Segment-wise Shape-based Symbolic Time Series Representation

Herbert Teun Kruitbosch, Ioannis Giotis and Michael Biehl

Abstract—This paper introduces a symbolic time series representation using monotonic sub-sequences and bottom up segmentation. This representation allows for machine learning techniques to analyse and detect time series patterns. The representation minimizes the square error between the segments and their approximation by monotonic functions. The function that *fits* best is a symbol which classifies the segment. The representation can robustly classify the direction of a segment and is scale invariant with respect to the time and value dimensions.
This paper describes two experiments. The first shows how accurately the monotonic functions are able to discriminate between different segments. The second tests how well the segmentation technique recognizes segments and classifies them with correct symbols. Finally this paper illustrates the proposed representation on data from the IJkdijk data set.

Index Terms—time series, classification, linear least square fit, linear least square estimation, segment classification, maximum likelihood estimation, maximum likelihood classification, bottom up segmentation, IJkdijk

1 Introduction

A time series is a sequence of measurements taken at successive moments in time. For example, the IJkdijk project [1] measures temperature, water pressure and other physical quantities occurring on the Dutch dikes, typically at equidistant moments in time and at different locations on and inside the dikes. The goal of the project is that all Dutch water management agencies use dike monitoring by 2014.

During distinctive periods in time, weather and dike conditions lead to characteristic behaviour in the measurements. Locating meaningful segments, which identify such periods, may help detect anomalies or provide useful input for the successful administration of dikes under extreme conditions. Detecting such conditions allow water management agencies to estimate when maintenance to a dike is needed and hence prevent untimely or unnecessary efforts.

However, time series often contain a large number of measurements, possibly too many to store in memory or do calculations within a reasonable amount of time. Several reduced representations which split a time series into pieces of equal or variable size and represent each piece by a relatively small feature vector or symbol have been proposed. Amongst others, two important *reduced* time series representations are: Piece-wise Linear Representation, PLR [6] and the Symbolic Aggregate Approximation, SAX [11]. PLR segments a time series into pieces of possibly different sizes, i.e. segments, and approximates them with linear curves. Other higher order polynomial representations are SwiftSeg [3] and the shape space representation [4]. SAX splits a time series into segments of an equal, predefined size and assigns a symbol to each piece, depending on the range in which the mean value of the segment lies. These representations have been used by various researchers to support clustering [6, 11, 8], classification [11, 8], association rule mining [11, 10], query by content [11] and anomaly detection [6, 12, 14] of time series data.

The SAX representation is effective, because a time series pattern will robustly yield a symbolic string. However, the SAX representation is not scale-invariant with respect to time and does not store information about the shape of a segment. Moreover, the SAX representation requires a parameter to state the size of an individual segment. An accurate value of this parameter may rely on local behaviour of a time series; certain sizes may be too big to see small patterns or too small to see big patterns.

This paper will introduce a piece-wise representation which tries to find monotonically increasing or decreasing pieces, which may differ in size. Each piece is represented by a symbol, which indicates it's direction and it's curvature. Hence our representation is symbolic like SAX, but instead of assigning symbols according to the mean value, the symbols are assigned according to shape. On the other hand our representation segments like PLR but tries to find monotonically increasing or decreasing pieces instead of linear segments. The monotonically increasing or decreasing segments are found using the bottom up segmentation technique. Bottom up segmentation starts off with a fine grained segmentation and merges adjacent segments until some optimal between the amount of segments and the error is met [6]. In the following we will introduce the aforementioned representation and test it using synthetic signals with additive white Gaussian noise.

This paper is organized in the following way. Section 2 gives a brief overview of techniques used by the proposed representation. Section 3 discusses how these are used to create a symbolic monotonic-segment-wise representation, section 4 shows how well our representation performs on synthetic signals and an example of an IJkdijk time series represented with our technique. Finally, section 5 discusses the conclusions and directions for future work.

2 Background and related work

Our symbolic representation uses linear least square estimation of a segment to approximate it and classify it as a certain shape. Therefore this section will first discuss linear least square estimation and a relevant Bayesian consideration regarding classification. Next we discuss the use of bottom up segmentation to segment a time series using linear least square approximations of the segments. Finally, the use of the GAP statistic [13] to determine the number of segments is given.

2.1 Linear least squares estimation

In order to approximate and assign a symbol to a segment, we use linear least square estimation. A linear least square estimation of a segment can be justified if we assume that a segment $\mathbf{s} = s_1, \ldots, s_N$, can be described as $s_n = \sum_{m=1}^{M} \theta_m f_m(n) + w_n$, where $f_1, \ldots, f_M$ is a set of basis functions. Here $w_n \sim \mathcal{N}(0, \sigma^2)$ denotes white Gaussian noise with variance σ^2, which is identically distributed and independent along the samples $n = 1, \ldots, N$ (i.i.d.). A (linear) least square estimation determines the unknown parameters $\theta_1, \ldots, \theta_M$, such that the *error* between s_n and $\sum_{m=1}^{M} \theta_m f_m(n)$ is minimal, e.g.

$$\hat{\theta}_{LSE} = \mathrm{argmin}_\theta \sum_{n=1}^{N} \left(s_n - \sum_{m=1}^{M} \theta_m f_m(n) \right)^2 . \qquad (1)$$

In our case, we only consider linear least square estimations based on one monotonically increasing basis function f and one basis func-

- *Herbert Teun Kruitbosch is a Msc. student at the university of Groningen, E-mail: herbertkruitbosch@gmail.com,*
- *Ioannis Giotis, second supervisor, Target Holding B.V. Groningen, I.E.Giotis@rug.nl,*
- *prof. dr. M. (Michael) Biehl, first supervisor, Professor University of Groningen, m.biehl@rug.nl.*

Manuscript received 20 December 2013 as a part of the 15EC internship for the MSc Computing Science of Herbert Teun Kruitbosch.

tion which is constant. This way the approximation $\theta_0 + \theta f(x)$ will always be monotonically increasing or decreasing. On the other hand, using two monotonic functions may result in a non-monotonic approximation. The three linear combinations $y_i = \theta_0 + \theta_1 f_1(x_i) + \theta_2 f_2(x_i)$ for $i = 1,2,3$ are based on two linearly independent, non-constant basis functions f_1 and f_2. We state that $x_1 < x_2 < x_3$ and that θ_0, θ_1 and θ_2 can always be found such that $y_1 < y_2 > y_3$. One can confirm this by solving for some $\varepsilon \neq 0$:

$$\begin{pmatrix} 1 & f_1(x_1) & f_2(x_1) \\ 1 & f_1(x_2) & f_2(x_2) \\ 1 & f_1(x_3) & f_2(x_3) \end{pmatrix} \begin{pmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{pmatrix} = \begin{pmatrix} \varepsilon \\ 0 \\ \varepsilon \end{pmatrix}.$$

