CHANGE POINT DETECTION IN END-TO-END MEASUREMENTS TIME SERIES

Diego Ximenes Mendes

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientador: Edmundo Albuquerque de Souza e Silva

Rio de Janeiro
Janeiro de 2017

CHANGE POINT DETECTION IN END-TO-END MEASUREMENTS TIME SERIES

Diego Ximenes Mendes

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

RIO DE JANEIRO, RJ – BRASIL
JANEIRO DE 2017

Ximenes Mendes, Diego

Change Point Detection in End-to-End Measurements Time Series/Diego Ximenes Mendes. – Rio de Janeiro: UFRJ/COPPE, 2017.

IX, 13 p.: il.; 29, 7cm.

Orientador: Edmundo Albuquerque de Souza e Silva

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2017.

Bibliography: p. 13 – 13.

1. Change Point Detection. 2. Time Series. 3. Machine Learning. I. Albuquerque de Souza e Silva, Edmundo. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

CHANGE POINT DETECTION IN END-TO-END MEASUREMENTS TIME SERIES

Diego Ximenes Mendes

Janeiro/2017

Orientador: Edmundo Albuquerque de Souza e Silva

Programa: Engenharia de Sistemas e Computação

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

CHANGE POINT DETECTION IN END-TO-END MEASUREMENTS TIME SERIES

Diego Ximenes Mendes

January/2017

Advisor: Edmundo Albuquerque de Souza e Silva

Department: Systems Engineering and Computer Science

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Contributions

1.2 Dissertation Outline

Chapter 2

Change Point Detection

A change point detection algorithm is concerned to identify points in time where the statistical properties of a time series have changed. This problem have a broad application in different knowledge fields, and in general, the algorithms performance are closely related with the data characteristics. Also, when the latent information of the procedures which generated a time series is abscent the target statistical properties and other features can be considered subjective, bringing difficulties not only in the detection phase but also in the problem formalization.

In this context, this chapter formalizes the problem and briefly presents several change point detection algorithms. The literature of this area is extensive, and it is common to find methods which presents a poor performance due to the fact that doens't fit the data caracteristics and detection goals, or because it was only analyzed through a theoretical and limited perspective. Therefore it was chosen a set of methods that can provide a good practical and theoretical perspective and also flexibility to adaptations and better handling of input peculiarities. Also through the presentation is presented several obstacle when dealing with real data and some adopted solutions which are not presented in the literature.

2.1 Problem Definition

The problem can be categorized in offline or online. In the offline version, to decide that a point in t is a change point the algorithm has available all the time series, including information past and future of t. In the other hand, in the online version the information is available up to time t. It is intuitive that the offline version is more robust since there is more information to make a decision. In practice, to increase the statistical confidence of the decision, the online definition is relaxed and to decide if a point in time t is a change point it is possible to use information to a small window ni the future, that means wait until more data is available, it is a trade off between minimize the time to detect a change and correctly detect that it is really

a change point. Therefore in some cases, the online version can transformed in the offline version modifying the input availability. The choice between these options are defined by the application domain, in some applications data are processed in real time and the change points must be detected as soon as possible, but in others the change points are idenifyied by historical purpouses data of all time series is available.

In this work it is considered the following input and change points caracteristics, which were defined considering the final application area:

- Univariate time series. However, is possible to easily extend several methods presented here to deal with multivariate data.
- Unevenly time series, that is, data is not regularly sampled in time.
- Unknow number of change points.
- Time series with different lengths.
- Different number of points between change points.
- It is focused on changes in the underlying mean and distribution, disconsidering other kind of changes such as periodicity.
- Outliers are not considered statistical changes.
- There is no latent information of the time series.
- It is considered the online and offline options.

2.2 Notation

An univariate time series composed of n points is defined by two vectors: $\mathbf{x} = (x_1, ..., x_n)$ and $\mathbf{y} = (y_1, ..., y_n)$. The value y_i indicates the $i-$th sampled value and x_i indicates the associated sample time. It is assumed that the points are sorted by time, that is, $x_{i-1} < x_i$ for $i = 1, ..., n$. Since we consider unevenly time series $x_i - x_{i-1}$ can be different for different i values. For $s \geq t$ the following convention is adopted $\mathbf{y}_{s:t} = (y_s, ..., y_t)$.

