National Research University Higher School of Economics

Faculty of Computer Science
HSE and University of London Double
Degree Programme in Data Science and
Business Analytics

BACHELOR'S THESIS
Software Project
Detecting Structural Breaks Using Time-Series Embeddings

Prepared by the student of Group 181, Year 4,
Mikhail Ovyan

Thesis Supervisor:
Candidate of Sciences, Associate Professor, Victor A. Lapshin

Moscow
2022

Abstract

Points of a sudden change in the structure of a time series are often cause for alarm since they can indicate a notable deviation from the data generating process. Detecting these points is hence a very important problem for data analysts and researchers. Identifying structural change is a crucial step in the analysis of time series. In this paper, I present a new and practical approach to using vector embeddings and the Word2Vec model to construct embeddings for time-series data. The clustering and cosine similarity algorithms then utilize these embeddings. Results, produced on different datasets, including synthetic and real-world data, surpass many of the well-known methods to detect structural breaks. Besides, the approach includes two completely different methods for change point classification, which is efficient in memory and time complexity. Finally, this approach is capable of producing decent results while not requiring any specific properties of time series.

Table of Contents

Introduction

Traditionally, in the study of time series there is an assumption that the statistical properties of the observed are constant in time or change slowly. However, working with the real-life datasets this property is not always satisfied. Therefore, for many practical purposes, the detection of an abrupt change in the properties of the observed series occurring at an unknown point in time in advance is an important task.

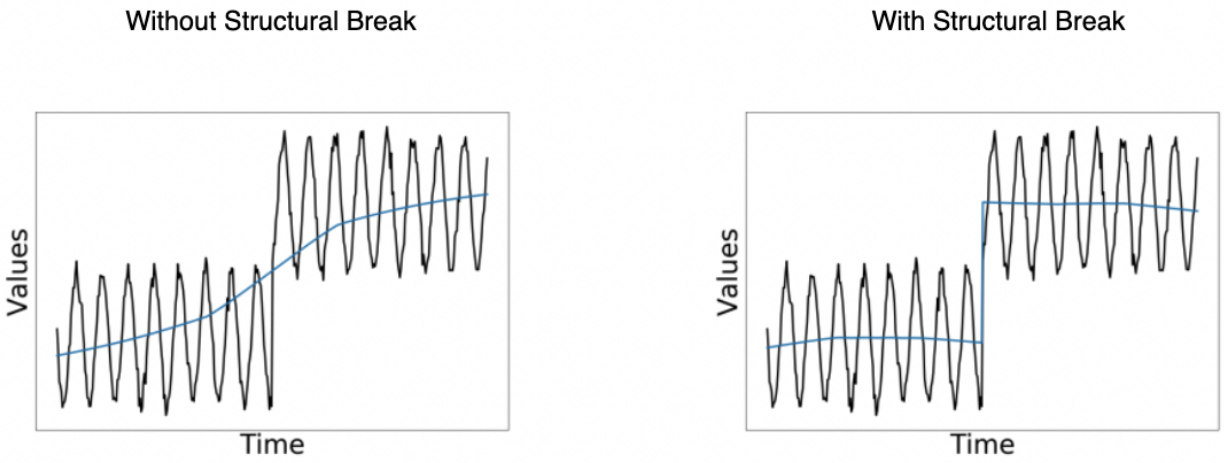


Figure 1.1 Analyzing a trend line in the time-series with a break

Points of sudden change in the structure of a time-series are often cause for alarm since it can be an indication of a notable deviation to the data generating process. Detecting these points is hence a very important problem for data analysts and researchers. Identifying structural change is a crucial step in analysis of time series. The longer the time span, the higher the likelihood that the model parameters have changed because of major disruptive events, such as the 2007–2008 financial crisis and the 2020 COVID–19 outbreak [1]. Models that previously worked well become imprecise and misleading in these cases. The identification of periods with different dynamics is a very important task. This task can be seen as a segmentation.

There are generally two types of such segmentation: a priori and a posteriori. A priori segmentation is the task of predicting the exact moment of discontinuity. A posteriori segmentation means the task of dividing the available time series into several segments with different dynamics. In this work, I will focus on the posteriori segmentation. Detecting the existence of breaks and dating them is therefore not only necessary for estimation purposes but it is also crucial for understanding root cause of the changes and effects it generates.

The times in which the parameters change is called “change points” in the statistics literature and “structural breaks” in economics. As both terms are synonymous to each other, in the paper I will use the latter term or just “breaks”.

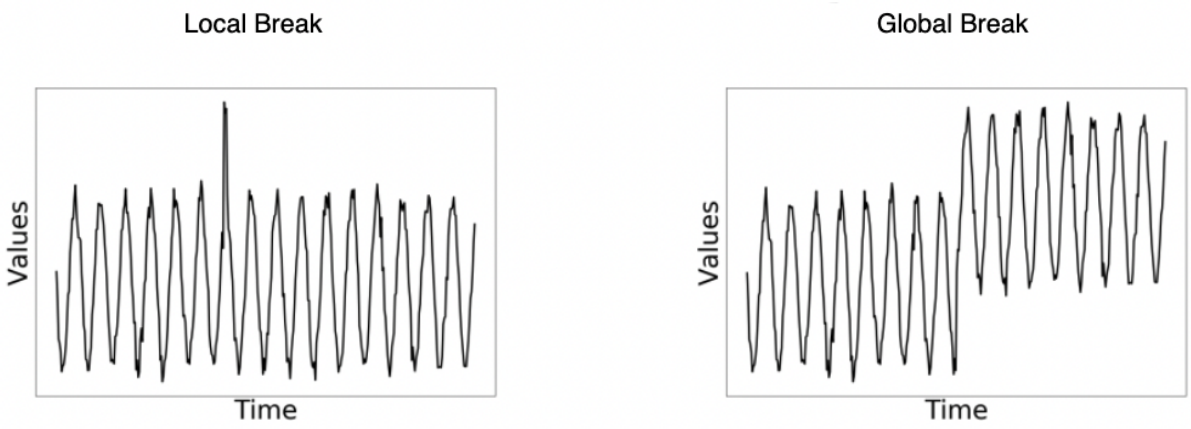


Figure 1.2. Local and Global structural breaks

There are two general types of structural breaks: a local break and a global break, which are demonstrated on the Figure 1.2. In this paper, I will focus on the global type of the structural breaks because of the features of the method I will propose later in this work.

The present paper dived into five sections. The first section outlines the overview

of a structural break, the second section deals with the different approaches which are currently used to analyze time-series and to segment disruptive intervals, including tests for known breakpoints and unknown breakpoints. The third section focuses on the proposed method of using vector embeddings to solve the problem of identifying structural breaks. The final sections describe the conducted experiments on the different datasets, including both real-world and synthetic data, and compare different methods using the appropriate metrics.

Literature review

In statistics and econometrics literature there is an extensive amount of work on testing for structural breaks. Earlier works such as Chow (1960) tests for parameter stability under a known break date using a F-statistic, and Quandt (1958, 1960) suggests using a maximum F-statistic over all values of the potential break date when the break date is unknown. Andrews (1993) studies the properties of such tests and derives the asymptotic distribution of the test statistic. Brown, Durbin, and Evans (1978) provide a test on the stability of parameters by considering partial sums of the standardized forecast errors of rolling regressions, referred as the CUSUM test.

The Chow Test

The Chow (1960) [2] test was one of the first tests which set the foundation for structural break testing. The main idea is that if we assume that the parameters do not change over time then the forecast for out-of-sample data should be unbiased. There is a null hypothesis of no presence of a structural break against the alternative hypothesis that there is in fact a structural break at a certain point in time T_b. The test studies a linear model separated into samples at a chosen time point, such that:

$$Y_t = x'_t\beta_1 + u_t, \text{ for } t \leq T_b$$

and

$$Y_t = x'_t\beta_2 + u_t, \text{ for } t > T_b$$

The test estimates coefficients for each period and forecasts in the future to compute an F-statistic to determine the stability of the coefficients between two segments.

One huge drawback of the Chow test is that it is necessary to choose a concrete point in time to test for the structural break. What is more, the break must be exogenous, or then the assumption of the standard distribution of the statistic will not be valid.

The Quandt Likelihood Ratio Test

The Quandt Likelihood Ratio (QLR) (1960) test builds on the Chow test and attempts to eliminate the need for picking a break point by computing the Chow test at all possible break points. The largest Chow test statistic across the all points potential break points is chosen as the Quandt statistic as it indicates the most likely break point.

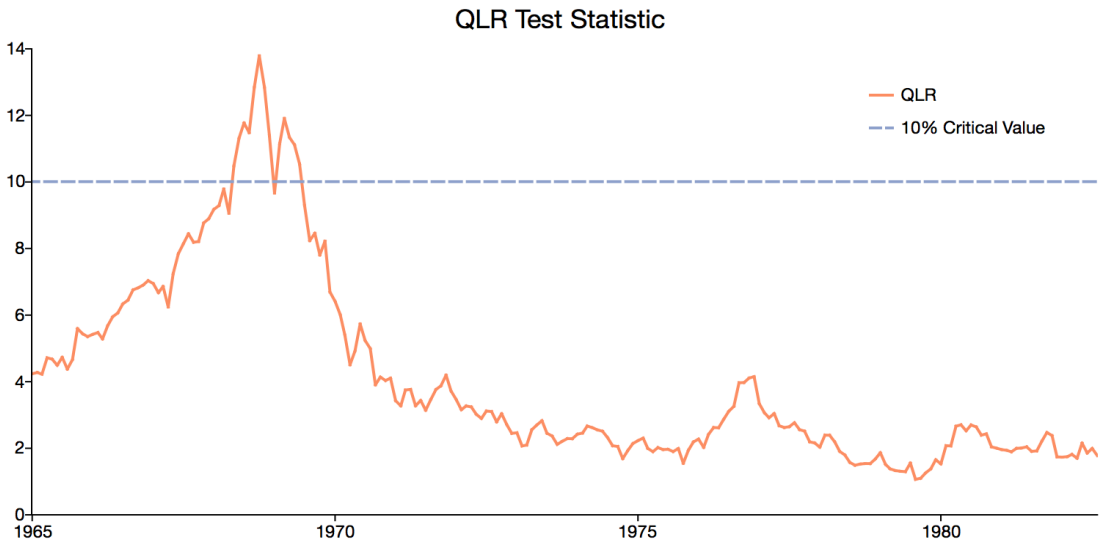


Figure 2.1. QLR test

This test was widely unusable because the limiting distribution of the test statistic under the assumption of an unknown break point was not known. However, the test became statistically relevant when Andrews and Ploberg (1994), developed an applicable distribution for the test-statistic for cases such as the Quandt test.

