Estimating the probability of event occurrence

ALEXANDRE GUINAUDEAU
Estimating the probability of event occurrence

ALEXANDRE GUINAUDEAU
Abstract

In complex systems anomalous behaviors can occur intermittently and stochastically. In this case, it is hard to diagnose real errors among spurious ones. These errors are often hard to troubleshoot and require close attention, but troubleshooting each occurrence is time-consuming and is not always an option.

In this thesis, we define two different models to estimate the underlying probability of occurrence of an error, one based on binary segmentation and null hypothesis testing, and the other one based on hidden Markov models. Given a threshold level of confidence, these models are tuned to trigger alerts when a change is detected with sufficiently high probability.

We generated events drawn from Bernoulli distributions emulating these anomalous behaviors to benchmark these two candidate models. Both models have the same sensitivity, $\delta_p \approx 10\%$, and delay, $\delta_t \approx 100$ observations, to detect change points. However, they do not generalize in the same way to broader problems and provide therefore two complementary solutions.
Sammanfattning

I komplexa system kan anomala beteenden uppträda intermittent och stokastiskt. I de här fallen är det svårt att diagnostisera verkliga fel bland falska sådana. Dessa fel är ofta svåra att felsöka och kräver noggrann uppmärksamhet, men felsökning av varje händelse är mycket tidskrävande och är inte alltid ett alternativ.


Vi genererade händelser som drogs från Bernoullifördelningar som emulerar dessa avvikande beteenden för att utvärdera dessa två kandidatmodeller. Båda modellerna har samma sensitivitet, $\delta_p \approx 10\%$ och fördöjning, $\delta_t \approx 100$ observationer, för att upptäcka ändringspunkter. De generaliserar emellertid inte på samma sätt till större problem och ger därför två kompletterande lösningar.
Contents

1 Introduction ................................................................. 1
  1.1 Problem statement ..................................................... 2
  1.2 Scope and objectives ................................................... 3
  1.3 Thesis overview ....................................................... 3

2 Background ........................................................................ 4
  2.1 Related Work .............................................................. 4
     2.1.1 Retrospective change point detection .................. 4
     2.1.2 Real time change point detection ......................... 5
     2.1.3 Binary segmentation ............................................. 6
     2.1.4 Corrections of the family-wise error rate ............ 6
     2.1.5 Hidden Markov models .......................................... 7
  2.2 Background on Binary Segmentation Model ................. 8
  2.3 Background on Hidden Markov Model ......................... 9

3 Methods ............................................................................ 11
  3.1 Data ........................................................................... 11
  3.2 Using a null hypothesis to tune our models ............... 12
  3.3 Tuning the Binary Segmentation Model ..................... 13
  3.4 Tuning the Hidden Markov Model ............................. 17

4 Results ............................................................................... 22
  4.1 Sensitivity ................................................................. 23
  4.2 Delay .......................................................................... 25
  4.3 Comparison to a standard model for event detection .... 26
      4.3.1 False positive rate of the Sliding Window Model .... 27
      4.3.2 Comparison to the sensitivity of the Sliding Window Model ........................................ 28
      4.3.3 Comparison to the delay of the Sliding Window Model ................................................. 29
Chapter 1

Introduction

In complex systems anomalous behaviors can occur intermittently and stochastically, which makes it hard to diagnose real errors among spurious ones. In many cases, troubleshooting each occurrence of a non-deterministic error is time-consuming. So, these errors are often considered as unreliable and discarded altogether, which creates a risk of ignoring a real error. This risk is present in many fields, from intensive care units in hospitals [1], [2] to alarms for aircraft pilots [3] or automobile [4]. It is therefore critical to detect when errors occur at an unexpected rate and to alert when the rate of failure is likely to have evolved.

Moreover, the cost of troubleshooting errors and performing maintenance operations can be very high. The detection system should therefore have a low false positive rate.

The detection of change-points is a well-studied problem, either \textit{a posteriori} [5], [6] or in real-time [7], [8]. However, to the best of our knowledge, no detection model is optimized to explicitly minimize the false positive rate. On the one hand, offline models usually use a fixed number of change points $K$, which is likely to erroneously trigger alerts if there are in reality fewer than $K$ change points. On the other hand, online models often use a fixed window size, which has a low statistical significance, and potentially leads to a high rate of false positives [9]–[11].

Instead, we decide to enable users to set their target false positive rate $\alpha$, and then generate models based on that threshold. This should reduce the risk of ignoring real errors, because users would know that when an alert is triggered, they have high confidence that a change actually occurred.
We illustrate this by defining two models commonly used for change point detection, a Binary Segmentation Model and a Hidden Markov Model, and adapting them to use $\alpha$ as their only parameter.

1.1 Problem statement

We consider a complex system composed of many different physical parts, and equipped with a logging system that triggers an error when an unusual behavior is detected. This logging system stores a sequence of observations $t$, for which events either occur or do not. These observations could be of various time scales, either periodic or based on some cycle. We consider a single event (or error) $e$, independently of the other ones that are recorded by the logging system.

For each observation $t$, the event may or may not be detected. We assume the probability an event occurs $p(t) = \mathbb{P}[y_t = 1]$ to be piece-wise stationary. While this may seem like a strong assumption, it is in fact rather common because it can approximate any situation, and is easily interpretable [12], [13]. While $p$ may be non-stationary, we only focus on significant changes, and we approximate $p$ with a constant between two significant changes. These change points can be interpreted as an external factor that caused $p$ to increase or decrease significantly.

Given a fixed false positive rate, i.e. the probability that a change is detected although no change actually occurred, we define models with the highest sensitivity and the lowest delay possible. The sensitivity is the minimal increase or decrease in the frequency of occurrence that we are able to detect, and the delay is the minimal number of observations after a change in frequency required to detect the change.

This work is aimed at handling the following questions:

- Can we define an event detection model that uses a target false positive rate $\alpha$ as its only parameter?

- What is the sensitivity of detection of changes in the frequency of event occurrence of such a model?

- In other words, given a target level of confidence, what is the minimum change in the frequency that can be identified, and what delay is necessary to reliably identify this change?
1.2 Scope and objectives

The objective of this thesis is to propose and study the performance of computational approaches to the problem of detecting changes in the frequency of occurrence of an event. Given a target level of confidence, these models should trigger an alert when there is sufficiently high confidence that a change in the frequency of occurrence of the event has actually occurred.

