Time series segmentation for context recognition in mobile devices

Johan Himberg Kalle Korpiaho Heikki Mannila Johanna Tikanmäki
Hannu T.T. Toivonen

Nokia Research Center, Software Technology Laboratory
P.O. Box 407, FIN–00045 NOKIA GROUP, Finland
johan.himberg@nokia.com

Abstract

Recognizing the context of use is important in making mobile devices as simple to use as possible. Finding out what the user's situation is can help the device and underlying service in providing an adaptive and personalized user interface. The device can infer parts of the context of the user from sensor data: the mobile device can include sensors for acceleration, noise level, luminosity, humidity, etc. In this paper we consider context recognition by unsupervised segmentation of time series produced by sensors.

Dynamic programming can be used to find segments that minimize the intra-segment variances. While this method produces optimal solutions, it is too slow for long sequences of data. We present and analyze randomized variations of the algorithm. One of them, Global Iterative Replacement or GIR, gives approximately optimal results in a fraction of the time required by dynamic programming. We demonstrate the use of time series segmentation in context recognition for mobile phone applications.

1 Introduction

Succesful human communication is typically contextual. We discuss with each other in different ways depending on where we are, what time it is, who else is around, what has happened in the past, etc.: there is lots of context information that is implicitly being used in everyday life. Communication that is not aware of its context can be very cumbersome. The need for context-awareness is especially large in mobile communications, where the communication situations can vary a lot.

Information about the context of, say, a mobile phone can be used to improve the user interface. For example, if we know from the context information that the user is running, the font used in the display can be larger. Similarly, audio volume can be adjusted to compensate for higher levels of noise. Context awareness is currently studied in various aspects. Example of such studies include work on context sensitive applications, wearable computers and environmental audio signal processing, e.g., in [4, 5, 6, 9, 10, 14, 15, 17].

In this paper we discuss ways of achieving context-awareness in mobile devices such as mobile phone. A mobile device can infer useful context information from sensors for, e.g., acceleration, noise level, luminosity, and humidity. In general, figuring out what the user is actually doing is difficult, and we tackle one important subproblem. Specifically, we consider the problem of segmenting context data sequences into non-overlapping, internally homogeneous segments. The segments that are found may reflect certain states where the device, and eventually its user, are. Compact representations of the recognized segments can be used as templates against which the actions of the user (e.g., phone calls) are compared, so that for example prediction of future actions becomes possible. Formally, the segmentation problem is a special case of the general clustering problem, but the temporal structure of the data provides additional restrictions that can be used to speed up the computations.

The time series segmentation problem has been widely studied within various disciplines. A similar problem is the approximation of signals using line segments [1, 3, 7, 11, 13, 19]. The aim is often to compress or index the voluminous signal data [16, 18]. Computer graphics, cartography and pattern recognition utilize this reduction technique in simplifying or analyzing contour or boundary lines [12, 13]. Other applications range from phoneme recognition [14, 20] into paleoecological problems [2]. For an excellent review of time series segmentation, see [8] in this volume.

Our focus is on i) minimizing the cost function with a given number of segments, ii) cost functions that are sums of segmentwise costs, and iii) off-line segmentation, where all the data to be segmented is readily available.

0-7695-1119-8/01 $17.00 © 2001 IEEE

Based on dynamic programming (e.g., [1]), optimal solutions to the time series segmentation problem can be found in time $O(kN^2)$ for sequences of length N and for k segments. However, for large values of N and k the algorithms are not efficient enough. Greedy algorithms can be used to solve the segmentation problem in time $O(kN)$ with very small constants (e.g., [8]). The algorithms provide solutions that in most cases are very close to the optimal ones. We describe experimental results showing the quality of context recognition in a test scenario.

2 Definition of k-segmentation

A *time series* s consists of N samples $\mathbf{x}(1), \mathbf{x}(2), \ldots, \mathbf{x}(N)$ from $\mathbf{R}^d$. We use the notation $s(a, b)$ to define a *segment* of the time series s, that is, the consecutive samples $\mathbf{x}(a), \mathbf{x}(a+1), \mathbf{x}(a+2), \ldots, \mathbf{x}(b)$ where $a \leq b$. If $s_1 = s(a, b)$ and $s_2 = s(b+1, c)$ are two segments, then $s_1 s_2 = s(a, c)$ denotes their concatenation.

