	0

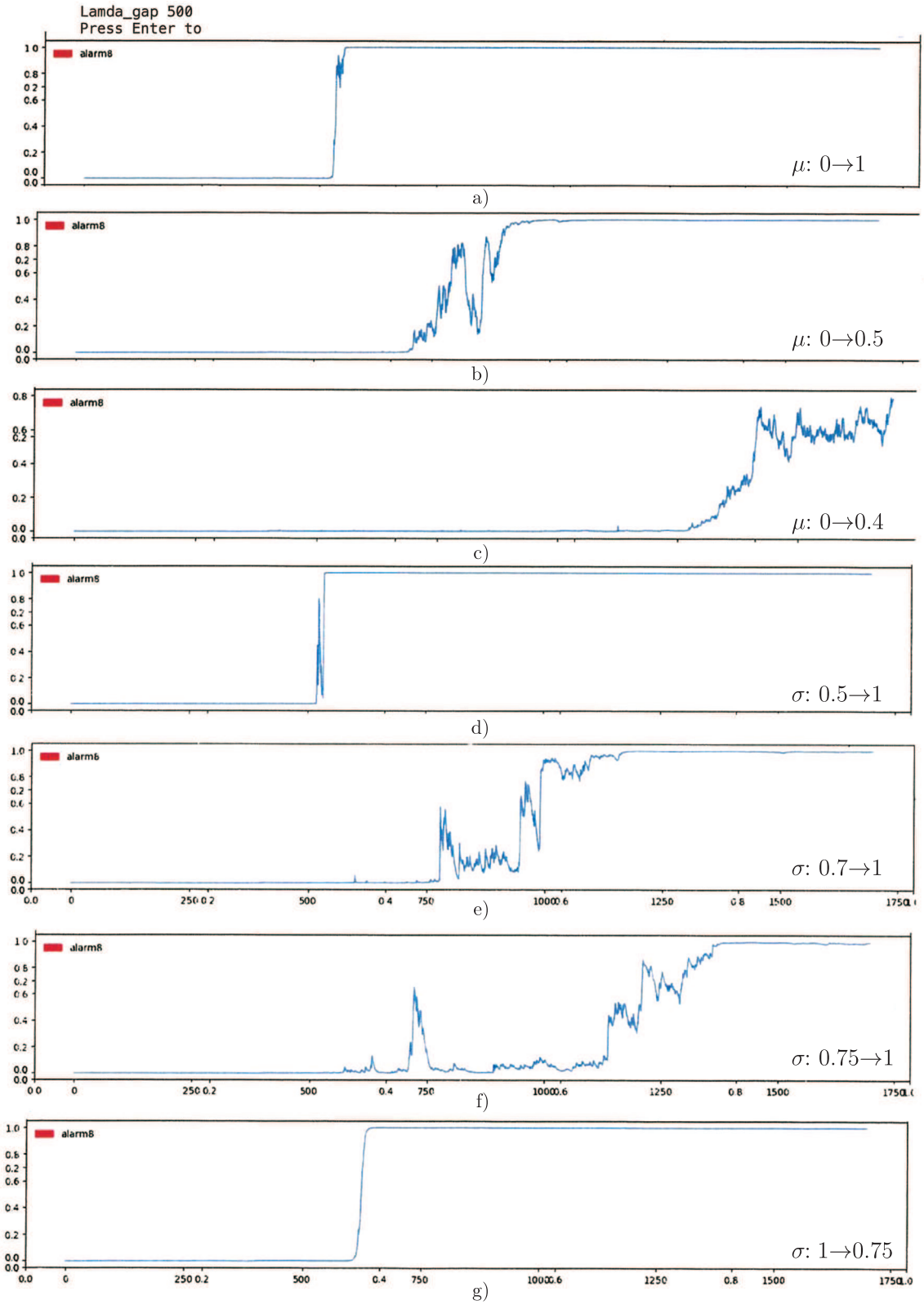

Lamda_gap 500
Press Enter to
alarm8
μ: 0→1
a)
alarm8
μ: 0→0.5
b)
alarm8
μ: 0→0.4
c)
alarm8
σ: 0.5→1
d)
alarm8
σ: 0.7→1
e)
alarm8
σ: 0.75→1
f)
alarm8
σ: 1→0.75
g)

Figure 8

Check that this $= 0$ when $\sigma_1, \mu_1 = \sigma_2, \mu_2$. As mention before, it is vital to assign the correct distribution to p, q. Since $KL(p,q) \neq KL(q,p)$.