The only parameter of our models should be the target level of confidence (for instance $\alpha = 0.01$). Users can set their own threshold, based on the precision-recall rate they need. Given a sequence of events, the models detect changes in the probability of occurrence of an error. This has two main implications:

- Alert when events start occurring more frequently.
- Troubleshoot errors by pinpointing when the probability of occurrence changed.

The scope of the project is limited to one signal, and the data is generated from piece-wise stationary probabilities. We will mention how the models could adapt to a more general case in the Discussion.

1.3 Thesis overview

Chapter 2 introduces the two models we consider, the Binary Segmentation and the Hidden Markov Model, as well as related work in change point detection. Chapter 3 presents the data, the methods used to tune our algorithms and the evaluation methods. In Chapter 4, the results are presented and analyzed. Chapter 5 discusses key findings and limitations. A conclusion is presented in Chapter 6.
Chapter 2

Background

2.1 Related Work

Detection of anomalies or abrupt changes in time series has applications in many fields, such as medicine [1], [15], [16], aerospace [3], automobile [4], climate [17], [18], or finance [16], [19].

Therefore, many methods have been developed to detect change points, either retrospectively or in near real-time. In many cases, the underlying parameters are assumed to be piecewise constant [5]–[8], [20], and the goal is to segment the series into the parts where the signal is homogeneous. This assumption makes the model easy to interpret: each change point corresponds to some external factor which impacted the model. Moreover, the current model is simply the base model with the parameters since the last change point, which makes it simple to predict future values; this also means that events that occurred before that change point can be ignored, which makes online algorithms more efficient.

2.1.1 Retrospective change point detection

For retrospective change point detection algorithms, the input is a full time series, and the goal is to find the number of change points and their location, such that each segment is likely to have been drawn from an homogeneous signal [21]. These algorithms usually assume the number of change points \( K \) is known, and then compare the score of the optimal split for different values of \( K \). The main difference between all these algorithms is their search function to find optimal segmentation and cost.
function to compare scores for different values of $K$.

In theory, there are $\binom{T}{K}$ possible ways to split the interval into $K$ sub-segments, so checking all possibilities is not an option. The most common workaround is to use dynamic programming [22]–[24], which has a complexity of $O(KT^2)$. This model has multiple extensions that make it more efficient in some cases. For instance, the forward dynamic programming [25] makes it more efficient to find the optimal value of $K$. The pruned optimal dynamic programming recently introduced by Rigaill [26] reduces the candidate change points, which decreases the complexity in average. However, both of these extensions have the same worst-case complexity.

Unfortunately, these models can hardly be adapted to real-time detection of change points. Their worst-case complexity is in $O(T^2)$, and the partitioning of $[0, T]$ cannot be simply re-used to make the partitioning of future intervals $[0, T']$ more efficient. Moreover, $K$ is unknown for real-time detection, and these algorithms are not optimized at all to adapt to an unknown number of change points.

### 2.1.2 Real time change point detection

To address these limitations, dynamic programming algorithms have been adapted. To do so, online algorithms search for change points in smaller intervals, rather than searching across the full time series.

There are multiple ways to do so. Using a sliding window of fixed length [9], [10], [19] enables algorithms to have a linear complexity, and therefore to run in real-time. However, the window size has an important impact on the change points detected: it detects either local or global changes, but it can’t detect both kinds of changes. Moreover, the algorithms are used on a subset of the events, which reduces their statistical power [9]–[11]. Finally, these models ignore old observations, so the probability of detecting a change point increases with $T$. Even for a homogeneous time series, there is a high probability of detecting a change point eventually.

Other ways to adapt the dynamic programming to real-time detection is to use a bottom-up or a top-down approach. The bottom-up approach splits the time series into many small segments, and recursively prunes these candidate change points [27]. Once again, the size of the initial segments has an important impact on the change points detected. If these segments are too small, the algorithm can be unstable as the first
segments that are compared are small and therefore have low statistical power. If they are too large, the approximated partitioning is far from the actual one, as the initial segments are the only candidate change points.

The top-down approach is called binary segmentation [28].

### 2.1.3 Binary segmentation

The binary segmentation algorithms searches for a single change point across the full time series. If such a point exists, it recursively searches for other change points on both left and right time series [28]. In our case, the detection of a change point on each segment can be formulated as a null hypothesis that determines whether all events were drawn from the same binomial distribution [29]. If the null hypothesis is rejected, it means there is a change point.

Binary Segmentation is widely used to detect change points, because of its simplicity and efficiency in $O(T \log T)$ [16], [30]. Like the bottom-up algorithm, it only approximates the change points [31], but this time the model does not have any initial segment size, so the approximations are done on larger samples and are therefore theoretically more significant. To improve this, it is even possible to give more weight to segments that are more statistically significant [20]. However, each change point is estimated based on previously selected change points, which can sometimes lead to poor approximations [31].

For online detection of change points, the model can be adapted to be even more efficient. When a new change point is detected, the left segment can be ignored as it was already evaluated by the model in the past.

### 2.1.4 Corrections of the family-wise error rate

One of the main difficulties for real-time detection is the trade-off between false positive rate and detection delay. If the users want to be alerted at most $t$ observations after the change point, using a sliding window of size $2t$ is a good solution, but it does not give more statistical confidence than that of a series of $2t$ observations.

On the contrary, we want users to be able to set their threshold of false positive rate, and in turn accept to be alerted a long time after the change point, if the change is small. Usually, the longer we wait, the
more likely we are to detect a change point. This problem is similar to the family-wise error rate (FWER), which is well known in the medical field.

When testing multiple hypothesis, the probability that at least one of them is true increases with the number of hypothesis. Using a target level of confidence $\alpha$, the probability that at least one of $\kappa$ independent hypothesis is true is in fact $1 - (1 - \alpha)^\kappa$ \[32\]. To counteract this, Bonferroni \[33\] introduced a correction $\alpha' = \frac{\alpha}{\kappa}$ so that the probability of making a Type I error would be $1 - (1 - \alpha')^\kappa \approx \alpha' \kappa = \alpha$, for $\alpha \ll 1$. This correction has many variants \[34\]–\[36\] that make smaller corrections while preserving the same rate of false positives, but all of these still assume that the number of hypotheses $\kappa$ is fixed.

In our case, $\kappa$ increases at each observation. Moreover, the hypothesis for $T$ and $T + 1$ are highly correlated, so the Bonferroni correction is suboptimal. For this reason, we decide instead to use a null hypothesis to tune the threshold use by our models, such that the false positive rate is $\alpha$ for any value of $T$ (see section 3.2).