A *k-segmentation* S of s is a sequence $s_1 s_2 \cdots s_k$ of k segments such that $s_1 s_2 \cdots s_k = s$ and each s_i is non-empty. In other words, there are segment boundaries $c_1, c_2, \ldots, c_{k-1}$, $0 < c_1 < c_2 < \cdots < c_{k-1} < N$, where

$$s_1 = s(1, c_1),\ s_2 = s(c_1+1, c_2),\ \ldots,\ s_k = s(c_{k-1}+1, N).$$

For ease of notation, we define additionally $c_0 = 0$ and $c_k = N$.

We are interested in obtaining segmentations of s where the segments are internally homogeneous. In order to formalize this goal, we associate a cost function F with the internal heterogeneity of individual segments, and aim to minimize the overall cost of the segmentation. We make two assumptions on the overall cost. First, the cost $\text{cost}_F(s(a, b))$ of a single segment is a function of the data points and the number of data points $n = b - a + 1$,

$$\text{cost}_F(s(a, b)) = F(\mathbf{x}; n | \mathbf{x} \in s(a, b)). \tag{1}$$

Second, the *cost of a k-segmentation* $\text{Cost}_F(s_1 s_2 \cdots s_k)$ is the sum of the costs of its segments $s_1, s_2, \ldots, s_k$:

$$\text{Cost}_F(s_1 s_2 \cdots s_k) = \sum_{i=1}^{k} \text{cost}_F(s_k). \tag{2}$$

An optimal k-segmentation $S_F^{opt}(s; k)$ of time series s using cost function cost_F is such that $\text{Cost}_F(s_1 s_2 \cdots s_k)$ is minimal among all possible k-segmentations.

The cost function F in Eq. 1 can be an arbitrary function. We use the sum of the variances of the components of the segment:

$$V(s(a, b)) = \sum_{l=1}^{d} \left[\frac{1}{n} \sum_{i=a}^{b} x_l(i)^2 - \left(\frac{1}{n} \sum_{i=a}^{b} x_l(i) \right)^2 \right], \tag{3}$$

where $n = b - a + 1$ and d is the number dimensions. Thus the cost function for segmentations is simply

$$\text{Cost}_V(s_1 s_2 \cdots s_k) = \frac{1}{N} \sum_{i=1}^{k} n_i V(s_i), \tag{4}$$

where the segments have length $n_1, n_2, \ldots, n_k$, the length N of the sequence is $\sum_{i=1}^{k} n_k$, and $V(s_i)$ is defined as in Eq. 3.

By rewriting Eq. 4 we get

$$\text{Cost}_V = \frac{1}{N} \sum_{i=1}^{k} \sum_{j=c_{i-1}+1}^{c_i} \|\mathbf{x}(j) - \mu_i\|^2 \tag{5}$$

where μ_i is the mean vector of data vectors in segment $s_i = s(c_{i-1} + 1, c_i)$.

The problem we address is finding the segment boundaries c_i that minimize the cost. The problem is similar to clustering, but simpler. Eq. 5 is well comparable to a typical error measure of standard vector quantization (clustering), but in this case the clusters are limited to being contiguous segments of the time series instead of Voronoi regions in $\mathbf{R}^n$.

3 Algorithms

Dynamic programming The k-segmentation problem can be solved optimally by using dynamic programming [1]. The basic criteria for the applicability of dynamic programming to optimization problems is that the restriction of an optimal solution to a subsequence of the data has to be an optimal solution to that subsequence. For example, for our problem, given an optimal k-segmentation $s_1 s_2 \cdots s_k$, any subsegmentation $s_i \cdots s_j$ $(1 \leq i < j \leq k)$ is an optimal $(j-i+1)$-segmentation for the corresponding subsequence, so the condition obviously holds. The computational complexity of the dynamic programming is of order $O(kN^2)$ if the cost of a segmentation can be calculated in linear time.