The presence of k change points indicates that the data is splitted into $k + 1$ segments, also called windows. Let τ_i indicate the $i-$th change point for $i = 1, ..., k$. Also let $\tau_0 = 0$, $\tau_{k+1} = n$ and $\boldsymbol{\tau} = (\tau_0, ..., \tau_{k+1})$. Then, the $i-$th segment is defined by $\mathbf{y}_{\tau_{i-1}+1:\tau_i}$, assuming that $\tau_{i-1} < \tau_i$ for $i = 1, ..., k + 1$.

2.3 Sliding Window Techniques

Through the previous definitions, mainly a change point detection algorithms aim to find both k and $\boldsymbol{\tau}$.

The sliding windows techniques uses two sliding windows over the data stream and then reduce the problem of detecting change points over data to the problem of testing wheter data from the two windows were generated by different distributions. The most common approach is to consider the distance between the two empirical distributions as the base to infer the change points. Let $d(\mathbf{a}, \mathbf{b})$ be the distance between two empirical distributions defined by two windows. Above is a pseudocode of a simple algorithm.

Algorithm 1 Sliding Window

```
1: i ← 1
2: while i + 2m − 1 ≤ n do
3:     if d(y_{i:i+m−1}, y_{i+m:i+2m−1}) then
4:         Report i + m − 1 as a change point
5:         i ← i + m
6:     else
7:         i ← i + 1
8:     end if
9: end while
```

n this approach when the distance between the distributions of the two windows are above some threshold than a change point is reported. This is a common a aproach for a online application, however it is possible to increase the precision in offline cases. As an example the figure presents the distance between two sliding windows in a simulated time series. The segment from 1 to x was generated sampling a $N(..., ...)$ and the segment from $x+1$ to y was generated sampling from a $N(..., ...)$.

It is possible to observer that peak of the distance is the exact location where the distributions changed. However using the threshold method it is possible to infer the localtion of the change point translated to the past. Therefore, instead of using a threshold is possible to use peak detection algorithm is distance of two sliding windows function.

The choice which distance function to use have a great impact on the performance. There are several empirical distributions functions that could provide a quantitative metric of distance between distributions such as mahalanobis distance, bhattacarya distance, hellinger distance, jensen shannon and KullbackLeibler divergence. In practice the Kullback-Leibler and bhattacarya distance were not considered besides they can be used to compare distributions they are not distance

fucntions since doesn't obey the trinagle rule. To work with real data the other cited ones involves represents the distributions through binarization, and these metrics compute the distance through a bin to bin comparison. But in the following toy problem it is

The work **??**tate that to improve performance it is possible to use parallell executions of the same this same sliding window algorithm but with different windows sizes, which enable to capture segments with different sizes more easily.

2.4 Optimization Model

Given a fixed value of k, one approach is to define a cost function that measure the homogeneity of a segment and therefore choose the change points that globally optimize this homogeneity. Let the cost of the i-th segment be defined as $C(y_{\tau_{i-1}+1:\tau_i})$. The cost of a segmentation is then sum of all segments costs.

A commom choice for function C is the MSE (Mean Squared Error) which can capture changes in the mean. Another usual approach is to consider a distribution changes through negative maximum log-likelihood functions, considering that data within a segment is iid.

Therefore, given a fixed k, the optimal segmentation is obtained through the following optimization problem, which is called the constrained case:

$$\min_{\boldsymbol{\tau}_{1:k}} \sum_{i=1}^{k+1} C(y_{\tau_{i-1}+1:\tau_i}) \tag{2.1}$$

This problem can be solved using dynamic programming with $O(kn^2 f(n))$ time complexity, where $f(n)$ is related with C evaluation. Several segment cost functions can be evaluated in $O(1)$ after a $O(n)$ preprocessing phase, implying in an overall $O(kn^2)$ complexity. It is possible to proof that MSE, negative maximum log-likelihood function of normal, exponential, poisson and binomial distributions have this characterestic. Also the formulation can consider a minimum value of a segment length.

Using both of these continuous distributions as the cost function can led to practical dificulties. The first one is that segments can form degenerate distributions, that is, the points of a segment can have zero variance. In these cases the negative likelihood functions will not be defined. Two approaches to avoid this situation is 1) try to avoid degenerate segments adding a white noise to the time series. 2) consider a constant and small value to degenerate segments. Since a single point compose a degenerate distribution to handle this case the options are the xonstant value or doesn‘t allow segments with length equal to one.