The CUSUM Test

In their 1975 paper Brown, Durban, and Evans propose the CUSUM test of the null hypothesis of parameter stability. The CUSUM test for instability is appropriate for testing for parameter instability in the intercept term. It is best described as a test for instability of the variance of post-regression residuals. The CUSUM test is based on the recursive least square estimation of the model:

$$Y_t = x'_t\beta_t + u_t, \text{ for all } k+1 \leq t \leq T$$

This yields a set of estimates for β_k, $\beta_{k+1, \ldots,}$ β_T. The CUSUM test statistic is computed from the one-step-ahead residuals of the recursive least squares model. It is based on the intuition that if β changes from one period to the next then the one-step-ahead forecast will not be accurate, and the forecast error will be greater than zero.

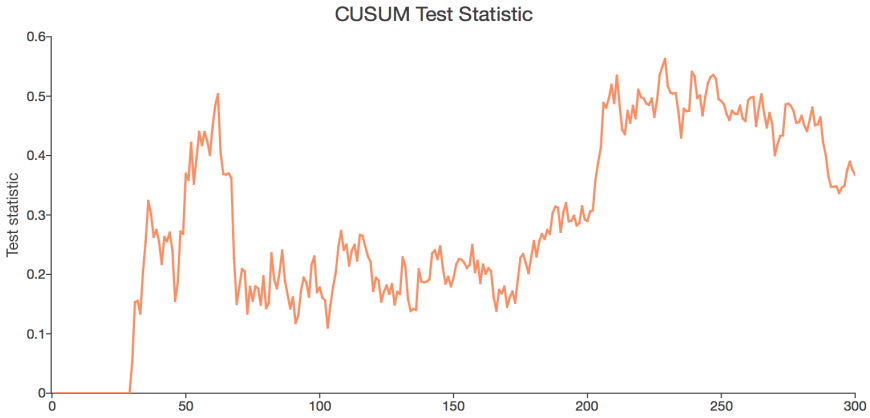


Figure 2.2. CUSUM Test Statistic value

Xtbreak

In article "Testing and Estimating Structural Breaks in Time Series and Panel Data in Stata" (2021) [1] authors introduced a new community contributed command called xtbreak, which provides researchers with a complete toolbox for analyzing multiple structural breaks in time series and panel data. Xtbreak model has the following structure:

We consider the following model with N units, T periods and s structural breaks:

$$y_{i,t} = x'_{i,t}\beta + w'_{i,t}\delta_j + e_{i,t}, \tag{1}$$

where $t = T_{j-1}, ..., T_j$ and $j = 1, ..., s+1$ with $T_0 = 0$ and $T_{s+1} = T$. Hence, there are s breaks, or $s+1$ regimes with regime j covering the observations $T_{j-1}, ..., T_j$. In order emphasise the break structure, we can write (1) regime-wise;

$$\begin{aligned}
y_{i,t} &= x'_{i,t}\beta + w'_{i,t}\delta_1 + e_{i,t} \text{ for } t = T_0, ..., T_1, \\
y_{i,t} &= x'_{i,t}\beta + w'_{i,t}\delta_2 + e_{i,t} \text{ for } t = T_1, ..., T_2, \\
&\vdots \\
y_{i,t} &= x'_{i,t}\beta + w'_{i,t}\delta_{s+1} + e_{i,t} \text{ for } t = T_s, ..., T_{s+1}.
\end{aligned}$$

For $N = 1$, this is a time series model, while for $N > 1$, it is a panel data model. The dependent variable $y_{i,t}$ and the regression error $e_{i,t}$ are scalars, while $x_{i,t}$ and $w_{i,t}$ are $p \times 1$ and $q \times 1$ vectors, respectively, of regressors. The coefficients of the regressors in $x_{i,t}$ are unaffected by the breaks, while those of $w_{i,t}$ are affected by the breaks. It is possible that all independent variables break, in which case $x'_{i,t}\beta$ is defined to be zero. The break dates are common for all units. This is a very common assumption that is reasonable in settings where the frequency of the data is not high. Let $\mathcal{T}_s = \{T_1, ..., T_s\}$ be a collection of s break dates such that $T_j = \lfloor \lambda_j T \rfloor$, where $\lambda_0 = 0 < \lambda_1 < ... < \lambda_s < \lambda_{s+1} = 1$. By specifying the breaks in this way we ensure that they are distinct from one another and that they are bounded away from the beginning and end of the sample. This is important because we need to be able to estimate the model within each regime.

xtbreak can detect the existence of breaks, determine their number and location, and provide break date confidence intervals. The toolbox is based on asymptotically

valid tests for the presence of breaks, a consistent break date estimator, and a break date confidence interval with correct asymptotic coverage. In fact, xtbreak includes three tests:

(i) A test of no structural change against the alternative of a specific number of changes

(ii) A test the null hypothesis of no structural change against the alternative of an unknown number of structural changes, and

(iii) A test of the null of s changes against the alternative of s + 1 changes.

The package also includes an algorithm that employs the last test consecutively to estimate the true number of breaks. The tested break dates can be unknown or user-defined, as when researchers have additional information and wish to examine whether there was a break in a specific point in time. Once the presence of breaks has been tested and confirmed, xtbreak estimates the locations of the breaks and provides the associated confidence intervals.

However, we are interested in the way this model identifies all breakpoints and segments the intervals of time-series. This is done using the idea that if the model has the correct number of breaks and the correct points in time, then the SSR should be smaller than for a model with a larger or smaller number of breaks.

For example: Break in period 2 ($T_1 = 2$), then $SSR = SSR(1,2) + SSR(2,T)$.

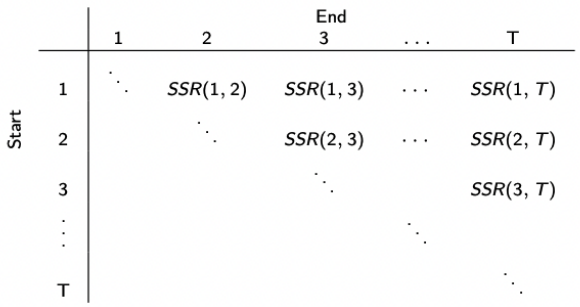


Figure 2.3. Dynamic Programming algorithm, Bai and Perron (2003)

Thus, the only task is to minimize SSR, which is implemented in xtbreak using the dynamic programming algorithm from Bai and Perron (2003), Figure 2.3.

Matrix Profile

In their work [3] the authors introduce a novel scalable algorithm for time series subsequence pair similarity search. For any scale datasets, the method can produce highquality approximate solutions in reasonable time. The exact similarity join algorithm computes the answer to the time series motif and time series discord problem as a side-effect, and the algorithm also provides the fastest known algorithm for both these extensively-studied problems.

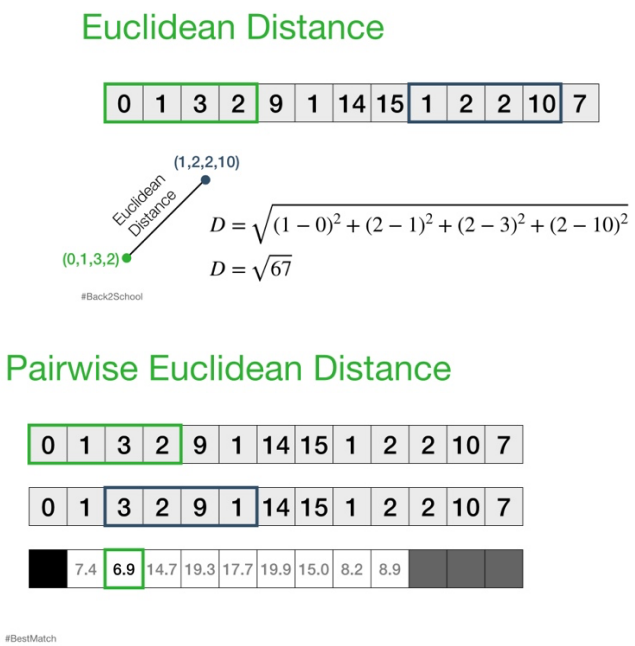


Figure 2.4. Matrix Profile – Pairwise Euclidean Distance

The general idea behind Matrix Profiling is to choose window size (Fig. 2.4, green rectangle) and using sliding window approach calculate pairwise Euclidean distances between these segments. Once this procedure is done for the first segment, the window is moved to the next set of points and the same series of steps is repeated.

In the end, this algorithm yields a matrix of pairwise distances between each segment in the time-series.

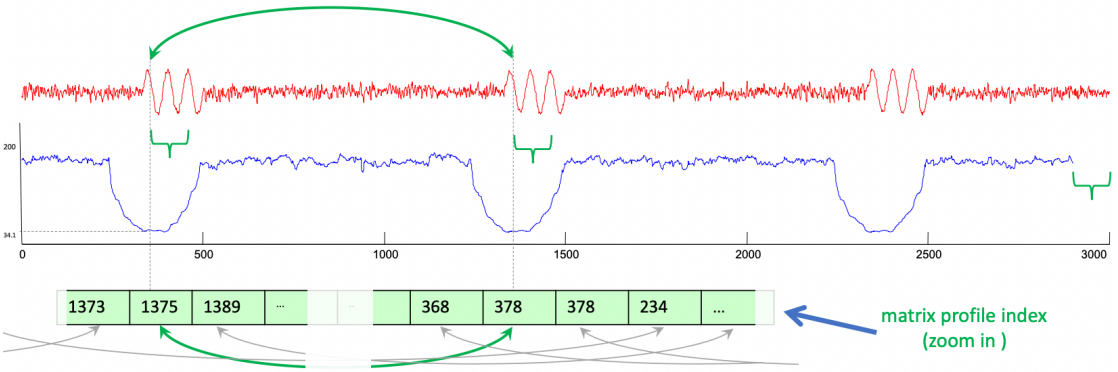


Figure 2.5. Result of Matrix Profile Procedure

The result of the Matrix Profile procedure is demonstrated on Fig. 2.5. The original time-series is highlighted in red color, while the meta time-series is shown in blue color. As it can be seen, there are notable changes in the meta time-series at the points where the original series changed their behavior. The matrix profile and the matrix profile index provide information about the location and distance of all subsequences nearest neighbors.