In this case, q should be the distribution of the 1st timeseries. Which would explain figure 8g, where the variance step size is reverse that of 8d, but the detection for 8d was easier to detect, meaning it should have a higher KL-divergence.

Figure 9 shows the x axis, KL-diverge vs the y axis, the time of declare. Changepoint set as 500 variance: constant. mean: $0 \to 1$. $\lambda = 500$ and Pre_M $= 0$, Pre_V $= 10$.

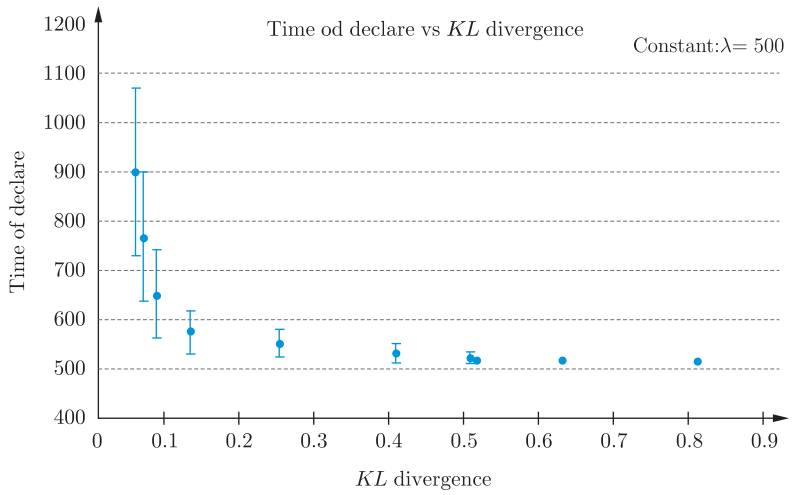


Figure 9

Notice the ever increasing KL-divergence make the detection problem increasingly easy. Regardless to whether the divergence originate form change in mean or change in variance or both.

α_0, β_0, μ_0, K prior

By numerical testing, and by studying figure 5d, 5e, 6d, 6e. We see that these parameters converges toward the true value very quickly, and since the process is online and memoryless, once it converges, it is as if you have started with the correct one to begin with.

One can simply set $H = 0$, until the convergence has pass a certain criteria before allowing the H to be non-zero. This would almost eliminate all probability of a false positive. However, it also renders the algorithm unable to detect change prior to said criteria is met. Like most bayesian algorithm, prior parameters value only really matter if there is very few data prior.

One observation however, $\sigma = \dfrac{\beta_0 K_0 + 1}{\alpha_0 + K_0}$, it you set σ_0 to be very small, false positive becomes very likely.

$H = \dfrac{1}{\lambda}$ prior

λ should ideally be of the order of t_1, (the time between change point), but upon inspection of testing, even if $\lambda = 1000t_1$, it only causes an increase of 1 to 5 delay in declaration time. However, if $\lambda < \dfrac{1}{2}t$, false positive starts to become very probable. Empirically speaking, I would set λ $10\times$ higher than I would expect the location of the change point. However, the accuracy of λ become very import when there is only 10-15 datum before change-point. In this case, λ can be considered as a "tuning" parameter, decrease it to increase detection sensitivity but at the risk of false positive.

High-dimensional online sparsed-changepoint detection

We want to extend the algorithm for high-dimensional case, in which the dimensions that participate in the "changing", is sparsed. The naive way is simply to do the 1-D online change-point detection on each individual dimension in parallel and observe the C_t plot, if enough of the dimensions has reached 0.9, we can declare.

But this is not the best we can do; suppose that each individual dimension only change by the tiniest magnitude, our study from the previous chapter shown that the delay before declare is exponentially bigger as change get smaller. We want to leverage our knowledge that: when 1 of the dimension experience change, a set of other dimensions also must change with it. Essentially to "borrow strength" from all the small changes across the set, to allow sooner detection. Which, as shown in the previous chapter, also equate to the better location of the change point, better inference to the change size.

Motivation

We already know the bigger the $K - L$ divergence is, between before and after the change point. The easier the detection problem is. Our goal is therefore to make the $K - L$ divergence larger using the dimensions provided. From (28), we have KL-divergent between 2 gaussian:

$$KL(p\|q) = \log \frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} + \frac{1}{2} \tag{28}$$

where σ_1, μ_1 is the standard deviation and mean of the distribution before changepoint. σ_2, μ_2 is likewise after the changepoint.