The computational complexity of the dynamic programming algorithm is too high when there are large amounts of data. Greedy methods can take advantage of the simple fact that when the segmentation for a subsequence is changed, e.g., if a segment is divided further or a set of subsequent segments is redivided, the reduction in the total cost can be calculated efficiently within the subsequence.

A well-known and fast greedy heuristic for segmentation is the *top-down* approach or binary-split (e.g., [8]). This makes splits in hierarchical manner.

Top-down The method starts by splitting the time-series s optimally into two subsequences s_1 and s_2. Now assume

that the algorithm has already segmented s into $m < k$ segments. Each of these segments $s_i, i = 1, 2, \ldots, m$ is split in turn optimally into two pieces s_{i_1} and s_{i_2} and the total cost of the segmentation $s_1 s_2 \ldots s_{i_1} s_{i_2} \ldots s_m$ is calculated. The split which reduces the total cost most is accepted. Now there are $m + 1$ segments and the procedure is carried on until there are k segments.

The top-down method never makes changes in the break points it has once set. The inflexibility of top-down is potentially a weak point, since it can turn out later in the process that the early decisions are far from optimal.

This problem can be assessed with dynamic procedures that first heuristically place all break points and then iteratively move one break point at a time using some decision rule that ensures convergence to some local optimum [11]. We next propose two greedy algorithms that move one breakpoint at a time straight into a local minimum.

Local iterative replacement (LIR) LIR is a simple greedy procedure where the new place for a break point is selected optimally between the neighboring two break points (including the beginning and ending of the time series). The approach is similar to [11] where break points are moved gradually towards better positions, rather than to the locally optimal ones.

1. Select the initial break points heuristically, e.g., by using evenly spaced or random initial locations, or with the top-down method.
2. Select a break point c_i, $1 \leq i \leq k-1$, either in random or sequential order, remove it and concatenate the two consecutive segments that meet at c_i into $s(c_{i-1} + 1, c_{i+1})$.
3. Find a new, optimal location for the break point in the concatenated sequence: locate an optimal 2-segmentation break point c_i' for the concatenated segment. Replace break point c_i by c_i' in the solution.
4. Steps 2 and 3 are repeated until a stopping criterion is met. (Possible stopping criteria are discussed below.)

Global iterative replacement (GIR) Instead of relocating the break point c_i between its neighbors c_{i-1} and c_{i+1}, the best location is searched in the whole sequence. This includes clearly local iterative replacement but it may avoid some local minima. The approach bears some distant similarities with [13], where segments are also split and merged. The core idea of [13] is to split segments with large errors and merge ones with small errors until given error thresholds are met, whereas GIR makes one (at that time) optimal split–merge pair at a time and keeps the number of segments constant.

1. Set the initial segmentation $S_n = s_1 s_2 \cdots s_k$; $n = 0$, as in LIR.
2. Select a break point c_i, $1 \leq i \leq k-1$, either in random or sequential order, remove it and concatenate the two consecutive segments that meet at c_i into $\hat{s} = s(c_{i-1} + 1, c_{i+1})$.
3. Find a new optimal location for a break point anywhere in the sequence. For each segment s_j', $j = 1, 2, \ldots, k-1$ in the new segmentation $S' = s_1 s_2 \cdots s_{i-1} \hat{s} s_{i+2} \cdots s_k$:= (renumeration) := $s'_1 s'_2 \cdots s'_{k-1}$, find the optimal 2-segmentation to s'_{j_1} and s'_{j_2}, and compute the respective (potential) savings $d_j = \text{cost}_F(s'_j) - (\text{cost}_F(s'_{j_1}) + \text{cost}_F(s'_{j_2}))$ in the segmentation cost.
4. Select the split with largest savings d_j, say s'_l with savings d_l, and set a break point at d_l. The new segmentation is $s'_1 s'_2 \cdots s'_{l_1} s'_{l_2} \cdots s'_{k-1}$.
5. Set $n := n + 1$ and renumerate the segments for the next round: $S_n = s_1 s_2 \cdots s_k$ (renumeration) := $s'_1 s'_2 \cdots s'_{l_1} s'_{l_2} \cdots s'_{k-1}$.
6. Steps from 2 to 5 are repeated until a stopping criterion is met.