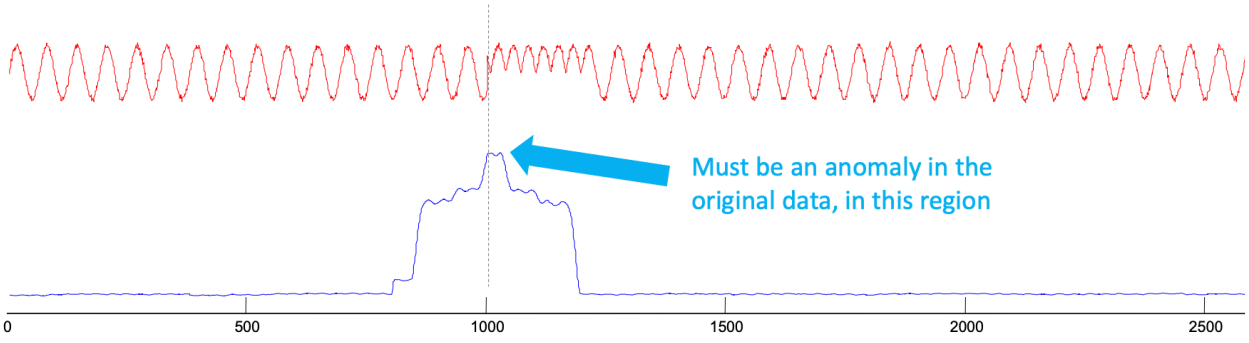


Figure 2.6. Time Series Discord

For our specific problem of identifying structural breaks, we need to introduce Time Series Discords from Matrix Profile. These are specific notations for the abnormally high points in the matrix profile array of distances. These points indicate an anomaly in the time-series and can be used to detect a change in the structure of the time-series or, in another words, a structural break.

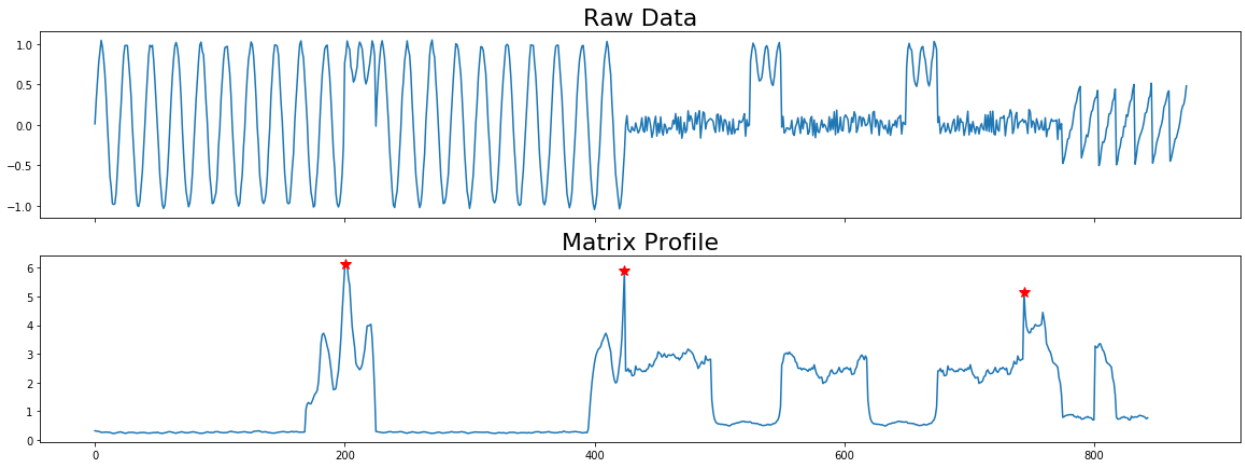


Figure 2.7. Matrix Profile and Discords

Fig. 2.7 also demonstrates a more complex example of the use of matrix profile. Bottom time-series clearly shows that there are several points of abnormal change, which we will later use as a prediction for the structural break locations.

Overall, this chapter focused on the existing methods, both well-known and experimental, which are used to detect structural breaks. Those are just a fraction of all the methods that can be used to tackle the issue. However, the procedures give a notion of how the problem of identification of structural breaks can be solved and they will be used later in the paper as benchmarks to compare with the introduced method.

Main Part

Background

In their work [4] authors proposed a new model Signal2Vec, which was used to classify time-series based on their embedding representation. The dataset consisted of information about energy consumption of different devices in the house and the task was to identify what devices were running in the certain point of time.

The idea was to use tokenization procedure and generate embeddings from the initial data. Once this step was completed, the model was fitted with the embeddings and corresponding devices, which produced those embeddings. Figure 3.1 shows the structure of the algorithm.

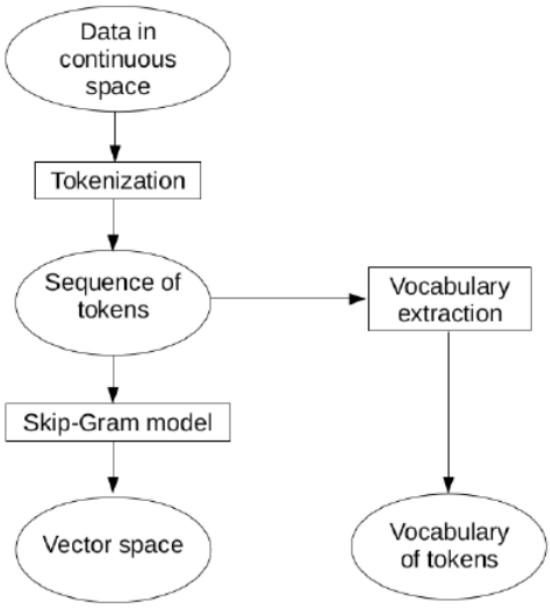


Figure 3.1. Signal2Vec Framework

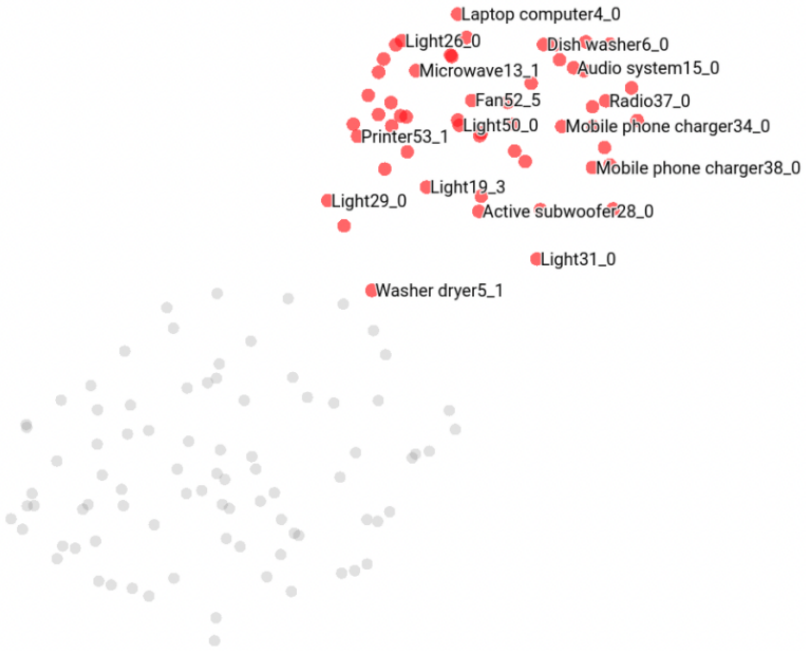


Figure 3.2. TSNE of embeddings

Later in the paper the authors demonstrated (Fig. 3.2) how those embeddings can be represented in vector space. We can clearly see that there are several clusters which include different devices.

Looking at this picture, an idea came to my head: if the embeddings of time-series data differ between each other in a high-dimensional vector space based on their characteristics, then embeddings of the segments of time-series before the structural break should belong to one cluster, and embeddings of the segments after the structural break (once the series changes its properties) should belong to another cluster.

Thus, the general idea is to obtain vector embeddings from time-series and propose an approach to cluster similar embeddings, hence identifying different structures inside the series. Then the boundaries of these structures would be exactly the break points.

Problem Formulation

The problem of locating the structural breaks can be described both as the classification and segmentation problem because we can either divide the time-series into heterogeneous segments, hence borders between them are structural breaks, or we can find the structural breaks, which in fact create segments. In this work I will focus on this problem as the classification one.

As the input we receive a time-series, a vector of pairs (x_t, y_t), which can be in any numeric format. As the result the algorithm should yield a set of points (p_1, …, p_N) – the locations of structural breaks.

Building Embeddings

This section will focus on describing the procedure of building the vector embeddings and data preprocessing. First, we start with a definition of embeddings.

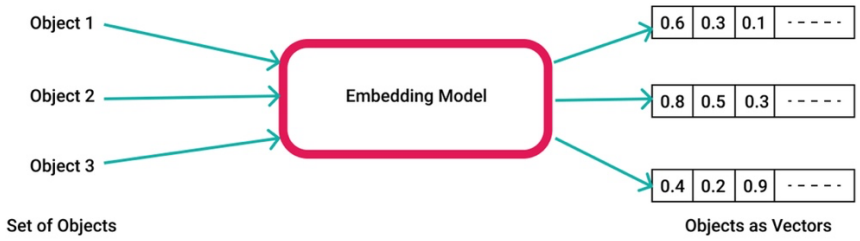


Figure 3.3. Embedding Model producing embeddings

In the context of Neural Networks, a vector embedding is a low-dimensional vector representation of any abstract discrete variable (usually complex). This approach is used in many Machine Learning fields, primary in NLP and recommendation systems.

The major advance of vector embeddings is that they preserve similarity between the objects in the original space and translate it into a new vector space. There are several ways to construct the embeddings.

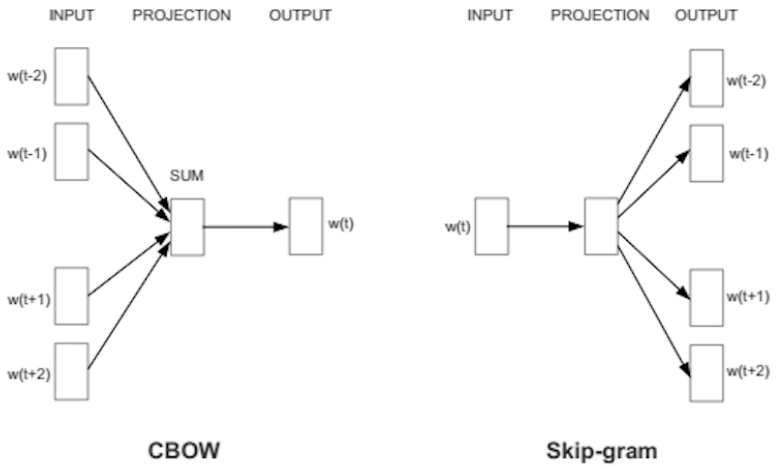


Figure 3.4. CBOW and Skip-gram

In this paper I will focus on the one of the most popular embedding algorithms: Word2Vec. It is a two-layer neural network which is capable of vectorizing words from the processed text. There are two approaches to embed tokens, which can be used in Word2Vec, namely CBOW (Continuous Bag of Words) and Skip-gram. In this paper I will use Skip-gram approach because it works better [5] with small datasets and better represents less frequents words, which suits our problem. The inner workings of the Skip-gram will be omitted in the paper, curious readers can find more information about it in the references.