Note that the matrix is non-singular since f_1 and f_2 are linearly independent and not constant. Also note that $\varepsilon, 0, \varepsilon$ is nor increasing nor decreasing. Finally note that this principle can be extended to more basis functions by setting their coefficients to 0 and to more samples by appending them. This proves the next theorem:

Theorem 2.1. *A linear combination $\sum_{i=1}^{M} \theta_i f_i(\mathbf{x})$ is monotonic for any $\theta = \theta_1 \ldots \theta_N$ **iff** the basis $f_1(\mathbf{x}),\ldots,f_M(\mathbf{x})$ is equivalent to some basis $g_1(\mathbf{x}), g_2(\mathbf{x})$ where $g_1(x) = constant$ and $g_2(\mathbf{x})$ is monotonic. Here we used these notations $\mathbf{x} = x_1,\ldots,x_N$ and $h(\mathbf{x}) = h(x_1),\ldots,h(x_N)$ for vectors and functions applied to vectors.*

This states that only bases of the form $1, f(x)$, with $f(\mathbf{x})$ monotonic, will never allow for non-monotonic approximations. Note that other forms of approximations, which are not linear bases, may also disallow non-monotonic approximations, for example $\alpha + \exp(\theta x)$. However non-linear least squares approximations are often harder to calculate. This paper only considers approximations which are linear combinations of basis functions.

Since our representation will segment into monotonically increasing or decreasing segments, we will fit a segment to the monotonic basis $f(x)$ which has mean 0. Then we have that $\sum_{n=1}^{N} f(x_n) = \langle f(\mathbf{x}), \mathbf{1} \rangle = 0$ and hence $\mathbf{1}$ and $f(\mathbf{x})$ are orthogonal. In general a least square estimation would be solved by:

$$\hat{\theta}_{LSE} = \operatorname{argmin}_\theta \|\mathbf{A}\theta - \mathbf{s}\|_2^2 = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{s}, \text{[5]} \tag{2}$$

where:

$$\mathbf{A} = \begin{pmatrix} 1 & f(x_1) \\ \vdots & \vdots \\ 1 & f(x_N) \end{pmatrix}, \theta = \begin{pmatrix} \alpha \\ \theta \end{pmatrix}, \hat{\theta}_{LSE} = \begin{pmatrix} \hat{\alpha} \\ \hat{\theta} \end{pmatrix}, \mathbf{s} = \begin{pmatrix} s_1 \\ \vdots \\ s_N \end{pmatrix}.$$

Since $\mathbf{A}$ has orthogonal columns, $\mathbf{A}^T\mathbf{A}$ is diagonal and hence the estimation simplifies to

$$\hat{\alpha} = \frac{1}{N}\sum_{n=1}^{N} s_n,\ \hat{\theta} = \frac{\langle f(\mathbf{x}), \mathbf{s} \rangle}{\|f(\mathbf{x})\|_2^2}.$$

Like aforementioned, estimating α and θ this way requires the assumption of the model $s_n = \alpha + \theta f(x_n) + w_n$ with w_n i.i.d. white Gaussian noise.

2.2 Bayesian classification

Using the likelihood of different models can allow us to classify a segment as one of these models. If we consider a set of monotonically increasing functions $f_1,\ldots,f_M$ which all are linearly independent we can calculate the maximum likelihood possible for each model M_i : $s_n = \alpha + \theta f_i(x_n) + w_n$, where w_n is i.i.d. white Gaussian noise, using maximum likelihood estimation of α and θ. Given a model M_i, we can calculate the likelihood of some set of data points:

$$\begin{aligned} P(S|M_i) &= \prod_{n=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} \exp\frac{-(\alpha + \theta f_i(x_n) - s_n)^2}{2\sigma^2} \\ &= \frac{1}{(\sqrt{2\pi}\sigma)^N} \exp\frac{-\sum_{n=1}^{N}(\alpha + \theta f_i(x_n) - s_n)^2}{2\sigma^2}. \end{aligned}$$

In order to classify a segment $\mathbf{s}$, i.e. $cl : Segment \mapsto f_1,\ldots,f_M$, we need to find the model with the highest the posterior probability

$$P(f_i|\mathbf{s}) = \frac{P(\mathbf{s}|\alpha + \theta f_i(x))P(\alpha + \theta f_i(x))}{P(\mathbf{s})}.$$

Hence we can classify by maximizing the posterior probability:

$$cl(\mathbf{s}) = \operatorname{argmax}_{\alpha + \theta f_i} P(\mathbf{s}|\alpha + \theta f_i(x))P(\alpha + \theta f_i(x)).$$

The prior probability $P(\alpha + \theta f(x))$ is hard to determine, since $P(\alpha, \theta|f_i)$ and $P(f_i)$ are often unknown. Therefore, the classification model is simplified to maximum likelihood classification:

$$\begin{aligned} cl(\mathbf{s}) &= \operatorname{argmax}_{\alpha + \theta f_i} P(\mathbf{s}|\alpha + \theta f_i(x)) \\ &= \operatorname{argmax}_{\alpha + \theta f_i} - \sum_{n=1}^{N}(\alpha + \theta f_i(x_n) - s_n)^2. \end{aligned} \tag{3}$$

This introduces a problem, since some models may always be less likely than others. For example, let us consider $f_1(x) = 0$ and $f_2(x) = x$. Then $P(f_1|\mathbf{s}) \leq P(f_2|\mathbf{s})$ for all segments $\mathbf{s}$, since the least square estimation of θ for f_2 will always allow for at least the same squared error as f_1 has for $\theta = 0$. In order to tackle this problem we introduced a heuristic based on the signal to noise ratio that will be discussed in section 3.2.