A natural stopping criterion for these algorithms is that the total cost cannot be decreased by any admissible move of a breakpoint. It is immediate that both LIR and GIR will stop in finite number of steps, since the cost decreases at each step and the number of points in the sequence is finite and discrete. A limit for the number of iterations is another simple stopping criterion.

The randomized iterative algorithms can be run a few times in order to reduce the chance of having an especially poor local minima.

The computational complexity of the greedy methods is linear in the size of the input data, if the cost of a segmentation can be calculated in linear time, as it can for the variance (cost function of Eq. 4). The complexity is of order $O(KN)$ where the factor K depends on the number of the break points k, on the number of iterations, and on the locations of the initial break points.

4 Experimental performance evaluation

The proposed iterative algorithms were benchmarked against the top-down algorithm. The optimal solution provided by the dynamic programming algorithm was used as a reference. The tested algorithms were

1. top-down approach
2. local iterative replacement (LIR)

3. global iterative replacement (GIR)

The cost function to be minimized was variance (Eq. 5). First, the algorithms were tested with a number of different artificial data sets.

4.1 Artificial data sets

The artificial data was generated as follows. First, the length $N \in \{100, 200, 300, 400, 500\}$ of the signal and the number $c \in \{6, 11, 16, 21\}$ of constant segments was fixed. Then the values and lengths for the constant segments were generated randomly, so that the $c - 1$ places for transitions (break points) were drawn randomly from $\{2, 3, \ldots, N-1\}$ with the restriction that the length of any segment had to be at least two. The value for each constant segment was randomly generated from a uniform distribution on $[0, 1]$.

Three different random prototype signals of this kind were generated for each combination of the number of segments and the length of signals, and three levels of gaussian *i.i.d.* noise was added to the signals. For the relative noise level (SNR; ratio of variance of signal to variance of noise) we used levels 10 and 1.0. For each prototype signal and for each combination for N, c and SNR level we generated 30 time series. Fig. 1 shows an example of two test signals ($N = 300$ and $c = 6$). Both are generated by adding different level of noise to the same prototype. Sample A has very low noise (SNR=100) while B has SNR=1.0. Each test signal was segmented once using the dynamic programming, top-down, LIR, and GIR algorithms. The number of segments k was set to c, the a priori number of the segments. We call this simply the "Test 1" data set.

For some larger experiments, another set of data was generated with the following procedure. One piecewise constant prototype was generated with $N = 500$ and $c = 16$, and fifty samples were generated by adding some gaussian *i.i.d* noise SNR=10. We call this the "Test 2" data set.

4.2 Comparison of partitionings

A random initial location for the break points was given for the iterative algorithms. Both LIR and GIR used the same initial segmentations. The iteration was stopped when the algorithms could not move the break points any more.

We compare the costs C of the segmentations achieved by top-down and proposed iterative methods LIR and GIR to the cost of the optimal solution C_{opt} achieved by dynamic programming. The comparison is made using a relative error measure since different signals and segmentations are compared:

$$err_{rel}(C) = \frac{C - C_{opt}}{C_{opt}}$$

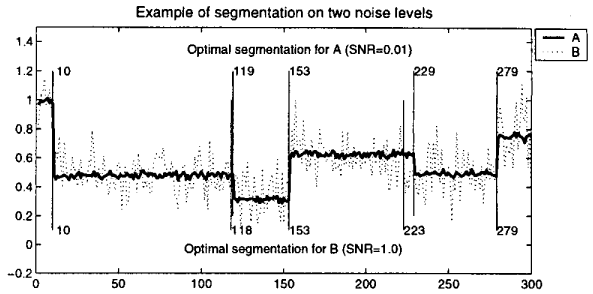

Figure 1. Two noise levels (SNR=100, SNR=1.0) added to an original signal of 6 constant segments, and their optimal segmentations.