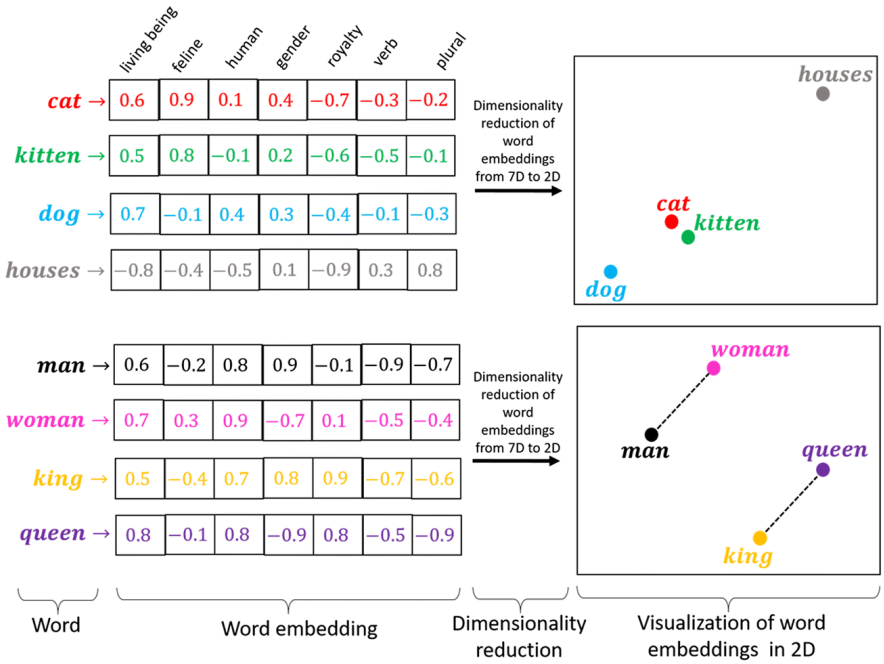
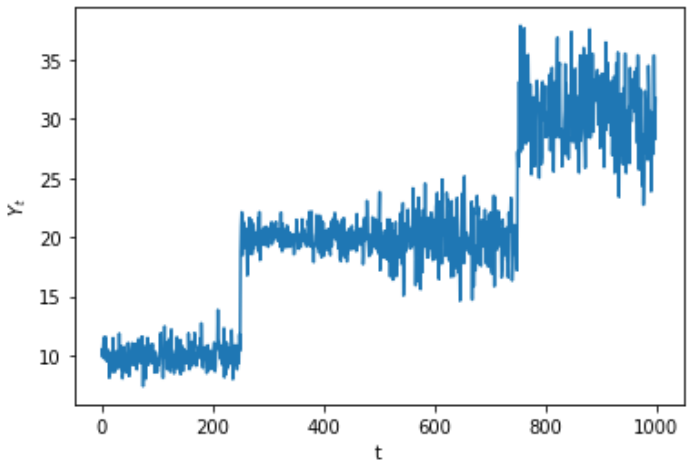


Figure 3.5. Word2Vec Results

Fig. 3.5 demonstrates the resulting vector embeddings produces by Word2Vec. As it can be seen, those embeddings preserve some initial properties. For example, in a vector space, “cat” is closer to “kitten” than to “dog”. What is more if we add vector “woman” to the vector “king” we achieve vector “queen”. This is working correctly due to the ability of Word2Vec model to create internal relations between words that a close to each other in the given text.

In our case we are dealing with numeric time-series, however, as was mentioned before, Word2Vec is only capable of working with text, i.e. string tokens. That is why the first step of the algorithm is to change the datatype of elements from float (or integer) to string. In this case, the problem of rounding arises.



Rounding	Size of vocab
3	974
2	724
1	224

Figure 3.6. Random time-series with 1000 elements

If we generate a random time-series of 1000 elements (shown on Fig. 3.6) and then run Word2Vec procedure, it will yield a vocabulary or in other words all unique tokens (words). The table on Fig. 3.6 shows the size of these vocabularies for different rounding (number of decimal digits). As it can be seen, the lower the number of decimal digits, the lower the size of the vocabulary. Too many unique tokens may result in a poor accuracy of embeddings, while too few tokens may result in overfitting and poor representativity. This problem will be explored further in the experiments section of the paper.

Model Description

Once the initial time-series goes through the processing described in the previous chapter and Word2Vec procedure is executed, for each point from the series we receive a vector embedding.

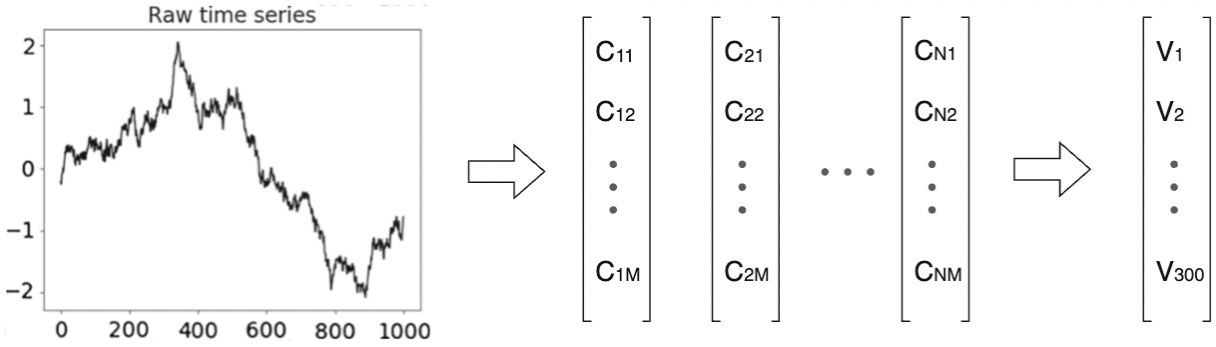


Figure 3.7. Raw time-series to embeddings

But at this stage it vector-representations of each point do not carry any significant important information, and hence should not be clustered and used further. In NLP notation it means that by comparing "words" we lose the fact that it is a continuous series of "words". Hence, it is better to compare "sentences" (sets of words) between each other and cluster them.

To obtain such "sentences" or for the time-series data I will call them segments, we take the average of all vectors inside this segment. This procedure is demonstrated on Fig. 3.7, where each point of the segment is converted to vector [C_1 … C_N], where N is the dimensionality of the embedding. Then, these vectors are averaged into one segment (or sentence) embedding [V_1 … V_N]. This is a well-known approach to receive sentence embeddings from word embeddings and described in detail in the original paper [6].

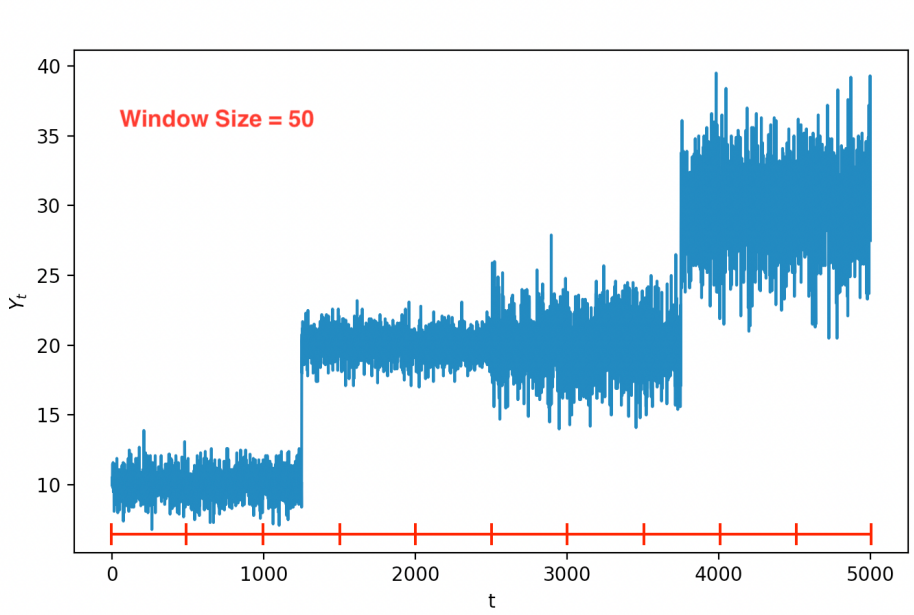


Figure 3.8. Segment separation for window size equals 500

To divide the initial set of points into segments it is important to decide the length of each segment or in other words the window size. On Fig. 3.8 shown the case of the time-series with the length 5000 and window size equals to 500. Smaller window size may introduce more noise to the embeddings, while greater size blurs the boundaries of the structural breaks. This should be considered as the hyperparameter of the model.

Clustering Approach

Once the vector embeddings for each segment were obtained, we can visualize them on a 2-dimensional plane. To obtain a lower-dimensional representation of embeddings Principal Component Analysis (PCA) is applied. The idea of this procedure is to summarize the data from the given dataset into a smaller number of variables, which are called principal components. It creates a lower-dimensional representation of the data preserving as much variance as possible, which is one of the most important parameters to preserve the information value the dataset has.