### 2.1.5 Hidden Markov models

While the previous models are simple to interpret and implement, they cannot be easily take advantage of situations where users have prior knowledge. For instance, when it comes to failure detection in manufacturing, prior knowledge could be that parts have a lifetime that follows a Weibull distribution \[37\], \[38\], and maintenance operations are executed once a month, or shortly after critical parts fail. To account for these kind of situations, the most adapted solutions are Bayesian methods. We study one of the most commonly used: hidden Markov models \[39\]–\[41\].

There are two main challenges to use hidden Markov models in our case. First, these models have many parameters, so we have to make assumptions on the prior distribution to reduce our model to the single parameter $\alpha$ (See section 3.4). Second, it may be harder to adapt them to online data. Chis and Harrison \[42\] suggest a solution to adapt the model by estimating its updated parameters. This could be used to avoid recomputing the hidden state with new values.
2.2 Background on Binary Segmentation Model

The first model we consider to estimate the underlying probabilities is based on binary segmentation. Given a sequence of events, we use a null hypothesis to determine whether a change point occurred or not. If it was the case, we recursively search for other change points in left and right sub-intervals.

Given a sub-interval from instant \( t_i \) to \( t_{j-1} \), for each \( k \in [i, j - 1] \), we define the null hypothesis:

\[
H_{0^{i,k}}: \text{The events on intervals } [i, k - 1] \text{ and } [k, j - 1] \text{ were drawn from the same Bernoulli distribution}
\]

To determine how likely this null hypothesis is to be true, we compute the probability that a binomial distribution of probability \( p_{ij} \) triggered the given number of events on both sub-intervals. The z-value associated with this null hypothesis is:

\[
z_{i,j,k} = \frac{|p_{ik} - p_{kj}|}{p_{ij}(1 - p_{ij}) \left( \frac{1}{k - 1} + \frac{1}{j - k} \right)}
\]

where \( p_{i1i2} = \frac{y_{1-i_{1}..i_{2}}}{t_{2} - t_{1} - 1} \) denotes the observed frequency of occurrence on interval \([i_1, i_2 - 1]\).

The null hypothesis is rejected if \( z_{i,j,k} \) is larger than some value \( z \). If the null hypothesis \( H_{0^{0,T,k}} \) is rejected for at least one \( k \), we consider the most significant change point \( k^* = \arg\max \ z_{0,T,k} \), and search for other change points on the intervals \([0, k^* - 1]\) and \([k^*, T - 1]\).

The recursion terminates when the null hypothesis is not rejected for any elementary interval. For each interval, we simply define the predicted probability of event occurrence as the average number of events that occurred.

In section 3.3, we find the minimal value of \( z \) such that \( z_{0,T,k} < z \) for all \( k \), given the level of confidence \( \alpha \), number of occurrence \( T \), and the frequency of occurrence \( p \).
2.3 Background on Hidden Markov Model

The second model is a hidden Markov model, where the hidden states correspond to the underlying probabilities.

We note $x_t$ the hidden state at $t$ and $y_t$ the corresponding observation. The parameters of a hidden Markov model are:

- The number of states $N$. In our case states correspond to probabilities $p_0, \ldots, p_{N-1}$
- The number of observations $T$, observations are noted $y_0, \ldots, y_{T-1}$
- The initial hidden states $\varphi_i = P[x_0 = p_i]$
- The transition probabilities $\phi_{i,j} = P[x_{t+1} = p_j | x_t = p_i]$
- The distribution from which observations are drawn. Here, observations are drawn from a Bernoulli distribution: $y_t \sim B(1, x_t)$

Given $N$, there are $O(N^2)$ independent parameters $\varphi_i$ and $\phi_{i,j}$. We make a few additional assumptions to reduce the complexity of the model.

Uniformly distributed states

We assume the states are uniformly distributed on the interval $[0, 1]$:

$$\forall i \in [0, N - 1], p_i = \frac{i}{N - 1}$$

This is not a loss in generality, as we can increase the number of states $N$ to cover any possible probability $p \in [0, 1]$

Uniform prior distribution

No particular prior distribution is assumed, so the initial states are sampled from a discrete uniform distribution over all possible states:

$$\forall i \in [0, N - 1], \varphi_i = \frac{1}{N}$$

Uniform change amplitude

We also assume that when a change occurs, all new states are equiprobable, in other words that, given $i$, $\phi_{i,j}$ is constant for any $j \neq i$. 
Uniform change likelihood

We also assume that the probability a change occurs is independent of the current state, in other words that $\phi_{i,i}$ is constant. In situations like the one described above (See section 2.1.5), the assumption could be adapted to account for the user’s prior knowledge.

With these two assumptions on $\phi$, the matrix of transition probabilities now has two values, one for diagonal coefficients $\phi_{i,i}$ and one for extra-diagonal coefficients $\phi_{i,j}$. We note $\rho$ the ratio between these two coefficients:

$$\rho = \frac{\phi_{0,1}}{\phi_{0,0}}$$

As for each row, the sum of its coefficients is 1, we can derive the value of all the transition probabilities from $\rho$:

$$\phi_{i,j} = \begin{cases} 
1 & \text{if } i = j \\
\frac{1}{1 + (N - 1)\rho} & \text{if } i \neq j 
\end{cases}$$

With these assumptions, we have reduced the complexity of our model to a single parameter $\rho$. In section 3.4 we find the optimal value of $\rho$ given the number of states $N$, the number of observations $T$, the probability of occurrence $p$ and the target level of confidence $\alpha$. 
Chapter 3

Methods

First, we find the tuning of the parameters for both candidate models that provide the target confidence level. Therefore, we use the mathematical definition to estimate the how the parameters should be tuned, and we approximate the actual parameters using Monte Carlo methods.

Then, we compare the sensitivity and delay of the models. We generate data with a single change point, and measure the number of observations required to detect the change, depending on the amplitude of the change.

Finally, we study how well both models could be generalized to real data. To do so, we generate sequences of probabilities of occurrence where changes occur with either low and high frequency, and derive events based on those probabilities. Then, we estimate the underlying probabilities based on the events, and compare the candidate models to find the one that performs best.

3.1 Data

All the data used in this report are generated, and Monte Carlo methods are used to tune and evaluate the event detection models.

To generate the data, we first define a fixed piece-wise stationary sequence of probabilities. Then, at each instant \( t \), we trigger an event with probability \( p(t) \). For the Hidden Markov Model, we also fix the number of hidden states \( N \). Finally, we set a target value \( \alpha \), which should be the false positive rate of our models.