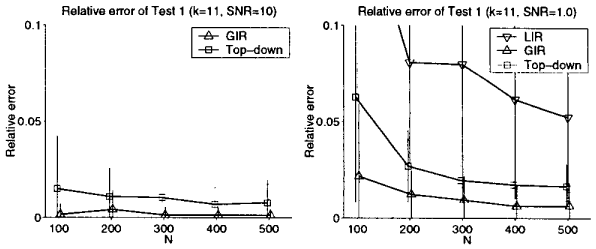

Figure 2. The average relative error of segmentation as a function of the length of the sequence (N), for low to high amounts of noise. Errorbars show one standard deviation. Local iterative replacement (LIR) is outside the visible area in the left panel.

This measure tends to increase without limit if C_{opt} goes to zero. However, since there was always noise present in the test signals C_{opt} is always greater than zero.

Experiments with simulated data (Test 1) and the "correct" amount of segments show that the relative errors of the partitionings produced by global iterative replacement are within a percent or two (Figs. 2 and 3). Local iteration performs badly, but the top-down method produces reasonable partitionings. The results indicate that for the purposes of the k-segmentation problem defined in 2 the top-down method and the global iterative method are sufficiently accurate.

More interesting results are obtained when the correct number of segments is not known. Of course, this is the situation in typical applications. Experiments with artificial data set "Test 2" show that GIR consistently outperforms the top-down approach, which in turn is superior to LIR. The better performance of GIR over the top-down method is probably explained by the fact that GIR can during the

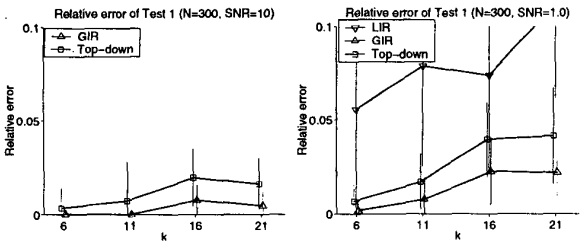

Figure 3. The average relative error of segmentation as a function of the number of segments (k), for low to high amounts of noise. Errorbars show one standard deviation. Local iterative replacement (LIR) is outside the visible area in the left panel.

operation change decisions made earlier, whereas the top-down method cannot.

4.3 Running times

As expected, the computational requirements of the top-down method and both local and global iterative replacements are linear in N, the length of the sequence, whereas the dynamic programming method is quadratic (Fig. 5).

For constant N, all methods behave roughly linearly in k (Figs. 6 and 7). However, dynamic programming has about two orders of magnitude larger consumption of computational resources (Fig. 7). Right panel of Fig. 7 allows a closer look at behavior in k. In average, both LIR and GIR behave in a similar fashion to the top-down algorithm.

Fig. 7 gives even an impression that, for this particular case, the tested greedy algorithms might behave sublinearly in k.

5 Context recognition

Real context data was collected with custom-built equipment. Sensor signals were logged from a certain user scenario where test subjects were told to perform different activities (Table 1).

5.1 Context data

The data were recorded using microphones and a sensor box that were attached to a mobile phone. The combination was hanging in users' neck in front of the chest. The data were logged by wire to a laptop that the user was carrying.

The raw signal was transformed to 19 variables, called *context atoms*, that indicate the amount of activation for an action or state: movement (running, walking fast, walking),

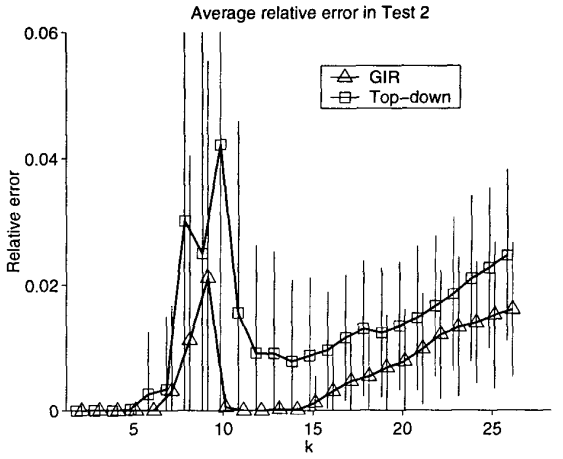

Figure 4. The average relative error of segmentation as a function of the number of segments (k), for an artificial sequence (Test 2: N=500, c=16) Local iterative replacement (LIR) is outside the visible area.