Algorithm 1 Bottom up segmentation as defined in [6]

```
Require: Time series $T = (t_1,\ldots,t_n)$
    Maximum error $\varepsilon$
    Fitter $fit : \mathbb{R}^n \mapsto \mathbb{R}^n$
    Minimum segment size $minsize$
    Error function $error : \mathbb{R}^i \mapsto \mathbb{R}^i \mapsto \mathbb{R}$
    Minimum number of segments $minsegments$
Ensure: A set of breakpoints $B \subset [1\ldots n]$ defining the segmentation
    of $T$

    $B = [1, 1 + minsize, 1 + 2 \cdot minsize, \ldots, N + 1]$        ▷ Fine-grained
    segmentation
    $C = []$                                ▷ Empty list for storing the merge costs
    for $i = 1$ to $\#B - 2$ do
        $lower, upper = C[i], C[i+2] - 1$
        $Sub = t_{lower},\ldots,t_{upper}$                    ▷ A subsequence of $T$
        $C[i] = error(Sub, fit(Sub))$
    end for
    while $min(C) < \varepsilon$ and $\#B > minsegments$ do
        $idx = argmin(i, C[i])$
        delete $B[i+1]$, $C[i+1]$
        $l_1, u_1, l_2, u_2 = C[i-1], C[i] - 1, C[i], C[i+2] - 1$
        $Sub_1 = t_{l_1},\ldots,t_{u_1}$
        $Sub_2 = t_{l_2},\ldots,t_{u_2}$
        $C[i-1] = error(Sub_1, fit(Sub_1))$
        $C[i] = error(Sub_2, fit(Sub_2))$
    end while
```

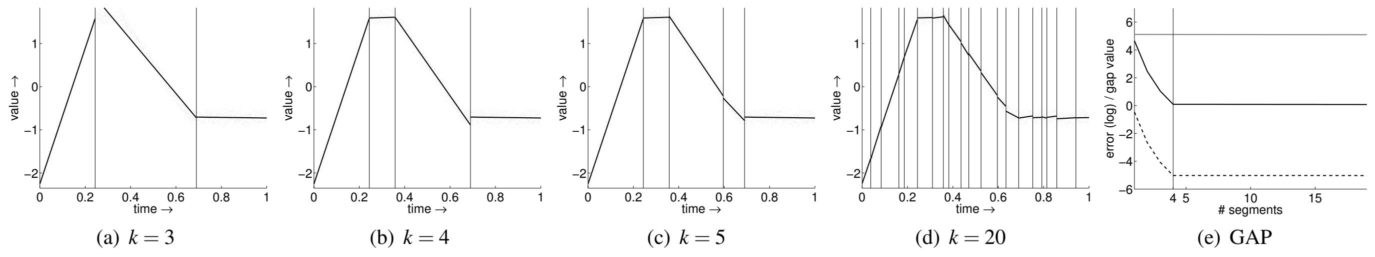

(a) $k=3$ (b) $k=4$ (c) $k=5$ (d) $k=20$ (e) GAP

Fig 1. Bottom up segmentation of a 4-piece piece-wise linear time series $\mathbf{t}$ with additive i.i.d. white Gaussian noise. The first 4 figures show segmentations for k segments, the solid lines are fits of the segments and the points are show the spread of the actual time series. The right-most figure shows the GAP statistic as discussed in section 2.4. For different numbers of segments $\varepsilon_{segmentation}$ of $\mathbf{t}$ and $\mathbf{u}$ are plot.

(e):
━ $\varepsilon_{segmentation}$ of $\mathbf{t}$
━ $\varepsilon_{segmentation}$ of $\mathbf{u}$
╍ $gap(k|\mathbf{t})$

2.3 Bottom up Segmentation

Bottom up segmentation is one of many segmentation techniques. Other well-known techniques are sliding window segmentation and top down segmentation. None of the algorithms consider all possible segmentations and hence have to deal with avoiding local optima. In general Bottom up outperforms other methods [7] and allows for use of the GAP statistic to determine the *optimal* number of segments. Section 3.4 will also discuss some considerations regarding the use of bottom up segmentation for our monotonic representation. The bottom up algorithm discussed in this section returns the *breakpoints* of the segments, instead of the segments themselves. A breakpoint is the first element of a segment.

Determining the *optimal* number of segments is an ill-posed problem; more segments result in a lower error. However, it is not realistic that all segments only contain a few samples. This trade-off between number of segments and error is solved by a heuristic introduced by Tibshirani et al. [13]: the GAP statistic. The statistic is based on an *elbow* between the number of segments and the amount of error and will be discussed in section 2.4.

The bottom up approach is shown in algorithm 1, the algorithm also uses a fitting function, called fit, this could be a linear least square estimation using the monotonic basis (see section 2.1) $f(x) = x$. In this case we would try to determine a piece-wise linear approximation. Section 3.1 will discuss the fitting function used by our representation.

The bottom up approach starts with a fine grained segmentation, where all segments have small and equal sizes. Then the algorithm calculates the error for each concatenation of two adjacent segments. The pair of segments in such a concatenation with the minimum error is merged to one segment. This is done iteratively until either the error exceeds some pre-defined threshold or the number of segments has reached some pre-defined bound. See algorithm 1. Finally, figure 2.2 shows a segmentation using bottom up using different numbers of segments.

2.4 GAP statistic

In order to determine the optimal number of segments, we use a similar technique as the GAP statistic as proposed in [13]. We must first define the error of a segmentation as the mean squared error:

$$\varepsilon_{segment}(\mathbf{s}) = \|\mathbf{s} - fit(\mathbf{s})\|_2^2, \;\; \varepsilon_{segmentation}(S) = \sum_{\mathbf{s}\in S} \varepsilon_{segment}(\mathbf{s}),$$

where fit is a function which approximates a segment $\mathbf{s}$ and S is a set of segments. This error is different from the mean squared error used in [13] since it sums the squared distances between a segment $\mathbf{s}$ and the fit of $\mathbf{s}$ instead of the squared distances between $\mathbf{s}$ and the mean of $\mathbf{s}$. This is justified, since instead of representing a cluster by its mean we represent a segment by its fit. The GAP statistic is based on the error of a segmentation of a time series, and the expected error of a uniform distribution in the same range. Therefore we define a reference signal $\mathbf{u} = u_1, \ldots, u_N$ where $u_i \sim \mathscr{U}(0,1)$: uniformly randomly distributed between 0 and 1. We also define a time series $\mathbf{t} = t_1, \ldots, t_N$ which is normalized to have values between 0 and 1. According to [13] the GAP statistic is:

$$gap(k|\mathbf{t}) = \log\left(\varepsilon_{segmentation}(S_{\mathbf{t},k})\right) - \log\left(\varepsilon_{segmentation}(S_{\mathbf{u},k})\right),$$

where $S_{\mathbf{t},k}$ and $S_{\mathbf{u},k}$ are segmentations of $\mathbf{t}$ and $\mathbf{u}$ of k segments using algorithm 1. We estimated $\varepsilon_{segmentation}(S_{U,k})$ by taking the mean of $Z = 10$ instances of $\mathbf{u}$, $\hat{\mathbf{u}} = \hat{\mathbf{u}}_1, \ldots, \hat{\mathbf{u}}_Z$, and corrected with the standard deviation:

$$\log \varepsilon_{segmentation}(S_{\mathbf{u},k}) \approx \mathsf{mean}(\log \hat{\mathbf{u}}) - \sqrt{1 + \frac{1}{Z}}\frac{Z-1}{Z}\mathsf{std\ dev}(\log \hat{\mathbf{u}})$$

The *elbow* of $gap(k|\mathbf{t})$ shown in figure 2.2e at $k_{elbow} = 4$ is the point where adding more segments will only result in fitting to noise. This elbow will typically not be located at $\mathsf{argmin}_k gap(k|\mathbf{y})$, because the values of $gap(k|\mathbf{t})$ for $k > k_{elbow}$ may be slightly smaller than $gap(k_{elbow}|\mathbf{y})$ due to fitting to noise or numerical instability.

We selected the lowest k_{elbow} such that for all $k \geq k_{elbow}$ the decrease between the gap values, $gap(k|\mathbf{t}) - gap(k+1|\mathbf{t})$, is lower than some threshold, in our case 0.05. Other estimates could be the k with maximal second order derivative or maximal curvature. We used the threshold on the first order derivative, since there may be multiple local maxima in the second order derivative which are all candidates for k_{elbow}. However, it must be noted that the threshold on the first derivative is unstable. In the conclusion we point to an article which may suggest a better method to estimate k_{elbow}.

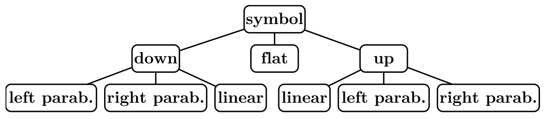
3 Method

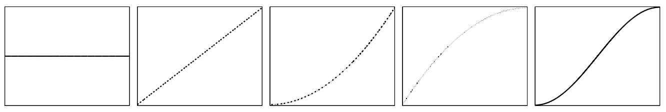
This section will describe how our reduced time series representation is formed on the basis of bottom up segmentation and linear least square estimation. First we will describe how linear least square estimation can fit different kinds of monotonic shapes on a segment. Such a *fit* results in an approximation and a shape assignment for the segment. The segment approximation is used by the bottom up algorithm to approximate the optimal segmentation and finally the shape assignment is used to create a symbolic string. This process looks like this:

$$\begin{tikzcd} \boxed{\text{Time Series}} \arrow[r, "\text{bottom up segmentation}"'] & \boxed{\text{Segments}} \arrow[r, "\text{shape assignment}"'] & \boxed{\text{Symbols}} \end{tikzcd}$$

3.1 Assigning Shapes to Segments

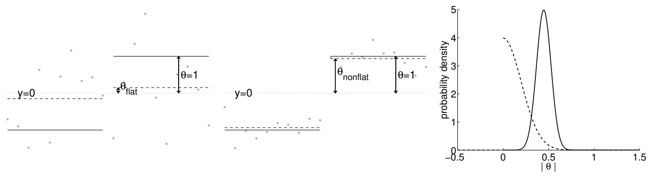
In order to detect non-monotonic structures in time series, this paper focusses on finding monotonic segments of a time series. This way a concatenation of monotonic segments defines a non-monotonic structure. The idea is that we classify a segment as one of five shapes, shown in figure 2. Except for the first shape, each shape is monotonically increasing. However, we also classify whether the fit of such a shape goes up or down. This results in a classification tree for shapes:





$f_0 = 0$ $f_1 = x$ $f_2 = x^2$ $f_3 = -(1-x)^2$ $f_4 = -\cos(\pi x)$

Fig. 2. Five monotonic shapes. They are used to classify time series segments. The shapes are functions on the domain $[0,1]$. With respect to the classification tree of section 3.1 all shapes also have a additive inverse which goes down. Moreover, the S-shape of (e) is not shown in the tree, it will be discussed in section 3.3.



(a) $\bar{s}_n = \overline{w}_n$ (b) $\tilde{s}_n = \theta f(x) + \tilde{w}_n$ (c) $|\theta|$ distribution

Fig 3. Respectively, (a) and (b) show the fitting of the function f on the data of a flat model $\bar{s}_n = \overline{w}_n$ and of a f model $\tilde{s}_n = \theta f(x) + \tilde{w}_n$ (see section 3.2). Both $\overline{w}_n$ and $\tilde{w}_n$ denote i.i.d. white Gaussian noise. In (c): the probability distribution of θ for the flat model and the f model for $\theta = 1$, $\sigma = 1$ and $N = 5$.

(a), (b):
● data
— f
- - fit of f
(c):
— flat shape
- - non-flat

Here a left parabola has the extreme value left and the right parabola has the extreme value right. We can model all these shapes using three basis functions, namely $f_1(x) = x$ (linear), $f_2(x) = x^2$ (left parabola) and $f_3(x) = -(1-x)^2$ (right parabola) sampled within the domain $[0,1]$. Then assume a segment $\mathbf{s} = s_1, \ldots, s_N$ which is normalized to $\mu = 0$ and $\sigma = 1$, and equidistant $x_1, \ldots, x_N$ within $[0,1]$. Then the sign of θ in $\alpha + \theta f_i(x)$ defines whether the shape goes up, goes down or is flat for $\theta \approx 0$.