Given these parameters, we tune the other parameter of the event detection model (respectively the z-value for the Binary Segmentation
Model and the probability of changing state $\rho$ for the Hidden Markov Model), such that it has a false positive rate $\alpha$. To do so, we use Monte Carlo methods to account for the different sequences of events that could be generated from the same underlying sequence of probabilities $p(t)$.

Table 3.1 sums up the different parameters of variables we use and their notation.

<table>
<thead>
<tr>
<th>Description</th>
<th>Name</th>
<th>Kind</th>
</tr>
</thead>
<tbody>
<tr>
<td>Probability of occurrence of an event</td>
<td>$p(t), p_1, p_2$</td>
<td>Fixed</td>
</tr>
<tr>
<td>Target false positive rate</td>
<td>$\alpha$</td>
<td>Fixed</td>
</tr>
<tr>
<td>Number of hidden state (Hidden Markov Model)</td>
<td>$N$</td>
<td>Fixed</td>
</tr>
<tr>
<td>Z-value (Binary Segmentation Model)</td>
<td>$z$</td>
<td>Derived</td>
</tr>
<tr>
<td>Probability of changing state (HMM)</td>
<td>$\rho$</td>
<td>Derived</td>
</tr>
<tr>
<td>Sensitivity</td>
<td>$\delta_p$</td>
<td>Output</td>
</tr>
<tr>
<td>Delay</td>
<td>$\delta_t$</td>
<td>Output</td>
</tr>
</tbody>
</table>

Table 3.1: Parameters and variables. Fixed parameters are set initially, derived parameters are estimated in sections 3.2 and 3.4, and output variables are computed in sections 4.1 and 4.2.

### 3.2 Using a null hypothesis to tune our models

To tune our models, we first set a constant probability $p(t) = p_1$. We then generate many sequences with this probability, the $100(1 - \alpha)th$ percentile corresponds to the threshold value for which the model has a false positive rate $\alpha$.

As mentioned previously, we want to define event detection models with a single parameter, the target level of confidence $\alpha$. We tune the parameters of our models such that the probability that we erroneously detect a change point is at most $\alpha$. Given a sequence of events $y_0, \ldots, y_{T-1}$ with mean $\bar{y}$, we consider the null hypothesis:

$$H_0: \text{ All events } y_0, \ldots, y_{T-1} \text{ were drawn from the same Bernoulli distribution } B(1, \bar{y})$$

If the models are correctly tuned, we should reject this null hypothesis with a probability of $\alpha$ for events drawn from a distribution with constant
parameters. Therefore, when our models detects a change point, we can assert that the underlying probability of occurrence has changed with a confidence of $1 - \alpha$.

### 3.3 Tuning the Binary Segmentation Model

For any $H_0^{i,j,k}$, the theoretical value of $z$ is $z_{\alpha/2}$. However, the null hypothesis we are testing is

$$H_0 \iff \forall k \in [0, T - 1], H_0^{0,T,k}$$

The more hypothesis are tested, the more likely it is that one of them is rejected \[32\], \[33\]. In fact, large values of $T$, at least one hypothesis is rejected almost surely. Therefore, we need to adapt the threshold value $z$. The Bonferroni correction is suboptimal in our case, because the null hypothesis $H_0^{0,T,k}$ are highly correlated. We therefore compute a correction for our specific problem.

We define a function that approximates the threshold value $z$. The actual formula of the function is not really relevant, but being able to compute $z$ from any sequence is more efficient than recomputing the approximate value using Monte-Carlo methods. It is also interesting to understand the impact the different parameters have on the threshold value $z$.

We assume this function can be decomposed into a product of elementary functions:

$$z = f_T(T)f_p(p)f_\alpha(\alpha)$$

and we therefore estimate separately the shapes $f_T$, $f_p$ and $f_\alpha$ with all other parameters fixed as a first step, and then derive a full formula.
Influence of $T$ on $z$

Figure 3.1: Influence of $T$ on $z$: minimum $z$-value threshold for which the model does not detect a change point, for different levels of confidence $\alpha$. The dashed lines represent linear functions of $T$ and $\sqrt{T}$.

Parameters: $p = 20\%$, $n_{\text{tries}} = 1000$

If all hypothesis $H_{0,T,k}^0$ were independent, the probability that at least one hypothesis would be rejected would be $\alpha' = 1 - (1 - \alpha)^T$. In our case, while the hypothesis are not independent, we still expect $z$ and $T$ to be positively correlated.

Figure 3.1 shows this correlation. Given $p$ and $\alpha$, we decided to fit a linear regression on $T$ and $\sqrt{T}$. The dashed lines on the figure represent such functions, which are acceptable approximations of the shape of the actual value of $z$.

Influence of $p$ on $z$

We expect the probability $p$ to have an impact on the threshold $z$-value. Indeed, for low probabilities, it is unlikely to have multiple events, so the different $H_{0,T,k}^0$ are less likely to be independent from one another. Note that there is a perfect symmetry around 0.5 in our model, as replacing $p$ by $1 - p$ is equivalent to considering observations without events instead
of events. Given the shape of the curves on Figure 3.2, we approximate this as a function of \(|0.5 - p|^6\).

Figure 3.2: Influence of \(p\) on \(z\): minimum-value threshold for which the model does not detect a change point, for different levels of confidence \(\alpha\).

Parameters: \(T = 100, n_{\text{tries}} = 1000\)

**Influence of \(\alpha\) on \(\frac{z}{z_{\alpha/2}}\)**

As mentioned previously, for \(H_0^{0,T,k}\), the theoretical value for \(z\) is \(z_{\alpha/2}\), which is the value such that \(\mathbb{P}[|\mathcal{N}(0, 1)| < z_{\alpha/2}] = \alpha\). The question here is whether \(\frac{z}{z_{\alpha/2}}\) is independent of \(\alpha\).

If the events were independent, the value of \(z\) would be \(z_{\alpha'} > z_\alpha\), so we expect \(\frac{z}{z_{\alpha/2}}\) to be an increasing function of \(\alpha\). Figure 3.3 confirms this hypothesis, and the correlation is linear. The dashed lines represent linear functions of \(\alpha\), and are acceptable approximations.