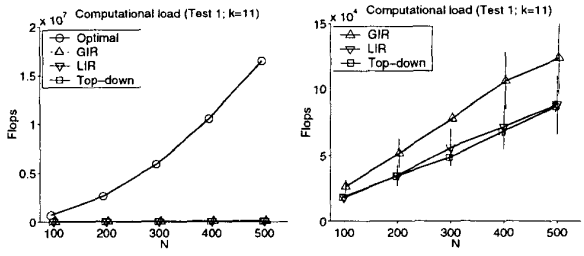

Figure 5. The average computational cost in floating point operations as a function of the length of the sequence (N). Both panels show the same data but on different scales. Errorbars show one standard deviation.

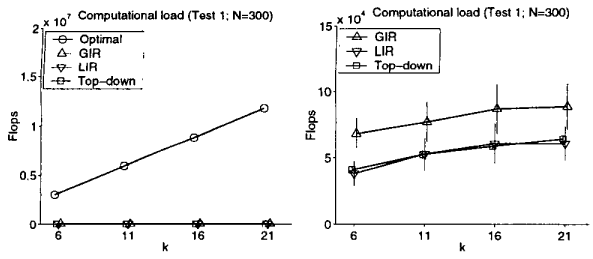

Figure 6. The computational cost in floating point operations as a function of the number of segments (k). Both panels show the same data but on different scales. Errorbars show one standard deviation.

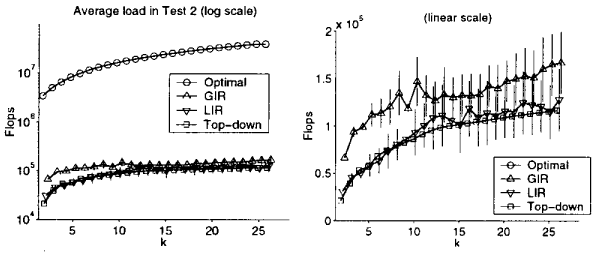

Figure 7. The average computational cost in floating point operations as a function of the number of segments (k), for an artificial sequence (Test 2: N=500, c=16). Errorbars show one standard deviation. Note that the scale on y-axis is logarithmic in the left panel and linear in the right one. The dynamic programming is out of the visible area in the right panel.

Table 1. Outline of activities in the user scenario

1.	user sits; device is on a table
2.	takes the device and puts it on
3.	stands up and starts to walk
4.	walks in a corridor
5.	walks down the stairs
6.	walks in a corridor
7.	walks outside
8.	walks in a porch
9.	walks in a lobby
10.	walks up the stairs
11.	walks in a corridor
12.	sits down
13.	puts the device on a table

sound pressure (loud, modest, silent), illumination conditions (total darkness, dark, normal, bright), touch (at hand), stability (unstable, stable), and device orientation (sideways left, sideways right, antenna up, antenna down, display up, display down). The pattern recognition algorithms for this transformation and the sensor box itself are outside the scope of this research, and we consider the context atom data as given.

The real context data set consisted of 44 time series arising from the same scenario. The lengths of the 19 dimensional time series varied between 223 and 267.

5.2 Performance on real context data

Each time series was segmented once to $2, \ldots, 21$ segments using the dynamic programming, top-down, LIR, and GIR algorithms. Results on relative error and running times (Fig. 8) confirm the observations made with the artificial data sets: GIR yields a small relative error with high computational efficiency.

5.3 Quality of context recognition

A study of the optimal 2,. . .,21-segmentations for the real data shows that there is certain stability in the locations of the break points (Fig. 9A). Most of the break points occur in almost all segmentations after they occur the first time. This gives certain credibility for the break points.