We can rewrite (28):

$$KL(p\|q) = \left(\log \frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2}{2\sigma_2^2}\right) + \left(\frac{(\mu_1 - \mu_2)^2}{2\sigma_2^2}\right) - \frac{1}{2} \tag{28.1}$$

It is easy to see the second term is simply the "signal to noise" ratio. We shall focus on maximizing this term. More specifically $(\mu_1 - \mu_2)^2$, since it does not affect the size of the other terms.

Let $(\mu_1 - \mu_2) = \Delta\mu$. Let $\Delta\mu_i =$ the mean change of the i^{th} dimension.

Hypothetically, if we do a changepoint detection on $X.V$ in figure 10, we will have a much easier detection. So we want to find V. $V = (V_1, \ldots, V_p)^T \in \mathbb{R}$, $V_i = 1$ if $\Delta\mu_i =$ positive, $V_i = 0$ if $\Delta\mu_i = 0$, $V_i = -1$ if $\Delta\mu_i =$ negative i.e. $(V_i = \text{sgn}(\Delta\mu_i))$.

We learnt that even the smallest change in the distribution would be register on the C_t plot, even though with a low slope (see Figure 8). Then assuming we know $|K|$ (size of the K see), then the dimension with top $|K|$ highest C_t are sure to give a good approximation of K. Therefore $V_{\{K\}} = \pm 1$. To decide on whether $+ve$ or $-ve$, we can look back to the sequence of mean update, i.e. (Figure 5e), an upward slope should imply a positive $\Delta\mu_i$ hence $V_i = \pm 1$, and vise versa for downward slope. $V_{\{K'\}} = 0$.

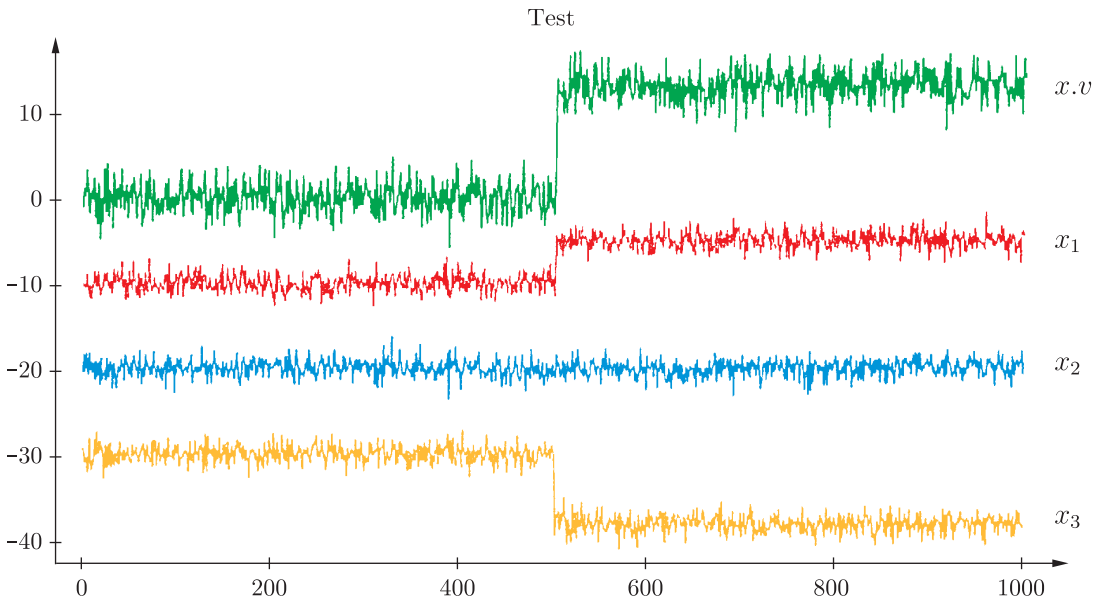


Figure 10: $X.V = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}$, $\Delta\mu_1 = 5$, $\Delta\mu_2 = 0$, $\Delta\mu_3 = -8$. $\Delta\mu$ for $X.V = 13$

Iterate through $|K|$

We do not, in fact, know what $|K|$ is, so we would have to iterate from $|K| = 1$, to $|K| = P$; we will create P extra projected direction. We will conduct a 1-D change point detection on each of these projected dimensions.