**Full approximation formula \(z(T, p, \alpha)\)**

Based on the previous studies, \(\frac{z}{z_{\alpha/2}}\) can be approximated with a linear function of \(T, \sqrt{T}, (0.5 - p)^6\) and \(\alpha\), or any combination of these. Using a linear regression, we evaluated the constants, and noticed
Figure 3.3: Influence of $\alpha$ on $\frac{z}{z_{\alpha/2}}$ : threshold ratio $\frac{z}{z_{\alpha/2}}$ for which the model does not detect a change point, for different numbers of observations $T$. The dashed lines represent linear functions of $\alpha$.

Parameters: $p = 20\%$, $n_{\text{tries}} = 1000$

that considering combinations of these parameters did not significantly improve the score of the regression. Here is the full expression to approximate the value of $z$:

$$z \approx z_{\alpha/2} \times \left[ c_0 + c_T T + c_{\sqrt{T}} \sqrt{T} + c_{\alpha} \alpha + c_p (0.5 - p)^6 \right]$$

where $c_0 = 1.1$, $c_T = -3.8 \times 10^{-4}$, $c_{\sqrt{T}} = 2.3 \times 10^{-2}$, $c_{\alpha} = 2.6$, and $c_p = 27$.

A more precise function could be obtained by increasing the number of tries and by increasing the set of parameters used for the Monte Carlo methods.
3.4 Tuning the Hidden Markov Model

The Viterbi algorithm \[40\] provides the most likely set of underlying probabilities, given the parameters of the model and a sequence of observations. It is composed of two phases. In the forward procedure, it computes the probability of seeing the sequence of observations, given an underlying state. In the backtracking procedure, it computes the probability of seeing the end of the sequence of observations.

We use this algorithm to detect change points. In this section, we estimate the threshold value of $\rho$ for which no change point is detected, i.e., where the most probable underlying state sequence contains a single state.

Once again, we assume this function can be decomposed into a product of elementary functions:

$$
\rho = f_N(N)f_T(T)f_p(p)f_\alpha(\alpha)
$$

and therefore estimate separately $f_N$, $f_T$, $f_p$ and $f_\alpha$ with all other parameters fixed as a first step, and then derive a full formula.

Influence of $\rho$ on the detection of change points

Our Hidden Markov Model has been reduced to a single parameter $\rho$. A high value of $\rho = \frac{\partial \Omega_i^j}{\partial \Omega_i}$ corresponds to a high relative probability of changing state. Our goal is to find the minimum value of $\rho$ for which the Hidden Markov Model detects a change point with probability at most $\alpha$.

As for the previous model, we find an estimation of $\rho(N,T,p,\alpha)$ to make future estimations more efficient.

Influence of $N$ on $\rho$

To determine the most likely sequence of underlying states, the Viterbi algorithm does a forward pass during which it computes the most likely state up to the current observation, and then a backtracking pass where it retrieves the most likely state to have lead to the final state.

This can be seen as a trade-off between two opposite "energies": changing the underlying state from $i$ to $j$ costs $\mathcal{E}_{i\to j} \sim \frac{\partial \Omega_i^j}{\partial \Omega_i}$, while staying in the current state $i$ costs $\mathcal{E}_{i\to i} \sim \mathbb{P}_k$ where $\mathbb{P}_k$ represents the probability of predicting the wrong state ($1 - p_k$ if the event occurred, $p_k$ otherwise).
The Viterbi algorithm finds the sequence of states that minimizes this energy: it is only worth changing state if many future observations are unlikely to be derived from the current state $i$.

Using our assumptions in section 2.3, these energies become $E_{i\rightarrow j} \sim \rho$ and $E_{i\rightarrow j} \sim \frac{E_{i}}{p_{j}}$. Note that these energies are now independent of $N$: the only impact is that, for small values of $N$, all possible probabilities $p_{i}$ and $p_{j}$ is not represented. In other words, there might not be the state corresponding to the actual underlying probability. However, for sufficient large value of $N$, $\rho$ should be independent of $N$.

Figure 3.4 confirms this hypothesis. It shows an example using Monte Carlo method with $n_{\text{tries}} = 1000$ tries, where we approximate the optimal $\rho$ with a precision of 0.001. In the rest of this report, we use $N = 101$ states, so that the states represent 0%, 1%, 2%, ..., 100%.

![Figure 3.4: Independence of $N$ of $\rho$: maximum transition probability ratio $\rho$ that does not detect a change point, for different levels of confidence $\alpha$](image)

Parameters: $p = 20\%$, $T = 100$, $n_{\text{tries}} = 1000$, precision = 0.001
Influence of \( T \) on \( \rho \)

At first look, we expect \( \rho \) to be inversely proportional to the maximum number of observations \( T \). Indeed, the expectation of the delay between two consecutive change points is of the order of \( \frac{1}{\rho} \). Therefore, for \( T \) observations, the maximum value of \( \rho \) for which the model does not detect any change should be of the order of \( \frac{1}{T} \).

However, similarly to the Binary segmentation model, for large values of \( T \), there are many candidate change points \( k \in [0, T] \), which increases the likelihood of detecting a change point. As shown on Figure 3.5, the correlation between \( \rho \) and \( T \) is well approximated by \( \rho \propto \frac{1}{\sqrt{T}} \).

![Figure 3.5: Influence of \( T \) on \( \rho \): maximum transition probability ratio \( \rho \) that does not detect a change point, for different levels of confidence \( \alpha \). The dashed line corresponds to our theory: \( \rho = \frac{c_T}{\sqrt{T}} \) for some constant \( c_T \).

Parameters: \( N = 101 \), \( p = 20\% \), \( n_{\text{tries}} = 1000 \), \( \text{precision} = 0.001 \)
Influence of $p$ on $\rho$

As for the previous model, there is a symmetry on $p$ with regard to 0.5, as replacing $p$ by $1 - p$ is equivalent to considering observations where no event occurred. Figure 3.6 shows that for large values of $p$, the threshold is high for extreme values of $p$. Therefore, we approximate this dependency as $\rho \propto \frac{1}{\min(p, 1-p)}$.