Next, examine a time series from context atom data and its optimal 13-segmentation (Fig. 9B). An evaluation of the segmentation against a video recording of the test shows that segmentation can be very useful for context recognition. The segmentations seem to capture the most important changes when compared with the real situation: putting the equipment on and standing up, walking in stairs, being out, going through doors, stopping, and getting the equipment off. The 13 segments correspond practically one-to-one to the activities in the user scenario (Table 1). Furthermore, by comparing Figs. 9A and B one sees that the break points for phases "getting equipment on", "being out", and "getting equipment off" come up first. In this case, the order of the appearance of the break points and their (subjective) importance in real world seem to be consistent as well.

6 Conclusions

Context-awareness is becoming one of the major factors in mobile communications. We have studied a particular problem arising from mobile phones with sensors, the task of time series segmentation.

We outlined the dynamic programming algorithm for finding the optimal k-segmentation for a given cost function. However, dynamic programming is computationally

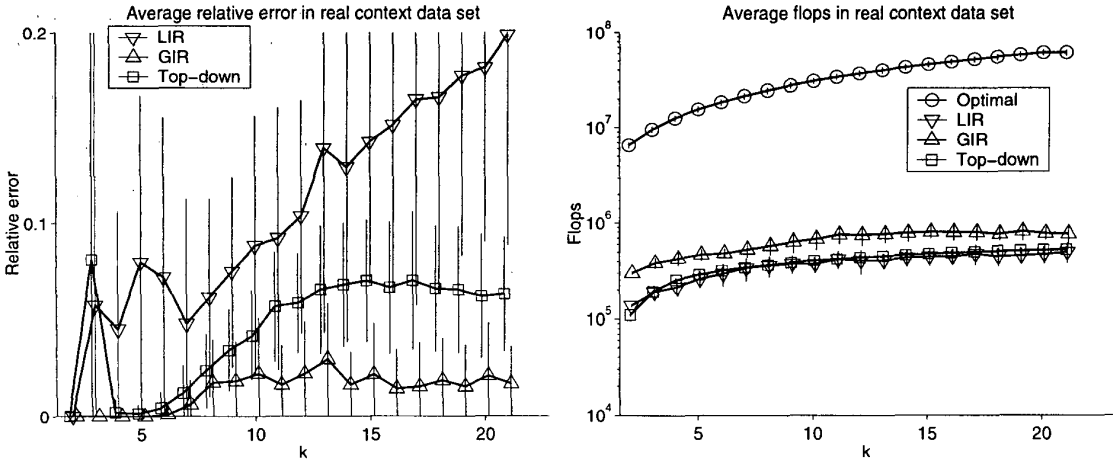

Figure 8. The average relative error (left) and computational cost in floating point operations (right) as functions of the number of segments (k), for real context data set. Errorbars show one standard deviation. Note that the scale on y-axis is logarithmic in the right panel.

too hard for long sequences, since the complexity of the algorithm is of order $O(kN^2)$ where N is the amount of data (assuming the cost of a single segment can be computed in linear time). For this reason, we proposed and analyzed fast greedy methods for time series segmentation. These iterative methods were tested against both the optimal segmentations and the top-down greedy segmentations. The proposed global iterative replacement method (GIR) outperformed other greedy methods in our empirical tests.

The cost function that was minimized in this work was the variance of the segment. This is just an example, and other cost functions might be considered. It is, however, advisable that the cost function could be calculated in linear time with respect to the amount of data, as is the case with the sample variance.

We applied the time series segmentation into sensor data that was collected using a sensor box in a mobile phone. The time series segmentation was used to capture interesting changes in the user's context. The experiment suggests that time series segmentation using a simple variance based cost function captures some essential phenomena in the context atom time series. The segmentations presented in Fig. 9 are optimal for the given data and cost function, but the reflections of this fact to the real world must be evaluated by an analyst. The analyst might be working using tools like visualization aids presented in Fig. 9, as well as video recordings of the tests to get an overview for some data in order to determine the usefulness of the emerged patterns.