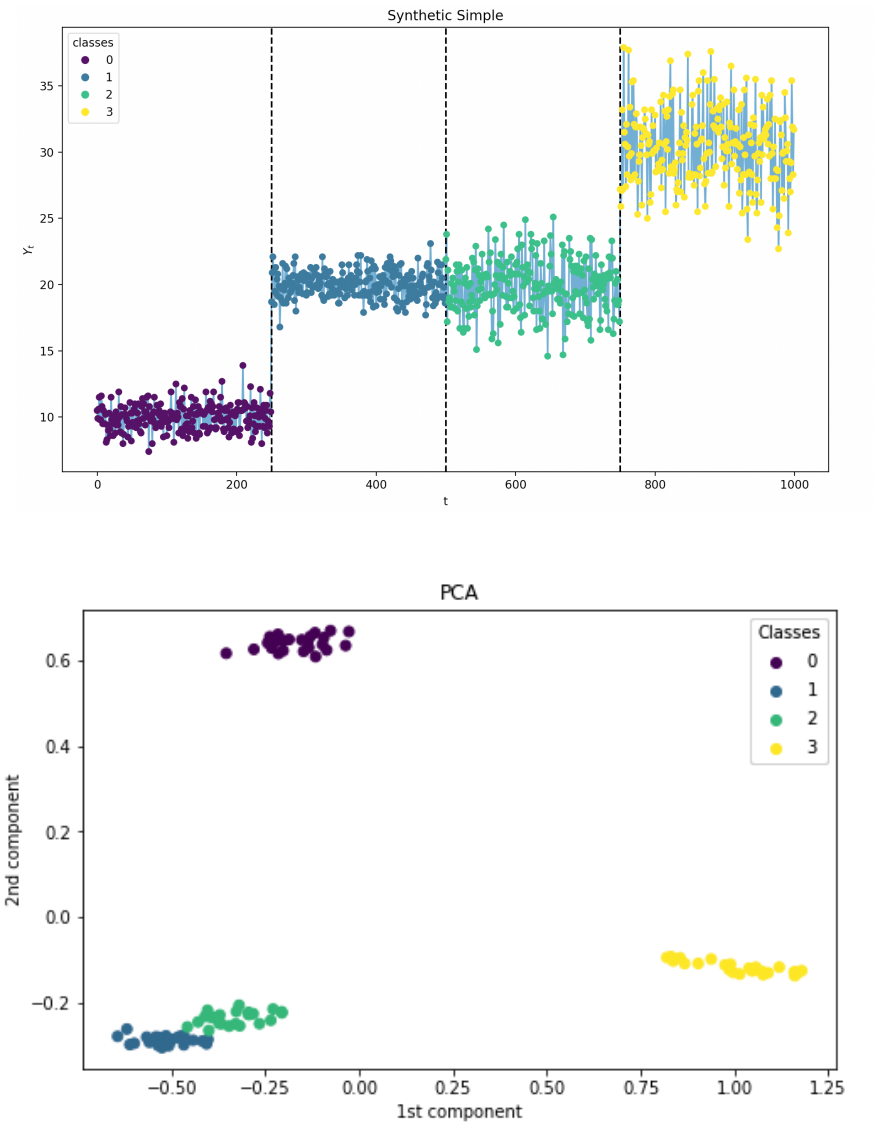


Figure 3.9. Generated dataset and corresponding PCA. Dashed lines show structural breaks. Window size = 50, N = 1000

Fig. 3.9 demonstrates a randomly generated time-series with three structural breaks producing four different segments. All segments were generated from the random normal distribution, the first two has different mean, the third one has different variance, and the fourth one has both parameters different from other segments.

From the PCA plot we can see that there is a clear presence of several clusters, namely classes 0 and 3 have their distinct clusters, while classes 1 and 2 are hard to distinct. These results match the initial time-series, where the first and the last structural break can be easily noticed, while the one in the middle is not very distinctive.

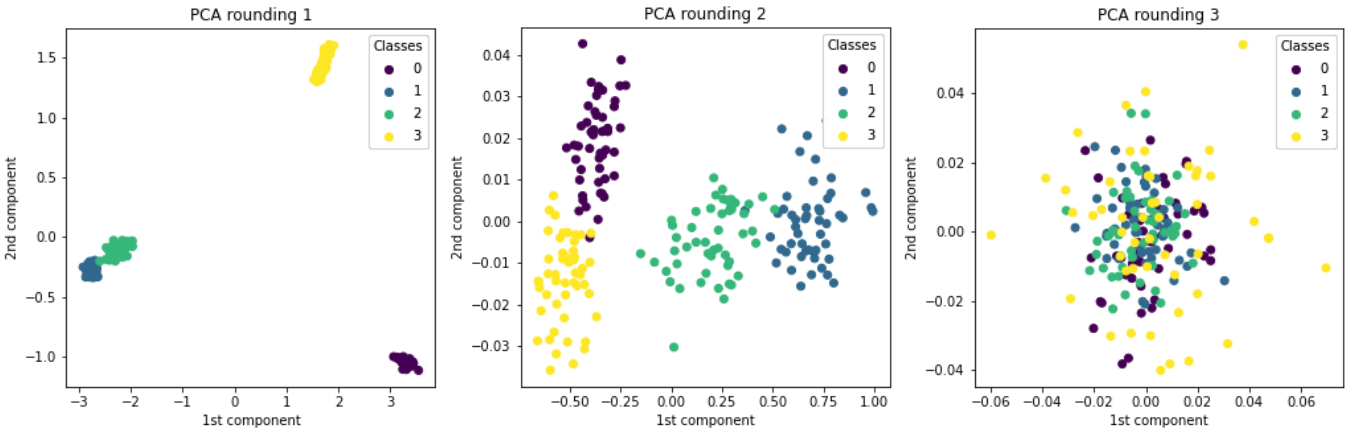


Figure 3.10. PCA clusters based on decimal points

Previously, it was mentioned that rounding plays a vital role in the size of the vocabulary and hence the accuracy of the algorithm. Fig. 3.10 shows how with an increase in the number of decimal points the clusters start to disintegrate. However, on the plot with PCA with rounding equals to 2, the distinction between class 1 and class 2 can be viewed easier, which also proofs that lower rounding leads to overgeneralization and overfitting.

In general, these plots show that it is possible to distinguish different segments of time-series into different clusters. Then boundaries of these clusters can be considered as the points of structural breaks.

To build such unsupervised clusters among the vector embeddings I will use K-Means Clustering method. Basically, it sets a random centroids in the vector space and then tries to minimize the total within-cluster variation. Fig. 3.11 describes the algorithm in detail.

```
K-MEANS({x_1, ..., x_N}, K)
 1  (s_1, s_2, ..., s_K) ← SELECTRANDOMSEEDS({x_1, ..., x_N}, K)
 2  for k ← 1 to K
 3  do μ_k ← s_k
 4  while stopping criterion has not been met
 5  do for k ← 1 to K
 6     do ω_k ← {}
 7     for n ← 1 to N
 8     do j ← arg min_j' |μ_j' − x_n|
 9        ω_j ← ω_j ∪ {x_n}  (reassignment of vectors)
10     for k ← 1 to K
11     do μ_k ← (1/|ω_k|) Σ_{x∈ω_k} x  (recomputation of centroids)
12  return {μ_1, ..., μ_K}
```

Figure 3.11. K-Means Algorithm

The next problem that arises is to identify the optimal number of clusters. There are several approaches to solve this issue, for example Elbow method and Silhouette Coefficient. We will not discuss Elbow method because it is hard to automate, and it is impracticable to this task. Hence, we will stick to the Silhouette Method.

Basically, Silhouette value shows how a point is similar to its own cluster compared to other clusters. Where *a(i)* is the average distance of *i* from other points in cluster, and *b(i)* is the average distance of *i* from points other clusters. And the desired number of clusters is the one that produces the lowest Silhouette Score. Worth mentioning that the cosine similarity clustering was also considered, but it provided much worse results compared to K-Means using Euclidean distance.

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, \text{ if } |C_i| > 1$$

and

$$s(i) = 0, \text{ if } |C_i| = 1$$

To efficiently construct clusters and perform similarity search I use Faiss [7], a library made by Facebook AI Research team. I wrote a wrapper to effectively construct K-Means clustering and use them to identify structural breaks.

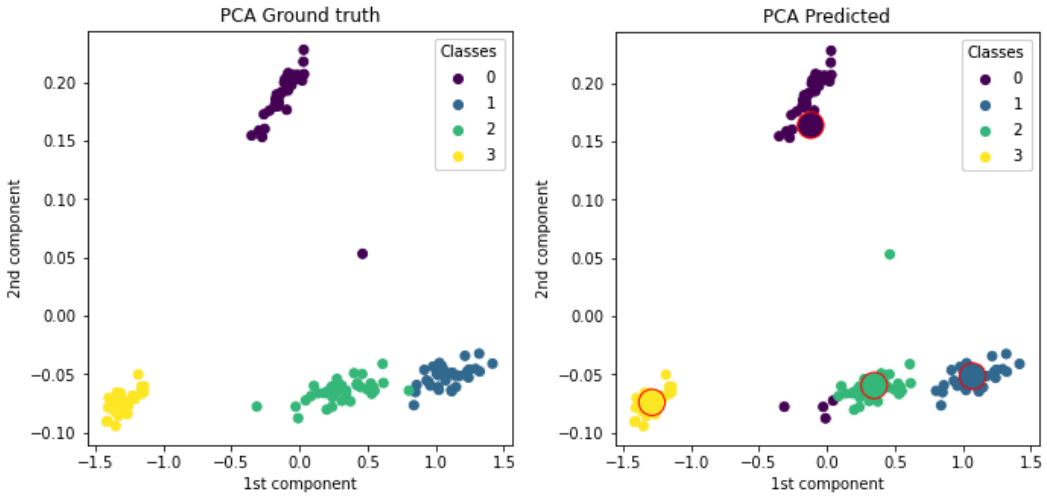


Figure 3.12. PCA ground truth and prediction

On Fig. 3.12 we can see how the algorithm predicted the correct number of clusters, which is 4, however some points were misclassified. This will shift the predicted border of the structural break by 1-2 window sizes.

Cosine Similarity Approach

Another approach to work with vector embeddings of segments can be to measure cosine similarity between neighboring segments. The cosine similarity is the measure of how two vectors are similar to each other.

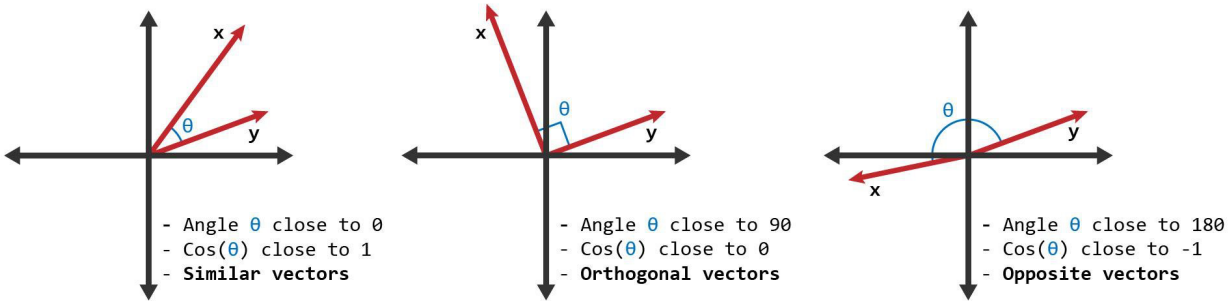


Figure 3.13. Cosine Similarity visually

Cosine similarity is basically the division of the dot product of two vectors by the multiplication of their magnitudes.

$$similarity(A,B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

It ranges from -1 to 1, where the higher the cosine the more similar two vectors are.

Therefore, by calculating the cosine similarity between two segments we can conclude how similar are those. For example, if two segments lay in between a structural break, we can expect their cosine similarity to be lower than the similarity of those segments that belong to one structure.