![Figure 3.6: Influence of $p$ on $\rho$: maximum transition probability ratio $\rho$ that does not detect a change point, for different levels of confidence $\alpha$. Parameters: $N = 101$, $T = 100$, $n_{\text{tries}} = 1000$, precision = 0.001](image)

Influence of $\alpha$ on $\rho$

We expect $\rho$ to be an increasing function of the level of confidence $\alpha$: if we loosen our restrictions and accept a higher risk of erroneously detecting a change, then we can increase the transition probability ratio. When plotting some examples (see Figure 3.7), it is clear that there is a linear correlation between $\alpha$ and $\rho$. Note that for larger values of $\alpha$ ($\alpha > 0.8$), the correlation isn’t linear anymore, but we ignore this case as we only focus on high confidence.
Figure 3.7: Influence of $\alpha$ on $\rho$: maximum transition probability ratio $\rho$ that does not detect a change point, for different probabilities of occurrence $p$

Parameters: $N = 101$, $T = 100$, $n_{tries} = 1000$, precision = 0.001

Full approximation formula $\rho(N, T, p, \alpha)$

Based on the previous studies, $\rho$ can be approximated with a linear function of $\frac{1}{\sqrt{T}}$, $\frac{1}{\min(p, 1-p)}$ and $\alpha$, or any combination of these. Using a linear regression, we evaluated the constants. Here is the full expression to approximate the value of $\rho$:

$$\rho(N, T, p, \alpha) \approx \frac{1}{\sqrt{T}} \times \left[ c_0 + c_{\alpha} \alpha + \frac{c_p}{\min(p, 1-p)} \right]$$

where $c_0 = 7.1 \times 10^{-3}$, $c_{\alpha} = 1.4$ and $c_p = 1.5 \times 10^{-3}$. 
Chapter 4

Results

We have now defined two models that can, given a target level of confidence $\alpha$, detect changes that occur with probability at least $1 - \alpha$. In this section, we compare the sensitivity and the delay of both models. To do so, we generate a sequence of events with a single change point at $t_1$, where event frequency increases from $p_1$ to $p_2$:

$$y_i \sim \begin{cases} 
Ber(1, p_1) & \text{if } i < t_1 \\
Ber(1, p_2) & \text{if } i \geq t_1 
\end{cases}$$

The sensitivity of the detection models is the minimal value of $|p_1 - p_2|$ such that a change is detected for some $t_2 > t_1$. The delay is $t_2 - t_1$. 
4.1 Sensitivity

Figure 4.1 shows the sensitivity of both models. They are similar: changes are detected for $p_2 < 10\%$ or $p_2 > 35\%$, the sensitivity is of the order of $\delta_p = 15\%$.

![Graph showing sensitivity for different times $t_2$.](image)

(a) Binary Segmentation Model  
(b) Hidden Markov Model

Figure 4.1: Comparison of the sensitivity of both models at different times $t_2$.

Parameters: $N = 101$, $t_1 = 100$, $n_{\text{tries}} = 1000$, $p_1 = 20\%$, $\alpha = 0.01$

This sensitivity is highly impacted by the initial frequency of occurrence $p_1$. When events are rare (Figure 4.2a, 4.2b), then only a few consecutive events enable us to detect a change point. For $p_1 = 10\%$, the sensitivity is of the order of $\delta_p = 10\%$.

On the contrary, for events that occur with $p_1 = 50\%$ probability (Figure 4.2c, 4.2d), it is hard to determine when the probability has changed: the models are less sensitive to changes. For $p_1 = 50\%$, the sensitivity is of the order of $\delta_p = 20\%$. 
Figure 4.2: Sensitivity of both models at different times $t_2$, for $p_1 = 10\%$ and $p_1 = 50\%$.

Parameters: $N = 101$, $t_1 = 100$, $n_{\text{tries}} = 1000$, $p_1 = 10\%$, $\alpha = 0.05$
4.2 Delay

Significant changes (e.g. 20% → 60%) can be detected with high probability only 10 observations after the change occurred (Figure 4.3). Smaller changes (20% → 40%) are detected after 300 iterations. Changes under $\delta_p = 10\%$ are not often detected, even after 1000 observations.

Both models have a similar delay, of the order of $\delta_t = 100$ observations. The impact of the initial frequency of occurrence $p_1$ on that value is insignificant.

Figure 4.3 represents, for different values of $p_2$, the probability that a change is detected after $t$ observations. When $|p_2 - p_1| < \delta_p$, the change is detected at most $\delta_t = 100$ observations after the change actually occurred. This value is the same for different values of $t_1$, and for both models.

Figure 4.3: Delay for $p_2 = 20\%$.
Parameters: $n_{\text{tries}} = 1000$, $p_1 = 20\%$, $\alpha = 0.01$. The model used here is the Binary Segmentation Model, the Hidden Markov Model has similar results.
4.3 Comparison to a standard model for event detection

To make our results more tangible, we compare them to a standard model. We decided to use a model based on a sliding window, which is commonly used because of its simplicity and ease of implementation [21]. Given a window size $w$, at every observation, we measure the discrepancy between the preceding and the following sequences of length $\frac{w}{2}$. A change point is detected if the discrepancy is larger than some threshold.

We define a Sliding Window Model, which has not been tuned to have a false positive rate of $\alpha$. We note $w$ the window size used for this model.

Like the Binary Segmentation Model, it uses statistical testing to determine whether a change point has occurred. However, for each observation, it performs a statistical test on the middle of the last interval of length $w$. The test is positive if the $z$-value corresponding to the intervals $[t - w, t - \frac{w}{2}]$ and $[t - \frac{w}{2}, t]$ is larger than $z_\alpha = F^{-1}(1 - \frac{\alpha}{2})$, where $F$ is the cumulative distribution function of the normal distribution.
4.3.1 False positive rate of the Sliding Window Model

While our models were tuned to have a fixed false positive rate $\alpha$, the Sliding Window Model has a larger false positive rate, which depends on the window size $w$ and the number of observations $t$.

The false positive rate increases with the number of observations. This is due to the fact that the statistical test is performed multiple times over a fixed window size $w$. In particular, these tests are independent from one another every $w$ iteration, so the risk of erroneously detecting a change point increases with $t$. It decreases with the window size, because a large window is more statistically significant. The drawback is that changes that occur before $t = w$ aren’t detected.

![Figure 4.4: False positive rate of the Sliding Window Model. The dashed line corresponds to the false positive rate of the Binary Segmentation Parameters: $p_1 = 20\%$, $\alpha = 0.01$](image-url)
4.3.2 Comparison to the sensitivity of the Sliding Window Model

Figure 4.5 shows the sensitivity of the Sliding Window Model, for the same parameters as the ones used for Figure 4.1. The window size is set to respectively $w = 50$ and $w = 100$.

This time the results are different from the ones for both other models. The high false positive rate is visible here: when $p_2 = p_1 = 0.2$, a change point is detected with over 90% probability after 1000 iterations.

As the Sliding Window Model only checks the last $w$ observations, the plots for $t_2 > w$ are irrelevant, as change point detected after $t_1 + w$ do not correspond to the actual change point. For $w = 100$, the sensitivity is of the order of $\delta_p = 0.3$.