Using the 3 models, we can classify a segment using equation (3). Summarized this means we take the least square estimated θ_i, where i is an index, for all models:

$$\hat{\theta}_i = \frac{\sum_{n=1}^{N} f_i(x_n) \cdot s_n}{\sum_{n=1}^{N} f_i(x_n)^2} \qquad \text{Maximum likelihood estimated } \theta^i$$

Furthermore, the log-likelihood that $s_n = \theta_i f_i(x_n) + w_n$ for all $n = 1, \ldots, N$ and some θ_i is:

$$\log L(f_i|\mathbf{s}) = -\sum_{n=1}^{N} (\theta_i f_i(x_n) - s_N)^2 \qquad \text{log likelihood of } f_i$$

In algorithm 2 the first statement of the for-loop calculates the least square fits of $\alpha + \theta_i f_i$ and for $i = 1,2,3,4$, the value of c is ignored. The second statement of the for-loop uses θ_i to calculate the log-likelihood $\mathscr{L}_i$ for $i = 1,2,3,4$. The statement $i = \text{argmax}_i \mathscr{L}_i$ selects the $\alpha + \theta_i f_i$ which has the maximum likelihood. The if-statement then determines, depending on the sign of θ_i, whether the segment goes up or down. It can, however, also decide that the segment is flat. This is discussed in the next section. The final two statements output the actual symbol of the segment and the approximation of the segment.

3.2 Detecting flat segments

In order to classify the flat shape of figure 2a, we could fit to $f_0(x) = 0$. However, for $\theta_i = 0$, each of f_1, f_2, f_3 will have the same likelihood as f_0. Therefore a maximum likelihood estimated θ_i will always result in a larger or equal likelihood of f_1, f_2, f_3 than of f_0. Hence the maximum likelihood classification does not allow to distinguish between f_0 and any of f_1, f_2, f_3. This could be solved by adding a prior, like stated in section 2.2. However $P(\theta|f_i)$ and $P(f_i)$ would need to be determined, which is not always possible.

Therefore we introduce a heuristic based on the signal to noise ratio to classify flat segments. Consider a noisy, flat segment $\bar{s} = s_1, \ldots, s_N$, $\bar{s}_n = \overline{w}_n$ ($\overline{w}_n$ is i.i.d. white Gaussian noise) for which we want to determine whether it belongs to the flat shape or the f shape. The f shape is the set of segments defined by the model $s_n = \theta f(x) + w_n$. We consider a simple case where f has half of the points equal or close to it's minimum -1 and the other half equal to or close to it's maximum 1.

Algorithm 2 Classification and approximation of a segmented time series

```
Require: Segment $S = (s_1, \ldots, s_N)$
    Threshold $T$ mentioned in section 3.2
Ensure: Returns a classification $C$ of $S$ as a $f_0$, $f_1$, $f_2$ and $f_3$ and
    whether the $S$ goes up or down. Also returns an approximation $A = (a_1, \ldots, a_N)$ of $S$.

  $\mu = mean(S)$; $\sigma = std(S)$                ▷ Mean and Standard Deviation
  normalize $S$ to have standard deviation 1 and mean 0
  let $x_1, \ldots, x_N$ be equidistant distributed on the domain $[0,1]$
  $f_4 = \cos(\pi x)$                                  ▷ See section 3.3
  for $i \in 1,2,3,4$ do
      $\theta_i = \frac{\sum_{n=1}^{N} f_i(x_n) \cdot s_n}{\sum_{n=1}^{N} f_i^2(x_n)}$          ▷ Linear least square fit of $f_i$
      $\mathscr{L}_i = \sum_{n=1}^{N} (\theta_i f_i(x_n) - s_N)^2$          ▷ Likelihood of $f_i$
  end for
  $i = \text{argmax}_i \mathscr{L}_i$
  if $|\theta_i| < T$ then          ▷ Is segment increasing, decreasing or flat?
      $i = 0$; $dir = none$
      $\theta_0 = 0$; $f_0(x) = 0$
  else if $\theta_i < 0$ then
      $dir = down$
  else
      $dir = up$
  end if
  $C = (f_i, dir)$                                  ▷ Symbol of the segment
  Set $A$ such that $a_n = \mu + \sigma \theta_i f_i(x_n)$          ▷ Approximation of the
  segment
```

Then $Var(f) = 1$. Such a function and a signal like $\bar{s}$ are shown in figure 2.4a.

Assume that $\overline{w}_n \sim \mathscr{N}(0, \sigma^2)$. Now we normalize $\bar{s}$ to have variance 1: $Var(\bar{s}_n) = 1$. Due to the shape of f, estimation of θ will result in the average of the left half's average and the right half's negative average. Therefore the value of the least square estimated $|\overline{\theta}|$ for f depends on the average value of $\bar{s}$. Then we have that

$$P(|\overline{\theta}|) = \begin{cases} 2\mathscr{N}(|\overline{\theta}|, 0, \frac{1}{N}) & \text{if } |\overline{\theta}| \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

i.e. the distribution of $|\overline{\theta}|$ is equal to the distribution of the absolute average value of $\bar{s}$.

Now consider a noisy, non-flat segment $\tilde{s}_n = \theta f(x) + \tilde{w}_n$ with $\tilde{w}_n \sim \mathscr{N}(0, \sigma^2)$ as shown in figure 2.4b. For this segment it holds that $Var(\tilde{s}_n) = \theta^2 + \sigma^2$. Hence if we normalize $\tilde{s}$ to variance 1, we get that $\tilde{w}_n \sim \mathscr{N}(0, \frac{\sigma^2}{\sqrt{\theta^2 + \sigma^2}})$. Furthermore the probability of a least square estimated $|\tilde{\theta}|$ for f becomes:

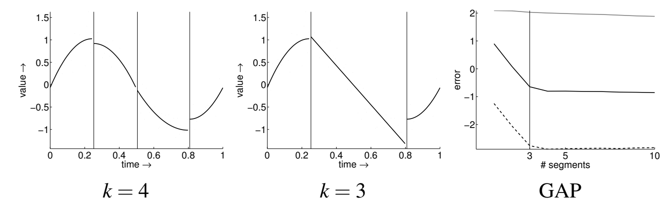


Fig. 4. Two segmentations for k segments of a sine wave. The segments in (a) shows that the middle two segments have curvature, when both middle segments are merged, like in (b), the curvature is not captured by the linear approximation. Figure (c) shows the GAP statistic like in figure 2.2. The optimal trade off at $k = 3$ shows that the linear approximation is used for the middle segment. The curves in (c) have equal patterns as figure 2.2e.