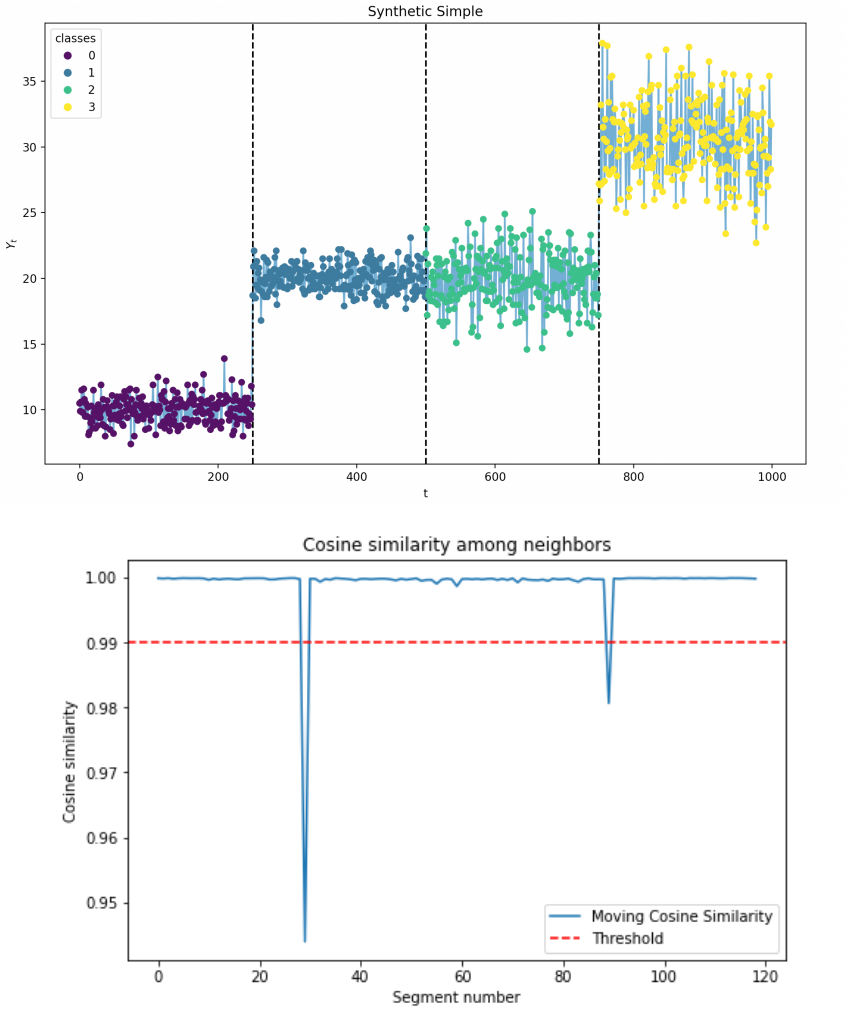


Figure 3.14. Detecting breaks using Cosine Similarity

Fig. 3.14 shows how using cosine similarity approach can find structural breaks in the given dataset. On the bottom plot the moving cosine similarity between neighboring segments is shown. There are different approaches to choose threshold such that any cosines that a lower such threshold should be considered structural

breaks. One approach that I propose is it construct 99% confidence interval and find the lower border for such threshold. The result of such algorithm is presented in Fig. 3.14.

Model Overview

To finalize model description I will shortly summarize the model structure and data manipulations. First, we construct vector embeddings from the initial data with the use of Word2Vec. Then we subdivide the time-series and create embeddings for each segment by summarizing the tokens inside. Once this has been done, we decide between two approaches: Clustering and Cosine Similarity. In clustering we construct the optimal number of clusters and hence identify the boundary segments. In Cosine Similarity procedure we calculate cosine between each pair of neighboring vectors and then decide on the threshold to identify segments producing structural breaks.

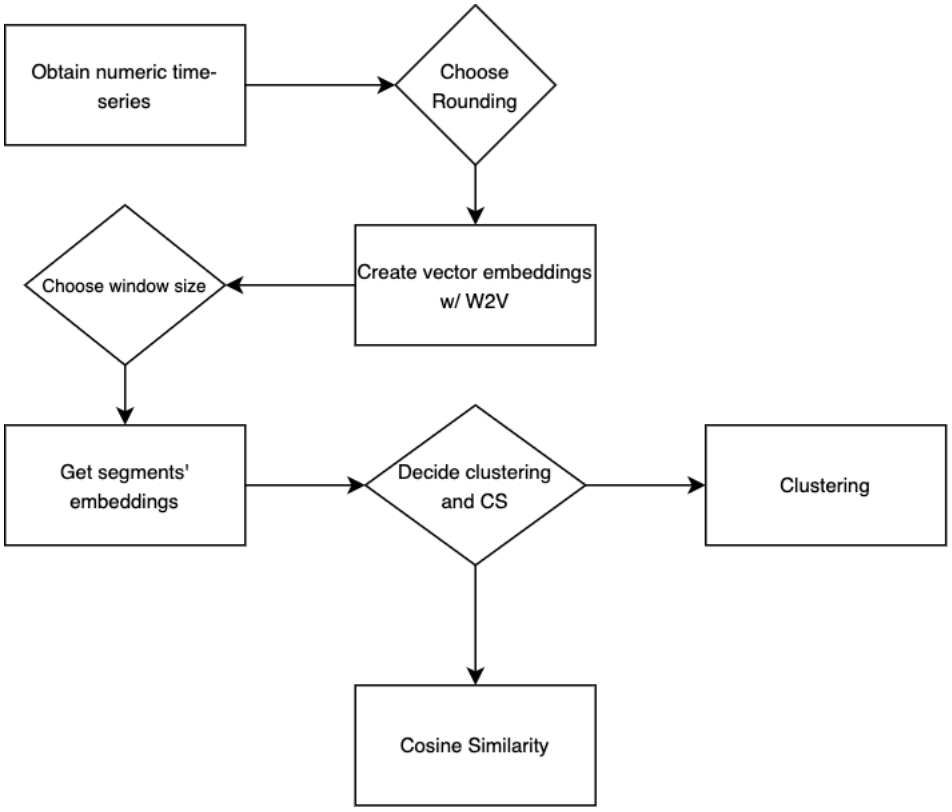


Figure 3.15. Model Overview

Metrics

As was mentioned in the problem statement, since we are dealing with classification task, we should choose metrics accordingly. Each point can be either a break point or not a break point. In the proposed model we work with segments, which are sets of neighboring points, identifying whether the structural break belongs to the segment or not. Hence, in the experiments section we will consider a correct prediction if the true structural break is located in the range of window size or in other words inside of the predicted segment.

The data is very imbalanced: the number of structural breaks usually is much lower than the total number of points in the time-series. Hence, we should focus on metrics that take it into account. These a several metrics that will be used to compare the proposed method with benchmarks:

TPR (True Positive Rate). Also known as sensitivity, the proportion of positive observations (real structural breaks) that were correctly recognized. This shows how effectively the model finds breaks/

FPR (False Positive Rate). Shows how many negative observations were predicted to be positive by the model. In another words, shows how many structural breaks the model predicted, while they were not there.

F1 Score. It is the harmonic mean of precision and recall. It is the number of true positive predictions divided by the number of true positive prediction plus the mean of false positive and false negative predictions.

Data

Synthetic Datasets

Testing different solutions to the problem of the structural breaks' segmentation has been always a cornerstone in the papers. Given that there is no strict mathematical definition of the structural break, it is very difficult to find a labeled time-series featuring breaks and different shapes.

For that reason, it is usual to generate several various synthetic datasets, which would include breaks at the known points and would feature different shapes, trends etc. For example, authors of the "Detection of multiple structural breaks in multivariate time series" [8] paper propose the following model:

$$X_{t,T} = \sum_{j=1}^{4} 1_{(\frac{j-1}{4}T, \frac{j}{4}T]}(t)\theta_j Z_t$$
$$\text{where the } \theta_1 = 1,\ \theta_2 = 1.25,\ \theta_3 = 2,\ \theta_4 = 2.25$$

and Z is a Gaussian white noise process. It features 4 segments which differ by the MA parameters. These breaks are also can be modified with the different variance of Z between them.

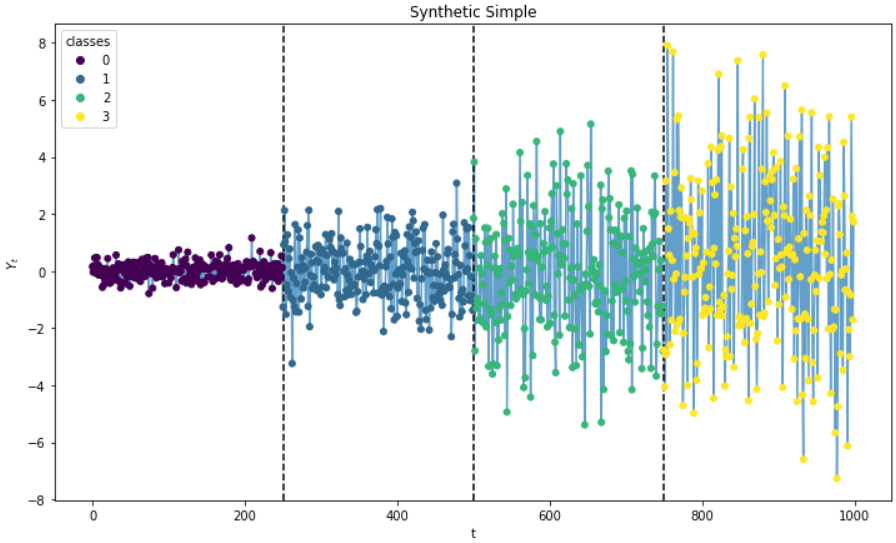


Figure 4.1. Synthetic Dataset with different theta parameters

Another synthetic dataset was proposed in the original "xtbreak" paper [1]. It has the following structure:

> We generate a panel dataset with 100 cross-sectional units ($N = 100$) and quarterly data from 1990Q1 to 2002Q4 ($T = 52$).
>
> Assume two breaks in variable w and no breaks in variable x .
>
> The DGP is:
>
> $$y_{i,t} = \alpha_i + \beta_0 + \gamma_s w_{i,t,s} + \beta x_{i,t} + \epsilon_{i,t}$$
>
> with $\gamma_1 = 2, \gamma_2 = 1, \gamma_3 = 0.5$, $\beta = 1$ and $\epsilon_{i,t} \sim N(0, 10)$.

Thus, featuring 3 breakpoints or 4 different segments with the known location of the bounds.

Turing Change Point Dataset

In the paper "An Evaluation of Change Point Detection Algorithms" [9], the authors state, that while many algorithms for change point detection have been proposed, comparatively little attention has been paid to evaluating their performance on real-world time series. Algorithms are typically evaluated on simulated data and a small number of commonly used series with unreliable ground truth.