Figure 4.5: Sensitivity of the Sliding Window Model
Parameters: $t_1 = 100$, $n_{\text{tries}} = 1000$, $p_1 = 20\%$, $\alpha = 0.01$
4.3.3 Comparison to the delay of the Sliding Window Model

Figure 4.6 shows the delay of the Sliding Window Model, for the same parameters as the ones used for Figure 4.3. The window size is set to respectively $w = 50$ and $w = 100$. The values are not defined for $t < w$ on these plots, and set to 0.

It is clear here that the risk of triggering an alert increases with the number of observations. Only the $w$ observations after the change point can correspond to the true change, but the probability that a change is detected increases fairly quickly. By definition, the delay of these models is at most $w$, but the increase in change point detection that occurs right after the change point occurred is unclear given the noise due to false alerts.

Figure 4.6: Delay of the Sliding Window Model
Parameters: $t_1 = 100$, $n_{tries} = 1000$, $p_1 = 20\%$, $\alpha = 0.01$
4.4 Generalization to multiple change points

In chapter 4, we studied the sensitivity of the models, which correspond to the detection to a single change point. To confirm our models behave as expected, we now apply them to sequences of events with multiple change points. This does not prove their accuracy or their capacity to generalize to any data, as the sequences are only specific examples. However, it gives us a better understanding of the behavior of the algorithms, the kind of prediction it makes, and can help determine a good level of confidence $\alpha$ for these models.

We generate the data based on probabilities that are increasingly closer. As we only generate an example, the actual frequency of occurrence might be different from the original probability the events were derived from.

Low frequency changes

In this section, we generate events from a sequence of probabilities that do not change too often. More specifically, we define series with a change point every $\delta_t = 500$ observation. Given the delay of our models is of the order of 100 observations, the $\alpha$-tuned models should have enough time to detect changes, as long as their amplitude is larger than the sensitivity of the models $\delta_p \approx 0.1$.

Figure 4.7 shows an example of the behavior of both $\alpha$-tuned models. As long as the changes are larger then the sensitivity $\delta_p = 0.1$ (up to $t = 2500$ in our example, all changes are significant), both models correctly estimate the probabilities, independently of the value of their threshold $\alpha$. Beyond that value, the models do not have sufficient confidence to detect change points.

For $\alpha = 0.001$, the Sliding Window Model behaves in a similar way. However, for larger values of $\alpha$, it is unstable, because the window size $w = 100$ is too small compared to the frequency of changes.
(a) True sequence of probabilities

(b) Sliding Window Model ($w = 100$)

(c) Binary Segmentation Model

(d) Hidden Markov Model

Figure 4.7: Sequence of underlying probabilities predicted by the models on data with low frequency of changes.

Parameters: Frequency of change points: 50, $\alpha = 0.01$

**High frequency changes**

In this section, we generate events from a sequence of probabilities that changes frequently.

We define series with a change point every 50 observation. Given the delay of our $\alpha$-tuned models is of the order of $\delta_t = 100$ observations, they should have not have enough time to detect changes, and may not correctly estimate the underlying probabilities.

Indeed, Figure 4.8 shows that the predictions only detect significant changes $\delta_p > 0.6$. Indeed, for $t > 200$, the changes are too small, and the models predict a constant value. For larger value of $\alpha$, the models are too
(a) True sequence of probabilities

(b) Sliding Window Model \((w = 100)\)

(c) Binary Segmentation Model

(d) Hidden Markov Model

Figure 4.8: Sequence of underlying probabilities predicted by the models on data with high frequency of changes.

Parameters: Frequency of change points: 50, \(\alpha = 0.01\)

unstable and trigger false positives. For small values of \(\alpha\), they correctly detected the first 2 changes, and ignore all following ones. This is exactly the expected behavior: quickly detect large changes, but accept to miss small changes in order to avoid triggering false alerts.

In this situation, the window size is perfectly adapted to the frequency of change, the the Sliding Window Model predicts the underlying probability well. While it performs better in this situation, the frequency of change is usually not known, and varies through time. While this model may predict the underlying sequence of probabilities better (in terms of \(L_2\) distance for instance), it also triggers false alerts more frequently.
Chapter 5

Discussion

5.1 Key Findings

Models can be effectively optimized to produce low false positive rates in detecting rare events. Compared to the standard model, the optimized models are more stable across different time scales. In general, event detection models robustly detect either local or global change points, but they do not generalize well enough to detect both types of events at the same time. As our models have a low false positive rate, they do not always perform equally well for high-frequency changes (in terms of $L_2$ distance from the actual probabilities distribution). Indeed, the tuned models ignore changes that are not statistically significant, but guarantee that the detected change points are particularly likely to have occurred.

5.2 Complementarity of the event detection models

The two models we defined produce similar results: their sensitivity is of $\delta_p \approx 10\%$ and delay is $\delta_t \approx 100$ observations. We believe this is the case because the uncertainty mostly comes from the data itself and the distribution it is drawn from.

But both models are designed differently, and therefore offer complementary solutions. Depending on the context, one model may be more adapted than the other. Here are a few differences between both models:
5.2.1 Complexity of the models

For the Binary Segmentation Model, at each step $k$, we store the number of events up to $k t_k$, each of the $T$ null hypothesis $H_0^{0,T,k}$ can be evaluated in $O(1)$ time and space. The total complexity to detect a change point can therefore be computed in $O(T)$ time and $O(1)$ space.

If a change point is detected, the same algorithm is applied recursively to both sub-segments. Therefore, the worse-case complexity to evaluate a full sequence of occurrences of length $T$ is $O(T \log T)$.

For the Hidden Markov Model, the complexity is $O(T N^2)$. In terms of space, this model uses $O(N^2)$ to store the transition probability matrix.

Both complexities are similar, the Hidden Markov Model should only perform better for very large values of $T$.

5.2.2 Application to online data

For the Binary Segmentation Model, most of the computation can be avoided, as $z_{0,T,k}$ is a good approximation of $z_{0,T+1,k}$. If no change point is detected at $T$, only a extreme values of $z_{0,T,k}$ are candidates for a detection at $T + 1$. Moreover, change points on the first sub-intervals do not change at $T + 1$. Therefore, the model can simply be updated to compute in $O(T)$, or even $O(1)$ if no change point is detected.