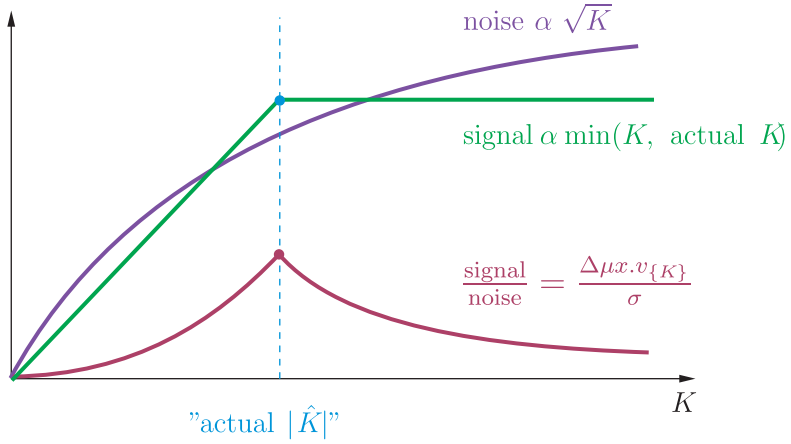


Figure 11

So in theory, the signal/noise ratio will peak at actual $|\hat{K}|$, therefore, the highest KL-divergent, therefore at "actual $|\hat{K}|$", We will have the soonest declaration when conducing ID online change point detection. Beating all the dimensions done individually. The resultant "actual $|\hat{K}|$" will also uncover how many "Independently, mean changed" dimension is present.

Notice that: this peak completely ignore dimension that doesn't have mean change (i.e. variance change only). Therefore, do not necessarily give the entire K.

Notice that: if the dimensions are correlated, the "actual $|\hat{K}|$" will over-estimate. This is likely to be the case, since the dimensions that change together during a change-point in reality, probably suggest that they are driven by the same hidden principle component.

Algorithm 2: Pseudo-code that "borrow strength" across the mean change direction

1. Initialize:
   α_{0_i}, β_{0_i}, K_{0_i}, μ_{0_i} for all dimension
2. Observe datum
3. Perform algorithm 1 on all P data dimension
4. Update μ and C_t plot for all P data dimension.
5. **FOR**
   $K = 1$ to $K = P$:
   find X_i with the highest C_t $\quad V_i = 1$
   if $\mu_{i_t} - \mu_{i_{t-1}} = -ve$ $\qquad V_i = -1*$
   all other $V - i = 0$
   create projected dimension $X_t \cdot V_t$
6. Perform algorithm 1 on all P project dimension
7. Update C_t for all P projected dimension with corresponding K
8. Check if any of the projected C_t is ≥ 0.9, if not, return to step 2
9. Output: declare time, K.

Numerical studies

I have implemented my proposed algorithm in python 2.7.

Figure 12 takes Figure 10 and compare it with algorithm finding, the green line is theoretical ideal projection of x_1, x_2, x_3 using $V = \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}$ consistently, the blue line is the projected x using algorithm 2, $K = 2$. The projection is very close to ideal; $X.V$ has $\sigma_2^2 = 2$. $\mu_2 = 0$, $\sigma_1^2 = 2$. $\mu_1 = 13$. $X\tilde{V}_{(K=2)}$ has $\sigma_2^2 = 2.14$, $\mu_2 = 0.04$, $\sigma_1^2 = 2.03$, $\mu_1 = 12.98$. The increase in σ_2^2 increase is to be expected, since the projection direction before the changepoint is approximately random and uniformly distributed, hence, adding to the overall variances of the projected signal

Figure 13 show a detection problem where $P = 6$, $|K| = 4$, $\Delta\mu = 0.25$, $K \in \{2, 3, 4, 5\}$ and their corresponding C_t plot, $X, \tilde{V}$ is the C_t plot of the projected direction in which was detected (i.e. $C_t > 0.9$) the soonest, which in this case was $X.\tilde{V}_{K=4}$ with $V = (0, 1, 1, 1, 1, 0)$, which was correct. These parallel plots shows that individually it would take way longer for the C_t plot to be > 0.9. Hence in this case, my algorithm has successfully "borrowed strength" from across the dimension.