Since the likelihood of different continuous distributions can‘t be directly compared, is not possible to apply the segmentation algorithm considering different types of continuos distributions. One possibility to handle this is to apply automatic methods to check which kind of distribution fits better to the segment, such as Kolmogorov-Smirnov. This was not approached in this work due to the computation time efficiency decrease and that since we deal with small number of data possibly these methods would have a poor performance.

Using discrete distributions is possible to direct compare the likelihood of two different distribution types.

When the number of change points is unknown a common approach is to introduce a non decreasing penalty function $g(k)$, the bigger k biggest is the penalty. Then the new optimization problem is, which is called the penalized case:

$$\min_{k,\boldsymbol{\tau}_{1:k}} \sum_{i=1}^{k+1} C(y_{\tau_{i-1}+1:\tau_i}) + g(k) \tag{2.2}$$

An approach to solve this problem is to solve the constrained for $k = 0, ..., K$. This lead to an $O(K^2 n^2 f(n))$. However if the penalty function is linear in k the problem can be formulated in a more efficient way and solved in $O(n^2 f(n))$ time complexity with dynamic programming..

Also there are several pruning algorithms to speedup the computation with the idea of eliminating from the dynamic programming computation some values of τ which can never lead to the optimial solution. Which is the case of the followinf algorithms pDPA, PELT and FPOF.

2.5 Bayesian Inference

I will erase this section if I don't use bayesian inference. Describe Fearnheard (offline) and MacKay (online) solutions. Say that there are other versions.

2.6 HMM (Hidden Markov Model)

There are several approaches to change point detection using HMM. The idea that each segment of the time series if associated with a specific latent variable of the procedure that generate the time series has a direct interpretation through a HMM model. Each hidden state of a HMM model is direct associated with a segment and the observation distribution is the distribution of that segment. Therefore using a HMM to detect change points is directly connected to identifying when the hidden state change in the HMM when processing the time series through the model.

There are several approaches in the training phase and in the detection phase. For example, given a trained HMM is possible to get the time series to be processed and get best hidden state path that this time series would have theough the viterbi algorithm, then the change points would be the points in time where the hidden states changed. Also is possible to evaluate the probability of change the hidden state in time t and apply a threshold method or peak detection as in the sliding window method. For the training phase is possible to use several time series to train the model using baum welch or simply train only with the same time series that will be processed.

It is important to note that structure of hidden state graph has a big impact in the performance. Using a full connected structure the number of states will defined the number of different distribution parameters of the different segments. Using a left to right structure the number of hidden states will impact the number of segments.

In **??** is stated that when using a full connected structure the the time that a time series keeps in a single hidden state is low, therefore there are a lot of transitions between hidden states and therefore a lot of change points. To overcome this problem **??** suggest using a penalization using a dirichlet prior giving priority to the time series be at the same state in the training phase.

2.7 Performance Evaluation

Describe how datasets are constructed in literature. Describe how an algorithm output is evaluated.

Chapter 3

Dataset

3.1 Description of End-to-End Packet Loss Measurements Time Series

Here will be presented the TGR dataset. Small description on how data are collected, including client informations (geographic position, routes, etc) Plots: distribution between two consecutive measures, autocorrelation after time binarization, loss distribution, hour of day x loss, day of week x loss. Maybe: clusterize clients by distribution or time series.

3.2 Change Points Classification Survey

Describe web system used to get "true" change points. Describe majority voting. Describe how data were divided in train/test dataset.

Chapter 4

Applying Change Point Detection

In each algorithm section I will: Describe adaptations and aproaches. Describe dificulties of this algorithms in real data and in the current dataset that lead to adaptation.

4.1 Preprocessing

Filters applied to time series before presenting to algorithms.

4.2 Tuning Hyperparameters

4.2.1 Grid Search and Randomized Grid Search

4.2.2 Bayesian Optimization

I don't know if I am going to use this method.

4.2.3 Particle Swarm Optimization

I don't know if I am going to use this method.

4.3 Sliding Window

4.4 Dynamic Programming

4.5 HMM

4.6 Bayesian Inference

I don't know I will use this: poor performance.

4.7 LSTM

I don't know I will use this: maybe I will not have enough data.

4.8 Ensembles

If there are enough models to be tested describe how to use ensembles.

Chapter 5

Results

5.1 Classification Accuracy

Present false positive/false negative/...

5.2 Unsupervised Analysis

Clusterize clients according with change points detected and check if latent information of clusters are also clusterized.

5.3 Algorithms Comparison

Compare algorithms results and computational performance.

Chapter 6

Conclusions

Bibliography