Clearly this does not provide sufficient insight into the comparative performance of these algorithms in the field. Therefore, instead of developing yet another change point detection method, they considered it vastly more important to properly evaluate existing algorithms on real-world data. To achieve this, they presented a dataset specifically designed for the evaluation of change point detection algorithms that consists of 37 time-series from various application domains and was manually annotated by volunteers.

All series are standardized to zero mean and unit variance to avoid issues with numerical precision arising for some methods on some of the time series.

Out of the 37 time-series I have decided to choose 4 the most distinctive. Those include the financial data, stock price, and even a scanline of an image.

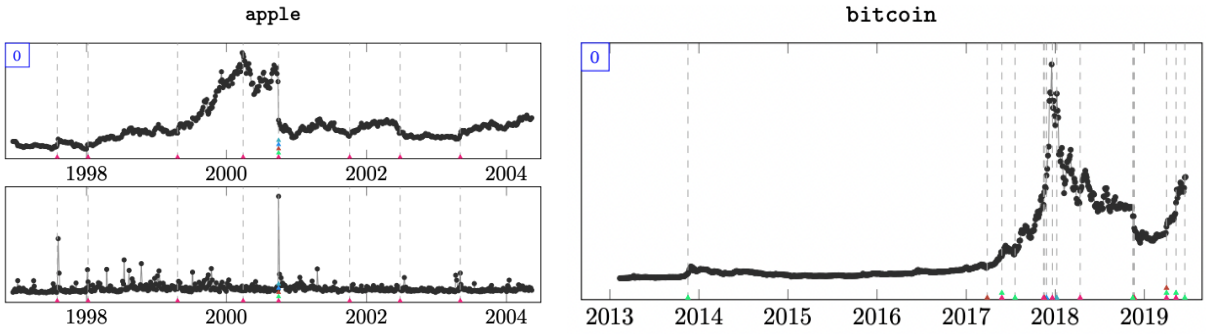


Figure 4.2. Apple stocks and Bitcoin price

Figure 4.2, on the left, features the daily closing price and volume of Apple, Inc. stock for a period around the year 2000. A significant drop in stock price occurs on 2000.09.29. On the right, the price of Bitcoin in USD from 2013 to 2020.

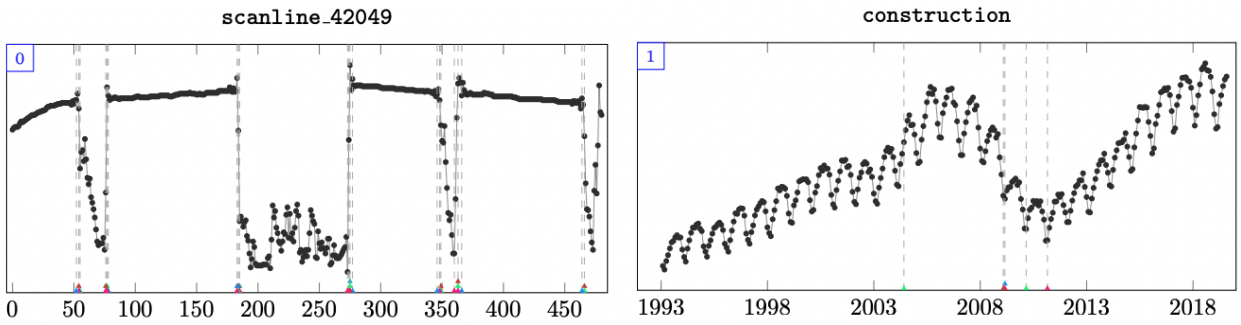


Figure 4.3. Scanline and Construction data plots

Figure 4.3, on the left, features a horizontal scan line of image no. 42049 from the Berkeley Segmentation Data Set. On the right, Total private construction spending in the U.S.

COVID-19 Dataset

This dataset was featured in the xtbreak paper [1] to find the evidence of multiple breaks. The data consists of both aggregate country and disaggregated state level US data about the covid cases and deaths.

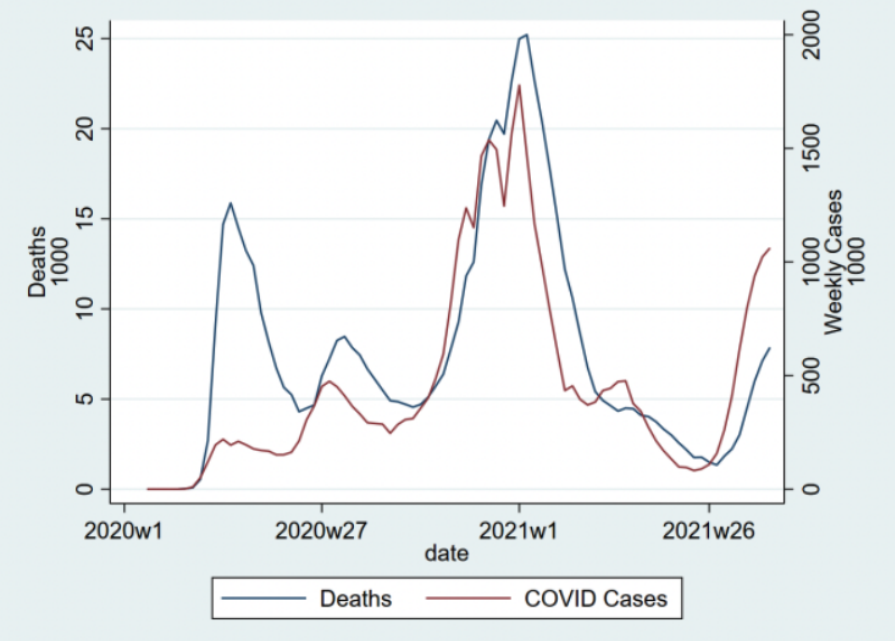


Figure 4.4. Covid-19 cases and deaths from 2020 to 2021

The data is made up by the weekly data on the number of deaths and new cases from the CDC. Starting in mid-December 2020, vaccines were introduced, and the authors of the paper assume the relationship between the number of cases and deaths to be changed. Therefore, it seems reasonable to expect at least two breaks. Figure 4.5 features the two structural breaks proposed by the original authors.

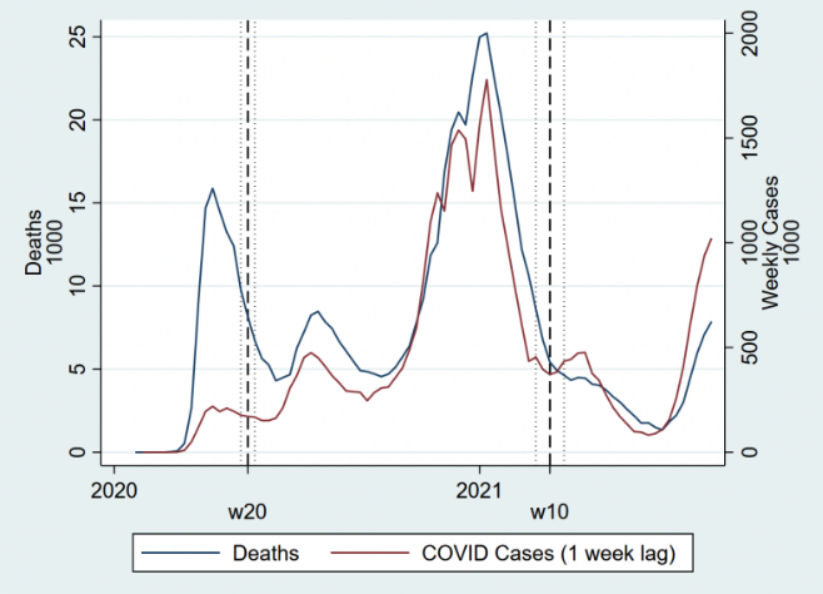


Figure 4.5. Structural breaks in the Covid dataset

Experiments

All presented methods to find structural breaks along with the proposed methods of Clustering embeddings and calculating Cosine Similarity were tested on 7 different datasets described in the previous section.

True Positive Rate											
	Clustering			Cos. Sim.			Chow	Quandt	Cusum	Xtbreak	MP
Rounding	1	2	3	1	2	3					
Synthetic 1	**1**	**1**	0.75	**1**	**1**	0.75	0.75	0.5	0.5	0.75	0.5
Synthetic 2	**0.66**	**0.66**	0.5	**0.66**	**0.66**	0.5	0.4	0.4	0.5	0.5	0.4
Covid	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**	0.25	**0.5**	**0.5**	**0.5**	**0.5**
Apple	**0.9**	0.76	0.35	**0.9**	0.53	0.53	0.35	0.53	0.61	0.69	0.35
Bitcoin	**1**	0.75	0.75	**1**	**1**	0.75	0.5	0.5	0.25	0.75	0.5
Construction	**0.66**	**0.66**	0.33	**0.66**	**0.66**	**0.66**	0	0.33	0.33	0.33	0.33
Scanline	**1**	0.81	0.81	**1**	**1**	0.81	0.36	0.54	0.36	0.72	0.45

Table 5.1. True Positive Rate

False Positive Rate											
	Clustering			Cos. Sim.			Chow	Quandt	Cusum	Xtbreak	MP
Rounding	1	2	3	1	2	3					
Synthetic 1	**0**	**0**	**0**	**0**	**0**	**0**	0.75	0.4	0.3	**0**	**0**
Synthetic 2	**0.1**	**0.1**	0.3	**0.1**	**0.1**	0.3	0.8	0.5	0.6	0.6	0.4
Covid	0.5	0.5	0.5	0.5	0.5	0.5	0.75	0.5	0.5	**0**	0.5
Apple	**0.23**	0.35	0.35	**0.23**	**0.23**	0.35	0.76	0.53	0.53	0.46	0.76
Bitcoin	**0.25**	**0.25**	0.5	**0.25**	**0.25**	**0.25**	1	1	0.5	0.5	1
Construction	**0.33**	**0.33**	0.66	**0.33**	**0.33**	**0.33**	1	0.66	0.66	0.33	1
Scanline	**0**	**0**	0.19	**0**	0.19	**0**	0.81	0.81	0.72	0.36	0.72

Table 5.2. False Positive Rate

F1 Score											
	Clustering			Cos. Sim.			Chow	Quandt	Cusum	Xtbreak	MP
Rounding	1	2	3	1	2	3					
Synthetic 1	**1**	**1**	0.83	**1**	**1**	0.83	0.12	0.4	0.3	0.4	0.12
Synthetic 2	**0.72**	**0.72**	0.4	**0.72**	0.4	0.4	0.1	0.1	0.1	0.3	0.1
Covid	0.3	0.3	0.3	0.3	0.3	0.3	0.1	0.2	0.2	**1**	0.2
Apple	**0.45**	**0.45**	0.2	**0.45**	0.2	0.2	0.1	0.1	0.2	0.2	0.1
Bitcoin	**0.8**	0.6	0.6	**0.8**	0.6	0.6	0.2	0.2	0.2	0.4	0.2
Construction	**0.4**	**0.4**	0.2	**0.4**	0.2	0.2	0	0.2	0.1	0.1	0.1
Scanline	1	0.8	0.6	1	1	0.6	0.1	0.15	0.1	0.4	0.15

Table 5.3. F1 Score

Interpretation of the Results

From the True Positive Rate table it can be seen that proposed methods of Clustering and Cosine Similarity produce a very high TPR. This in general means that if there exists a structural break then highly likely the algorithm will find it.