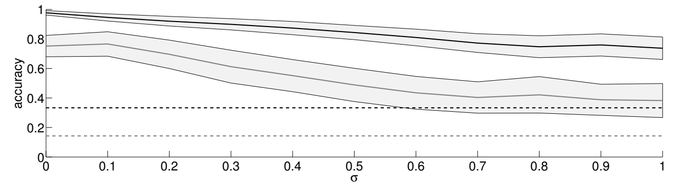


Fig 5. Segmentation and segment classification accuracy as described in section 4.3. The gray area is half of the standard deviation of the simulation. The MC (Monte-Carlo) classification accuracy is of a classifier which assigns a direction and shape uniformly random. In this case $T = 0.26$ and 228 simulations were run.

— direction accuracy
— shape accuracy
- - MC direc. accuracy
- - MC shape accuracy

$$P(|\tilde{\theta}|) = \begin{cases} \mathcal{N}(|\tilde{\theta}|, \frac{\theta}{\sqrt{\theta^2+\sigma^2}}, \frac{\sigma^2}{N\sqrt{\theta^2+\sigma^2}}) + \\ \quad \mathcal{N}(|\tilde{\theta}|, -\frac{\theta}{\sqrt{\theta^2+\sigma^2}}, \frac{\sigma^2}{N\sqrt{\theta^2+\sigma^2}}) & \text{if } |\overline{\theta}| \geq 0 \\ 0 & \text{otherwise} \end{cases} .$$

Notice that the normal distribution at the right part of the $+$ is almost equal to 0. Also notice that $\frac{\theta}{\sqrt{\theta^2+\sigma^2}}$ is θ normalized. This shows that the variance of θ for the flat shape and the f shape depends on the number of samples by a factor $\frac{1}{N}$ and to the signal to noise ratio $\frac{\sigma^2}{\sqrt{\theta^2+\sigma^2}}$ in case of a non-flat signal. Moreover, if the amount of noise increases, the means of both distributions move closer to each other; decreasing the discriminative power.

Figure 2.4c shows an example of both distributions. This figure suggests a threshold at approximately $|\theta| = 0.26$, such that $|\theta| < 0.26$ would classify as flat and $|\theta| \geq 0.26$ would classify as not flat. Such a threshold T is used in the classification algorithm as shown in algorithm 2. This analysis of the threshold T assumes that f has the block shape as shown in figure 2.4a. Assuming other shapes of f makes it harder to determine the distribution of $|\hat{\theta}|$, because the variance $Var(f(x) + w_x)$ is not necessarily $Var(f(x)) + Var(w_x)$.

3.3 Detecting S-shaped segments

The shapes in figure 2 are all monotonic, however a time series may have a part consisting of two adjacent monotonically increasing or two adjacent monotonically decreasing segments. Merging them may not always yield the desired result. Consider the time series of figure 4 which is segmented. Figure 4a has four segments and figure figure 4b is created by merging the two middle segments. The error curve in figure 4c shows that merging the middle two segments does not introduce much error and that the GAP statistic *suggests* it. The linear segment in figure 4b, however, does not capture the curvature of the S shape that is actually there: the parabola-like curvature is lost.

In order to prevent this information loss, we introduce an S-shape, as shown in figure 2. For simplicity, we model the S-shape by a cosine. In the case that the middle two segments are merged, the S-shape will fit better and hence this segment will be classified as an S-shape. As a post processing step, each S-shape can be replaced by a left and a right parabola. This way the segmentation and segment classification still comply with the symbols given in section 3.1 and we avoid redundancy in our set of symbols.

3.4 Bottom up segmentation

Bottom up segmentation is based on merging smaller segments until relatively few are left. The fine grained segments all are (approximately) monotonic because they are small. Since all models only allow for monotonic approximations, non-monotonic segments fit badly. Therefore two monotonic segments are not easily merged when they form a non-monotonic segment. Thus non-monotonic segments are not easily introduced.

In general the bottom up technique allows for getting stuck in local optima, since merging three segments may lead to an optimal segmentation, however merging any two of the adjacent segments within these three segments may be very expensive. We could call this situation an *error barrier*. Yet, any segmentation of a monotonic segment contains only monotonic *sub-segments*. If we assume that merging two monotonic segments to another monotonic segment is always cheap, then such an *error barrier* between the current and a *better* state never exists. This assumption *reflects* in our choice of shapes, see figure 2. The parabolas in figure 2, for example, can be approximated by a flat segment for the region near the extreme value and a linear segment near the skew region. The previous section about S-shapes also shows how bottom up can easily merge two monotonic segments to a new monotonic segment.

Finally, bottom up segmentation also allows to find segments of any size along the time dimension. Therefore the representation is invariant with respect to the amount of samples along this axis.

3.5 Creating the symbolic representation

Algorithm 3 summarizes the methods of section 3. The algorithm first segments using bottom up segmentation combined with the GAP statistic. Afterwards, each segment is classified using algorithm 2. The result is a symbolic string which can then be used by various machine learning algorithms.

Algorithm 3 Create a reduced symbolic representation

```
Require: Time series $T = (t_1, \ldots, t_n)$
  Threshold $t$ mentioned in section 3.2
Ensure: Returns a symbolic string $R$.

                        ▷ Threshold $t$ is implicitly used by algorithm 2
  $S = bottomup(T, \varepsilon = \infty,$
        # segments according to GAP,
        $fit$ according to algorithm 2)
  initialize $R$ as empty symbol list
  for Segment $s \in S$ do
    Classify $s$ according to algorithm 2, store in $sym$
    if $sym = (f_4, up)$ then
      append $(f_2, up)$ and $(f_3, up)$ to $R$
    else if $sym = (f_4, down)$ then
      append $(f_2, down)$ and $(f_3, down)$ to $R$
    else
      append $sym$ to $R$
    end if
  end for
```

4 Results

The accuracy of our symbolic representation is quantified using two measures, as described in section 4.1. These two measures are used

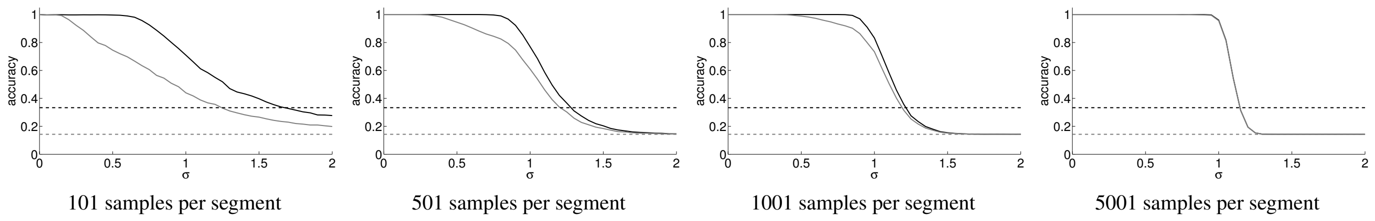


Fig 6. Accuracy of identifying the segment of one synthetic shape. The MC (Monte-Carlo) classification accuracy is of a classifier which assigns a direction and shape uniformly random.