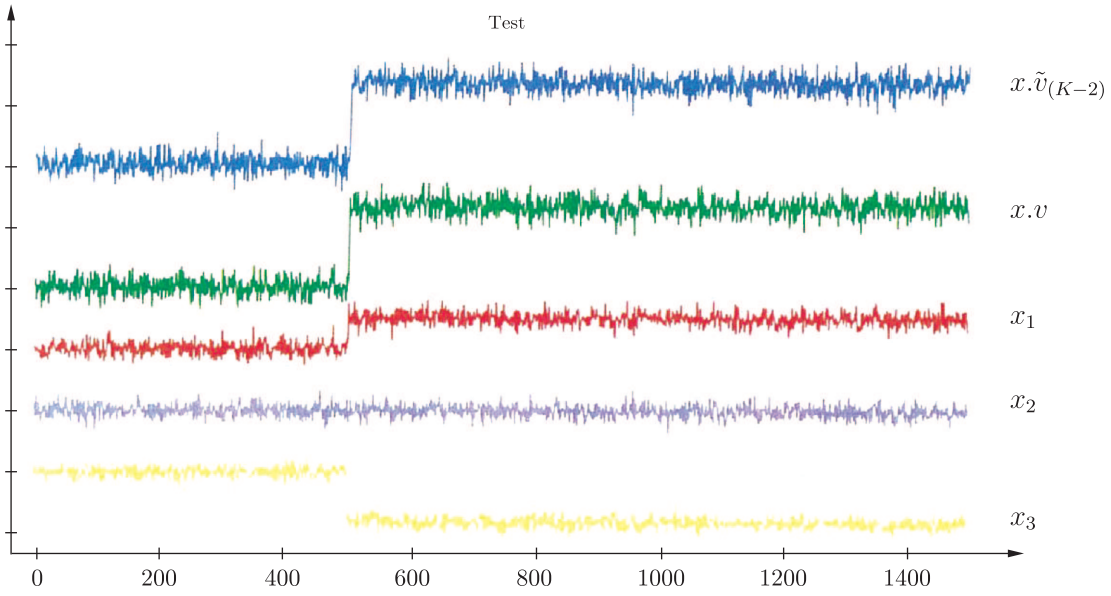


Figure 12

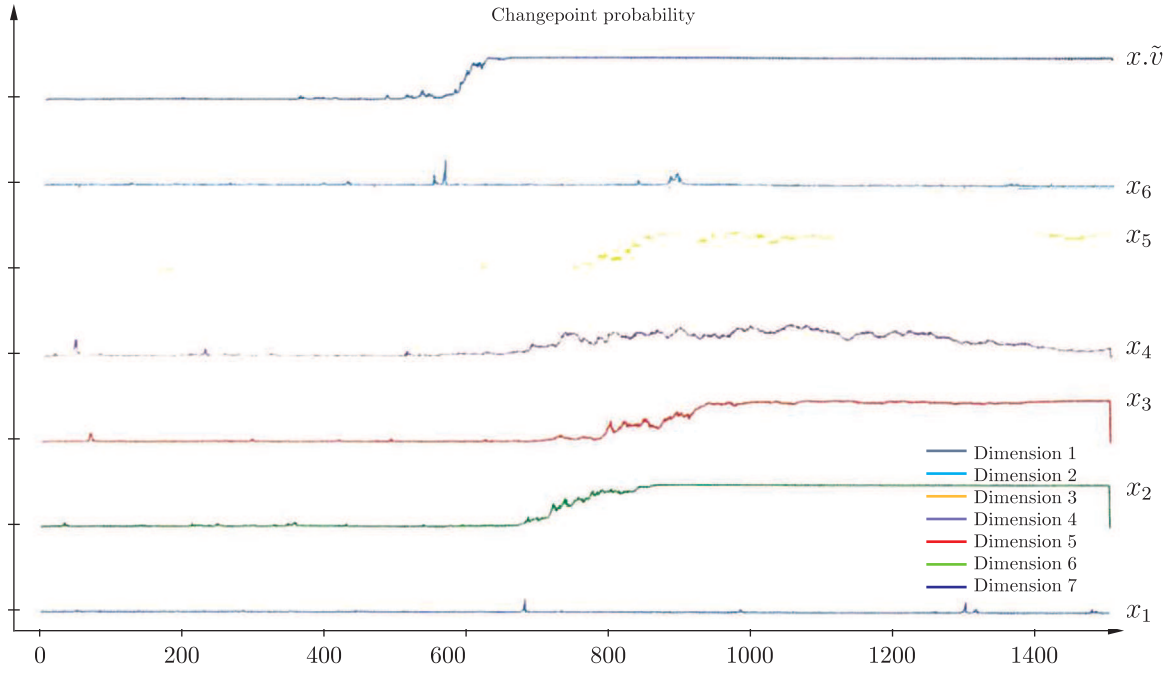


Figure 13

Table 2: $o = \Delta\mu_{\{K\}}$, t is time before change-point, $K = |K|$, $P =$ no of dimensions, $m\_k\_hat =$ the estimate K, $m\_percent$ the fraction of V_i that was guest correctly, m_delta_t is the average delay, tally is the number of attempt ran in order to achieve the averages. $false\_tally$ is the number of times the detector detects a changepoint before t, which we know must be a mistake.