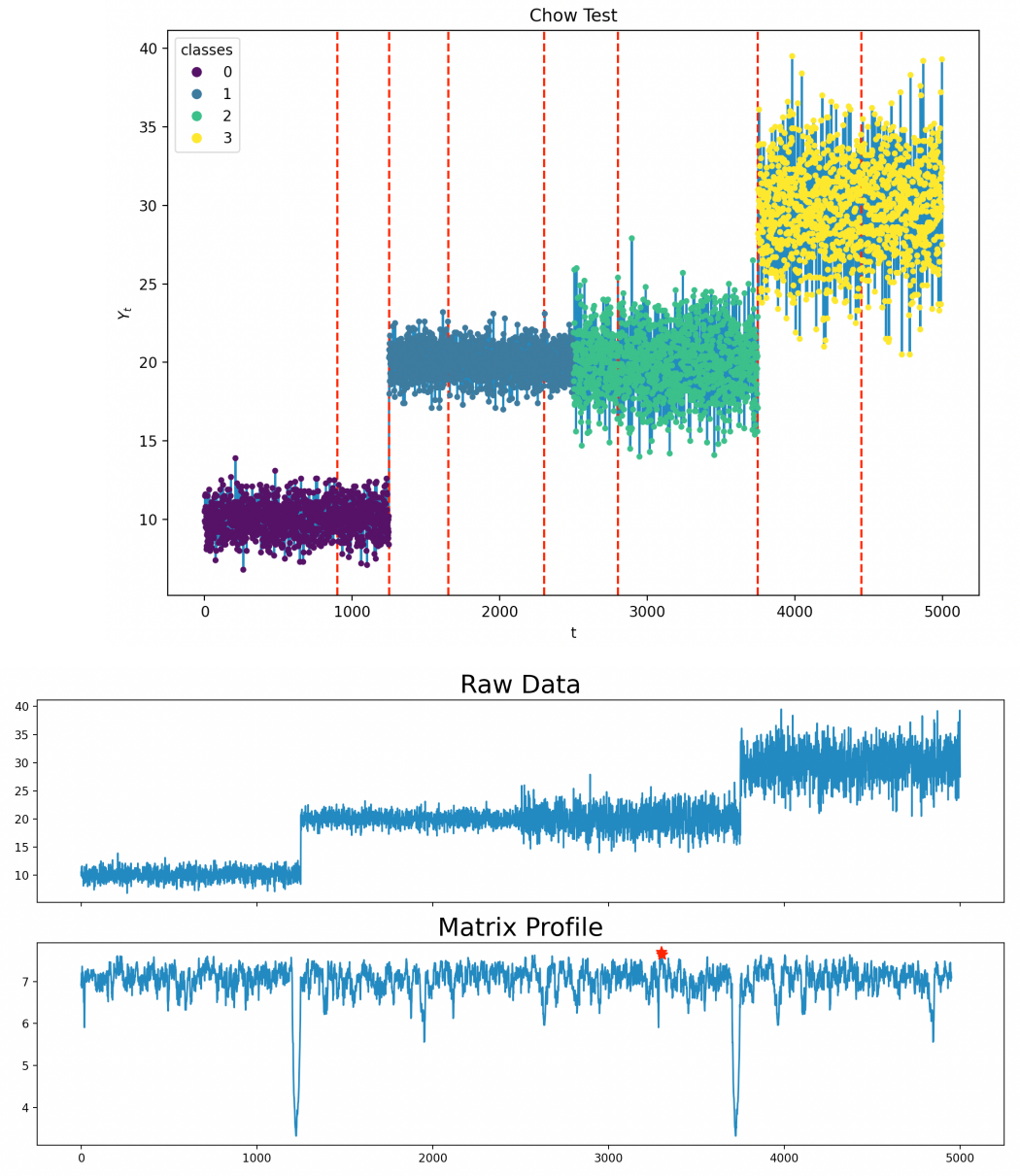


Figure 6.1 Chow Test predictions (red) and Matrix Profile

On the Fig. 6.1 it can be seen that Chow test seem to produce too many false positive predictions, hence in the FPR table he has almost the lowest scores. On the

other hand, Matrix Profile seem to produce a low number of false alarms. On the Fig. 6 we can see that it, along with my models, predicted only two structural breaks, while omitting the one in the middle.

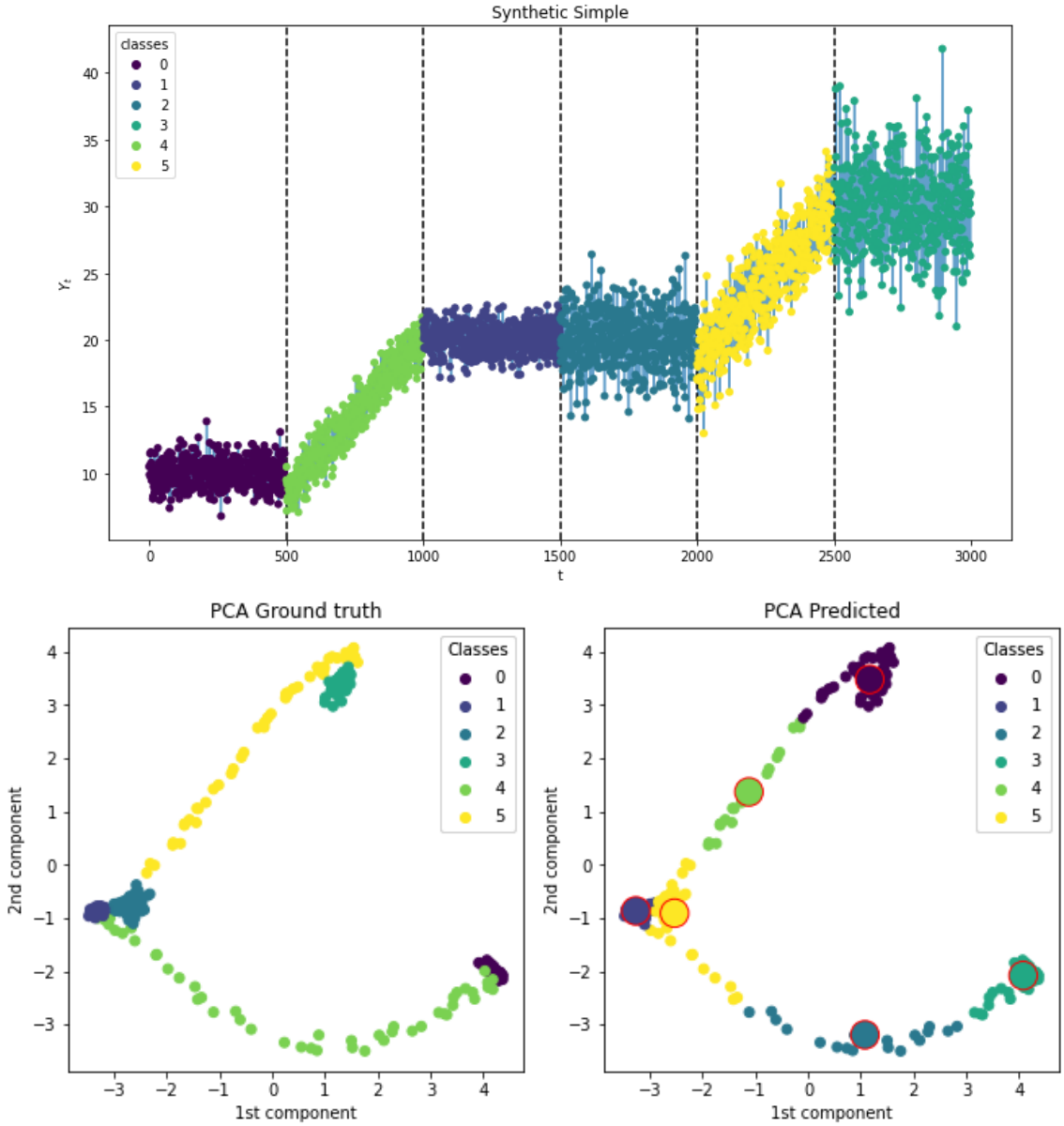


Figure 6.2 PCA of embeddings for Synthetic 2 dataset

Fig. 6.2 demonstrates the performance of Clustering method on the Synthetic 2 dataset. As we can see there are two additional segments added to smoother the change. From the PCA plots we can deduce that the method managed to correctly

capture the number of clusters, however, seem to misclassify some of the segments, which resulted in poor performance.

Overall, the proposed methods constantly outperformed another benchmark, showing a greater true positive rate and lower false positive rate. Thus, the model is sustainable to noisy data and outliers, however, requires a lot of tokens to build sufficient embeddings.

Conclusion

From this paper, I can conclude that the proposed method of utilization time-series vector embeddings is a functional tool for unsupervised structural break's detection. What is more, suggested algorithm includes two separate approaches, which both make use of embeddings in different ways and produce significant results. In addition, this method was tested on different dataset and proved to outperform well-known benchmark methods, while not requiring for any specific properties of time-series.

In addition, this method utilizes modern libraries for constructing and operating vector embeddings, which results in a fast execution speed along with a small memory load. This enables the researchers to run the methods on a large datasets in a reasonable time. One possible improvement could be the use of cumulative sum approach while comparing the cosine similarities of segments. Using the approach any merges or subdivisions of segments can be quickly recalculated in the constant time.

Finally, a topic of hyperparameter tuning was not discussed in detail. There are number of different parameters in both approaches, for example window size and number of decimal points. Several methods can be explored to optimize for such parameters opposed to a current grid search.

References

1. Testing and Estimating Structural Breaks in Time Series, Jan Ditzen, Yiannis Karavias
2. The Chow Test, 1960
3. Matrix Profile: All pairs similarity joins, 2016
4. Signal2Vec, Christoforos Nalmpantis, 2019
5. CBOW vs Skip-Gram
6. Sentence Embeddings, Lyndon White, 2015
7. Faiss library
8. Detection of multiple structural breaks in multivariate time series, Philip Preuss, 2013
9. An Evaluation of Change Point Detection Algorithms, Gerrit J., 2020
10. My GitHub repo: https://github.com/ovyan/thesis