━ direction accuracy
━ shape accuracy
╍ MC direction accuracy
╍ MC shape accuracy

to first determine how often the classification of a synthetic segment matches the label in section 4.2; this tests the accuracy of algorithm 2 and illustrates how well the classifier is able to discriminate between shapes. Section 4.3 illustrates the accuracy of algorithm 3; this shows how well the algorithm is able to find and identify monotonic segments which match the shapes of figure 2. Finally an example of real world data from the IJkdijk is segmented and classified according to algorithm 3.

4.1 Accuracy measures

We define two accuracy measures based on the *symbol tree* of section 3.1. The first measure quantifies the fraction of segments or samples for which the *direction* of the shape is correctly classified; we call this the direction accuracy. The direction indicates whether the shape goes up, goes down or is flat, but does not capture whether the *curvature* is linear, a left parabola or a right parabola. Hence we test how well the algorithm is capable of detecting linear segments, like the PLR described in [6].

However, our representation also allows to capture curvature. The second measure quantifies the fraction of segments or samples for which both the direction and the shape are correct; we call this the shape accuracy. In the presence of noise the detection of curvature may be impossible. These two measures allow to detect curvature and direction if the signal to noise ratio is adequate, on the other hand it also allows to only detect direction if the signal to noise ratio does not allow for curvature to be detected.

4.2 Classification accuracy

In order to measure the discriminating capability of our classification described by algorithm 2, we created 100 synthetic shapes for each direction of 4 of the template functions shown in figure 2. The cosine function is not tested, as it is similar to two parabolas. In order to test the classification algorithm (algorithm 2), we test the classification accuracy with respect to the amount of i.i.d. white Gaussian noise and the number of samples in a segment.

The results for different values of σ of the noise are shown in figure 3.2. The direction accuracy stays above 80% until at least $\sigma = 0.7$, depending on the number of samples. For 101 samples the shape accuracy drops below 80% at approximately $\sigma = 0.4$, however for 501 or more samples the shape accuracy drops below 80% at $\sigma = 0.9$ or later. For noise $\sigma \approx 1.2$ or higher, both accuracies drop very fast. This is related to the threshold T, see section 3.2. Due to the high amount of noise, an estimated θ for our shape $\alpha + \theta f(x)$ gets below T, and hence all segments are classified flat. We can note that for a given T both accuracies become invariant to the level of noise if there are enough samples.

4.3 Segmentation and classification accuracy

In order to further evaluate the accuracy and robustness of the proposed representation we measured the fraction of correctly classified samples instead of the fraction of correctly classified shapes.

At first 100 time series are created by concatenating monotonic segments of different time and value scale. The monotonic segments are based on the shapes in figure 2. Each time series has 4 segments and 701 samples. Then additive i.i.d. white Gaussian noise is added. Each sample in each synthetic time series is labelled by the shape of figure 2 its containing segment is created by. This concludes in a labelled set of synthetic time series. The segments are concatenated in such a way, that there are no gaps in the *y* direction.

Then each time series is segmented and each segment is classified according to algorithm 3. In a similar way as described in the previous paragraph, each sample is classified according to the classification of its containing segment. Note that this time, the segment was determined via bottom up segmentation and was not known via synthetic creation.

Finally the fraction of the samples where the classification and the label match, according to both measures, is calculated. The results for different noise levels are shown in figure 3.1. We can see that the direction accuracy is rather robust with respect to noise; the accuracy stays above 0.8 for up to noise $\sigma = 1$, whereas the shape accuracy scores lower.

4.4 IJkdijk data example

To conclude the results, some subsequences from the LiveDijk Eemshaven [2] measurements were taken and translated into symbolic strings. These subsequences are taken from two locations in an experimental dike, both measuring pH values. The sub-sequences of the time series were between the timestamps specified in table 1. The results are shown in figure 7.

subseq.	from (seconds since 1970)	to (seconds since 1970)
a,e	1259812794.065	1260612952.017
b,f	1263580103.098	1265176501.031
c,g	1276219556.058	1276529578.043
d,h	1296729988.044	1298360426.079

Table 1. Begin and end timestamps of the subsequences used in figure 7. All times lie between Januari 2010 and March 2011.

The plots show that indeed monotonic structures are found. In the signals shown in (e), (f) and (h) the patterns are *subtle* and due to the low levels of noise very visible. This illustrates that our representation is capable of finding small structures if noise levels are relatively small. On the other hand, figure (a) and (d) demonstrate that our representation is also capable of finding larger segments. This illustrates the scale invariance of our representation. Some segments are not entirely monotonic, and one could argue that they should be split. For example, the second segment in (c) trends to go up, but alternates between going up and down a bit. This behaviour, i.e. ignoring small patterns and only taking larger ones in account, is influenced by the estimation of the number of segments. Changing the trade-off between the amount of segments and the amount of error will change how the representation deals with such smaller, less defined monotonic segments.

5 CONCLUSION AND FUTURE WORK

This paper introduced a symbolic representation of a time series which is based on monotonic segments of possibly different sizes. The representation enables machine learning algorithms to efficiently find anomalies or structures in a time series based on the shape of monotonic sub-sequences. Our representation allows for classification at two levels: direction which is robust with respect to noise and curvature which allows to capture more subtle patterns, but is less robust with respect to noise. The representation is also scale invariant with respect to time, allowing to find both fine grained and large patterns. However, one would change the GAP statistic to allow for selecting smaller segments. Future work could define a hybrid representation, representing both smaller and larger encapsulating segments. For example considering a tree of segments, becoming more fine grained when closer to the leaves.

Future work should focus on similar shapes detecting capabilities of the representation. Clustering patterns generated by our representation of the IJkdijk data may give insight in the general behaviour of the time series and hence help define and determine anomalous sub-sequences. Another improvement would be to incorporate the faster sliding window bottom up (SWAB) segmentation, as introduced in [6]. Finally, the STOM method, proposed in [9], may improve the estimation of k_{elbow}.