For the Hidden Markov Model the Viterbi algorithm normally returns a full path of states. However, there are ways to derive an online algorithm [43], by returning the Viterbi path on sub-segments rather than the full sequence. It is possible to prune out segments that cannot be part of the most likely path, and therefore reduce the complexity of the algorithm without a significant drop of the accuracy.

5.2.3 Generalization of the models

As mentioned in the background section 2.1.5, the Hidden Markov Model can take advantage of situations where users have prior knowledge. The transition probability can be updated to reflect better the actual probabilities. In those cases, we expect this model to perform better.

The Binary Segmentation Model is not as easily configurable. However, in addition to triggering alerts when a change point is detected,
it provides extra information that is useful to our problem: it computes
the approximated probability that a change occurred at any time \( t \in [0, T] \).

This means that we could change the threshold level of confidence on-
the-fly and immediately know whether a change point occurred or not
with lower confidence. We could also determine the time range where
the change point could have occurred, rather than a single instant. If
the change is significant, we know precisely when the change occurred.
If the change has a small amplitude, we know a change occurred but
have a much lower confidence as to the exact time when the change has
occurred.

5.3 Original contribution

This thesis has shown that event detection models can be tuned to have
a low false positive. We used models which are well-studied, namely
binary segmentation \([16], [30]\) and Hidden Markov models \([39]–[41]\), but
we tuned them to optimize their performance based on a different criteria,
the false positive rate. Independently of the significance of the change and
the delay between consecutive change points, these models only trigger
an alert if there is sufficient confidence that a change in frequency of
events occurred.

Compared to sliding window models \([9], [10], [19]\), our event detection
models require time to be tuned and are less computationally efficient
because they consider all past events since the last change point. However, they adapt better to situations where changes occur with
varying frequency, because the only parameter they depend on is the
false positive rate and do not suffer any loss in statistical power \([9]–[11]\).

Both \( \alpha \)-tuned models have similar sensitivity and delay, of the order
of \( \delta_p \approx 10\% \), and the delay of detection is \( \delta_t \approx 100 \) observations. The
results are close for both \( \alpha \)-tuned models, and the Sliding Window Model
also has a similar sensitivity, which seems to indicate that the uncertainty
mostly comes from the data itself and the distribution it is drawn from.
However, for a similar sensitivity, the \( \alpha \)-tuned models have a much lower
false positive rate.
5.4 Limitations

In this thesis, we restricted our work to the study of individual events. Complex systems have multiple interactions, and therefore failures are not independent from one another. It would be interesting to generalize our models to multiple failures, to find the ones that are related. This would provide extra information to troubleshoot the failure, and could boost our models’ sensitivity and delay.

We also restricted our work to binary events. These models could be adapted to work on any real value, and detect potential changes in sensor data. We also assumed the underlying probability of occurrence of events to be piecewise stationary. While we explained in section 1.1 that we could detect significant changes in any situation, the models we defined may be less sensitive to detect a progressive increase of the frequency.

A main limitation to our models is also the time required to generate them. To ensure a fixed false positive rate, we decided to use Monte Carlo methods. We used \( n_{\text{tries}} = 1000 \) tries for each input parameter, because this value seemed sufficient to have stable results, but the precision could be improved by increasing the number of tries.

Moreover, we made strong assumptions about of independence of the parameters \( z \) and \( \rho \) depend on, when decomposing them in sections 3.3 and 3.4. We could avoid this assumption by estimating the values of \( z \) and \( \rho \) for any combination of their input parameters, which would take even longer.

Finally, the tuning of a model is only valid for some value of \( \alpha \). If users decide to increase or decrease the false positive rate, all other parameters have to be computed again.

5.5 Ethics and sustainability

The main ethical risk with our method is that it incentivizes users to ignore some abnormal behaviors detected by the system. If one of these errors were to be non-spurious, the model could be held for responsible. Ideally, of course, alerting systems should only trigger real alerts, but this is not always possible in complex systems, and logging systems always
bias towards triggering an alert.

Another risk is that our models justify flaky alerts, which is not the intended goal. Our models are simply temporary solutions until alerts are corrected and become more reliable.

In aeronautics, safety is the main concern. Therefore, whenever an alert is triggered, operators may replace multiple parts until the alert disappears. However, if the alert was due to a bad contact, many parts may end up being replaced unnecessarily. Using one of our models, the number of maintenance operations can be reduced, which is more sustainable.
Chapter 6

Conclusion

We successfully defined two different models to detect change points, with the false positive rate $\alpha$ as their only parameter. The results show that, for a false positive rate $\alpha = 0.01$, the sensitivity of the detection of change in frequency of occurrence is $\delta_p \approx 10\%$, and the delay of detection is $\delta_t \approx 100$ observations.

Both models give similar results, which indicates that when changes in frequency are not detected, it is seemingly due to the logic used to generate the data: false negatives correspond to changes in frequency that are not statistically significant. However, both models were designed differently, they could therefore have different results in some specific situations. For online data, the Binary Segmentation Model would likely be more efficient. For situations where there is some prior knowledge about the data, the Hidden Markov Model could use this extra information as an input and therefore be able to perform better.

This work has multiple applications. In complex systems, it is too time-consuming to monitor all the errors that occur. In that case, unusual errors are carefully troubleshooted, but intermittent and stochastic errors can end up being ignored. In that case, being able to flag when an error occurs more frequently than usual can be vital.

Furthermore, a significant increase in the frequency of occurrence of an error is likely to be due to some malfunction of the system. Indicating precisely when the change in frequency occurred is also useful to troubleshoot it. Using historical data, it is also possible to understand what actions led to a decrease in frequency of occurrence in the past, to take efficient actions when the error occurs again.
Finally, having a low false positive rate makes even expensive maintenance operations cost-worthy, because the alert is reliable enough. Thanks to these models, users who monitor large complex systems, can spend their time troubleshooting actual unexpected failures rather than spurious ones without risking to ignore important failures. The models also enable them to quickly determine when the error occurred in order to find the root cause of the increased frequency of occurrence.

6.1 Future Work

We limited our study to two models that are commonly used to detect change points, but that have different implementations. Using our method to tune models with Monte Carlo methods, it would be interesting to adapt other models to use $\alpha$ as their single parameter.

We mentioned in the discussion (see 5.2.3) that the Hidden Markov Model could generalize better to situations where we had prior knowledge. We did not confirm this hypothesis, so it would be interesting to generate an underlying sequence of probabilities for which the changes follow some distribution. We could than determine whether the prior knowledge about that distribution enables the Hidden Markov Model to perform better than the Binary Segmentation Model.
Bibliography