As we can see, as k get bigger, there is more dimension to "borrow strength" from, allowing easier detection. $\Delta\mu$ of 0.2 if done in 1 dimension would result in ≈ 2000 in delay. So this improve in performance is significant.

As for m_k_hat and $m\_percent$, the process is stopped too early in the projected direction, in Figure 13 we know the C_t plot fluctuate a lot initially as the C_t builds up, this makes the $\tilde{V}$

Table 2: Show part of the result table

o	t	k	P	m_k_hat	$m\_percent$	m_delta_t	$tally$	$false\_tally$
0.2	500	4	10	4.92	0.73	312.30	13	0
0.2	500	5	10	7.60	0.56	261.20	10	0
0.2	500	6	10	8.10	0.58	232.60	5	0
0.2	500	8	10	8.72	0.70	71.27	11	0
0.2	500	9	10	7.54	0.69	41.18	11	0

inconsistent, if we allow the process to continue, we will see both of them converge toward the right answer rather quickly.

Possible extension

Algorithm 2 take advantage of only the signal/noise part of KL-divergent. If I have more time, I would propose an algorithm that instead of using the $|K|$ with the highest C_t as a selective criteria for V and μ_t to decide its sign, I would truncate N most recent data points, and then retrospectively search through all possible change-point location, doing a mini-change-point detection on each hypothetical change-point in order to deduce a hypothetical KL-divergence if a change point was to be there. Contain the highest KL-divergence for that dimension, and then rank it with other dimensions, use the ranking as the selective criteria for V. This way, I will make full use of the change in variance as well as mean. However, in this method, it is no longer completely online. It would be interesting to study how much "online-ness" does one have to sacrifice in exchange for how much improvement in performance.

Conclusion

I have successfully proposed and tested a novelty algorithm using bayesian reasoning which supersedes its frequentist predecessor by achieving the following properties simultaneously:

- Online learning, without any obvious drop in performance vs many frequentist's retrospective learning algorithm
- Lower time complexity
- Detects any distributional change, rather than just mean structure change.
- Generalized for all sparsed or non sparsed high dimensional change.
- A rough Online approximation of the sparsenss of the changepoint
- Improve sensitivity to subtle changes that otherwise cannot be detected by 1-D changepoint detection.

I have also demonstrated empirically, the limitations that is associated with pre-selecting a bad bayesian prior. However, I also demonstrated how inconsequential these limitations will become (unless in situation where the number of data points provided are unrealistically insufficient).

References

[1] Online Learning: Theory, Algorithms, and Applications. Thesis submitted for the degree of "Doctor of Philosophy" by Shai Shalev-Shwartz Submitted to the Senate of the Hebrew University July 2007. P

[2] D. Barry and J. A. Hartigan. Product partition models for change point problems. The Annals of Statistics, 20:260-279, 1992.

[3] Ryan P. Adams, David J.C. MacKay, Bayesian Online Changepoint Detection, arXiv 0710.3742 (2007)

[4] Sequential PAC Learning, Dale Schuurmans and Russell Greiner, Proceedings of the Eighth Annual Conference on Computational Learning Theory (COLT-95), Santa Cruz, July 1995.

[5] A Stochastic Approximation Method, Herbert Robbins and Sutton Monro, The Annals of Mathematical Statistics,Vol. 22, No. 3 (Sep., 1951), pp. 400-407

[6] Edward Snelson and Zoubin Ghahramani. Compact approximations to Bayesian predictive distributions. In 22nd International Conference on Machine Learning, Bonn, Germany, August 2005. Omnipress

[7] Information Theory, Inference and Learning Algorithms 1st Editionby David J. C. MacKay

[8] Pattern Recognition and Machine Learning (Information Science and Statistics) Oct 1, 2007 by Christopher M. Bishop

[9] Notes from Survival Analysis Cambridge Part III Mathematical Tripos 2012-2017 Lecturer: Peter Treasure Vivak Patel March 23, 2013

[10] A novel approach to Bayesian online changepoint detection, Kelsey Craig Anderson University of Colorado Boulder

[11] Conjugate Bayesian analysis of the Gaussian distribution, Kevin P. Murphy* murphyk@cs.ubc.ca October 3, 2007