References

[1] IJkdijk website http://www.ijkdijk.nl/en/ijkdijk.

[2] Livedijk Eemshaven website http://www.ijkdijk.nl/en/livedijken/livedijk-eemshaven.

[3] E. Fuchs, T. Gruber, J. Nitschke, and B. Sick. Online segmentation of time series based on polynomial least-squares approximations. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(12):2232–2245, 2010.

[4] E. Fuchs, T. Gruber, H. Pree, and B. Sick. Temporal data mining using shape space representations of time series. *Neurocomputing*, 74(1):379–393, 2010.

[5] C. F. Gauss. *Theory of the motion of the heavenly bodies moving about the sun in conic sections: a translation of Carl Frdr. Gauss" Theoria motus": With an appendix. By Ch. H. Davis*. Little, Brown and Comp., 1857.

[6] E. Keogh, S. Chu, D. Hart, and M. Pazzani. An online algorithm for segmenting time series. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 289–296. IEEE, 2001.

[7] E. Keogh, S. Chu, D. Hart, and M. Pazzani. Segmenting time series: A survey and novel approach. *Data mining in time series databases*, 57:1–22, 2004.

[8] E. J. Keogh and M. J. Pazzani. An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. In *KDD*, volume 98, pages 239–243, 1998.

[9] S. Kitayama and K. Yamazaki. Compromise point incorporating trade-off ratio in multi-objective optimization. *Applied Soft Computing*, 12(8):1959–1964, 2012.

[10] V. Lavrenko, M. Schmill, D. Lawrie, P. Ogilvie, D. Jensen, and J. Allan. Mining of concurrent text and time series. In *KDD-2000 Workshop on Text Mining*, pages 37–44. Citeseer, 2000.

[11] J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pages 2–11. ACM, 2003.

[12] C.-S. Perng, H. Wang, S. R. Zhang, and D. S. Parker. Landmarks: a new model for similarity-based pattern querying in time series databases. In *Data Engineering, 2000. Proceedings. 16th International Conference on*, pages 33–42. IEEE, 2000.

[13] R. Tibshirani, G. Walther, and T. Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423, 2001.

[14] C. Wang and X. Sean Wang. Supporting content-based searches on time series via approximation. In *Scientific and Statistical Database Management, 2000. Proceedings. 12th International Conference on*, pages 69–81. IEEE, 2000.

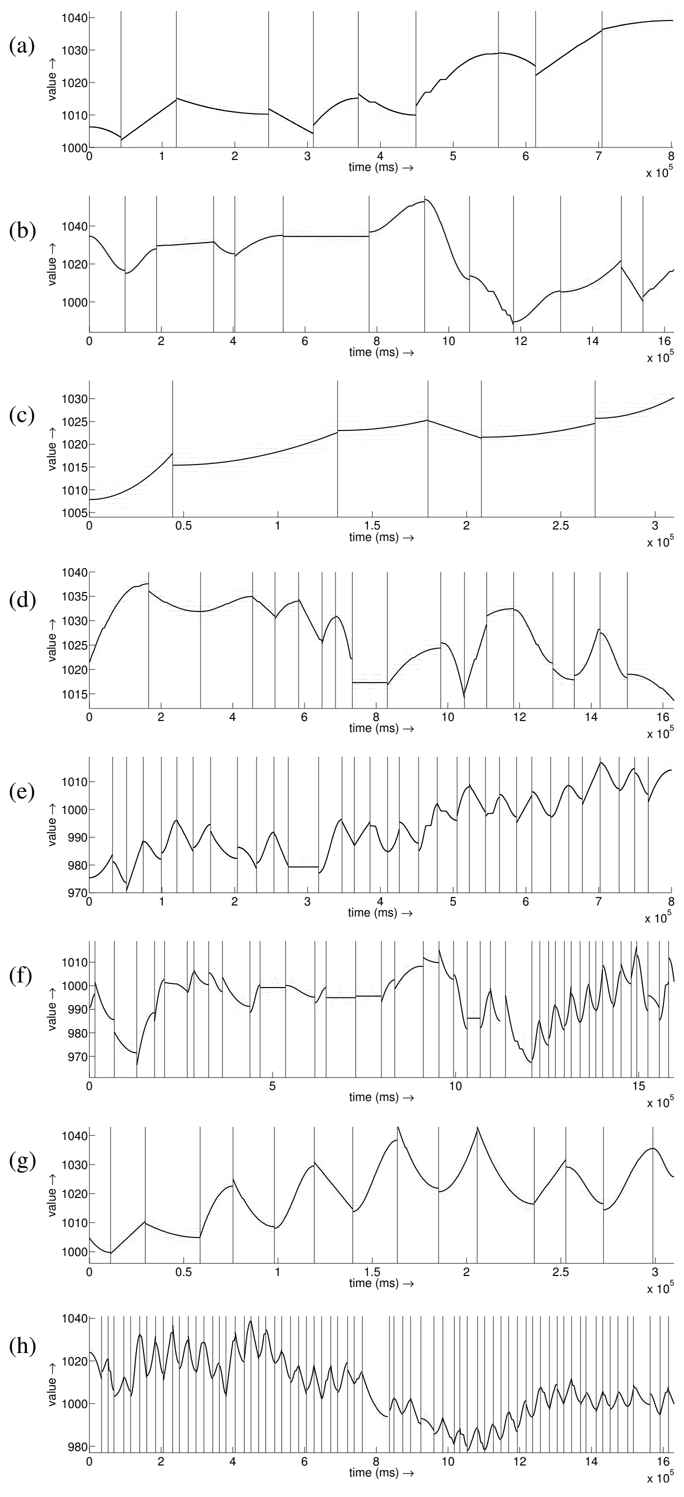


Fig. 7. Segmentation of sub-sequences taken from the IJkdijk data set. The data represents pH values, each row respectively depicts the segments shown in table 1. In this case the fine grained segmentation the bottom up algorithm starts with was 100 samples per segment to increase computational performance. The flat-threshold was set to $T = 0.26$. Each signal contains 5000 samples. The top 4 and bottom 4 signals come from different locations and hence different sensors in the Livedijk Eemshaven. The meaning of this data is explained in section 4.4. The timestamps between which the 8 time series are measured are shown in table 1