For other analysis purposes, the segmentation gives an adaptive length window where the situation within window is more or less constant and the break points occur at places of changes. This can be useful in preprocessing the data for forming higher-level contexts.

Acknowledgments

The authors would like to thank Esa Alhoniemi, Jani Mäntyjärvi, and Jari Paloniemi for useful comments.

References

[1] R. Bellman. On the Approximation of Curves by Line Segments Using Dynamic Programming. *Communications of the ACM*, 4(6):284, 1961.

[2] K. D. Bennett. Determination of the Number of Zones in a Biostratigraphical Sequence. *New Phytol.*, 132:155–170, 1996.

[3] A. Cantoni. Optimal Curve Fitting with Piecewise Linear Functions. *IEEE Transactions on Computers*, C-20(1):59–67, 1971.

[4] B. Clarkson, K. Mase, and A. Pentland. Recognizing User Context via Wearable sensors. In *Digest of Papers of the Fourth International Symposium on Wearable Computers*, pages 69–76. IEEE, 2000.

[5] B. Clarkson and A. Pentland. Unsupervised Clustering of Ambulatory Audio and Video. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing 1999*, volume 6, pages 3037–3040, 1999.

[6] S. Fels, Y. Sumi, T. Etani, N. Simonet, K. Kobayashi, and K. Mase. Progress of C-Map: a Context-Aware Mobile Assistant. In *Proceedings of AAAI 1998 Spring Symposium on Intelligent Environments*, pages 60–67, 1998.

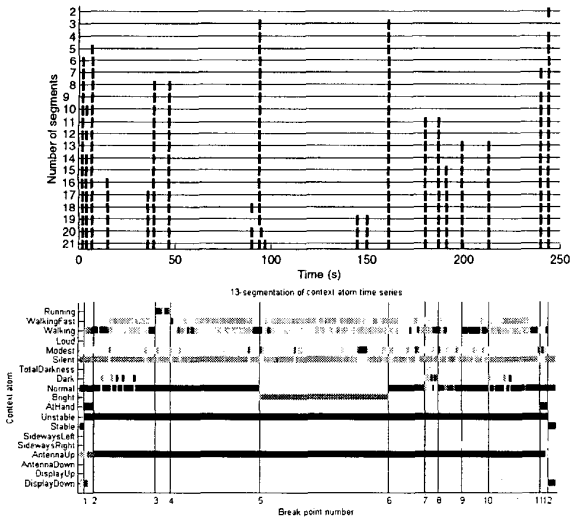

Figure 9. A. The optimal 2,...,21-segmentations of real data. The vertical line segments on each horizontal grid line of the y-axis presents one temporal segmentation. The x-axis shows time in seconds from the beginning of the test. B. The context atom data and the optimal 13-segmentation. The horizontal bars show the activation of the context atoms (labels on y-axis). Dark means high activation. The vertical lines that are numbered on x-axis show the places of break points. The segmentation here is the optimal 13-segmentation.

[7] H. Imai and M. Iri. An Optimal Algorithm for Approximating a Piecewise Linear Function. *Journal of Information Processing*, 9(3):159–162, 1986.

[8] E. Keogh, S. Chu, D. Hart, and M. Pazzani. An Online Algorithm for Segmenting Time Series. In *Proceedings of the First IEEE International Conference on Data Mining*, 2001. To appear.

[9] K. V. Laerhoven and O. Cakmakci. What Shall We Teach Our Pants? In *Digest of Papers of the Fourth International Symposium on Wearable Computers*, pages 77–83. IEEE, 2000.

[10] M. Lamming and M. Flynn. "Forget me not" Intimate Computing in Support of Human Memory. Technical Report EPC-1994-103, Rank Xerox Research Centre, Cambridge.

[11] T. Pavlidis. Waveform Segmentation Through Functional Approximation. *IEEE Transactions on Computers*, C-22(7):689–697, 1973.

[12] T. Pavlidis. Algorithms for Shape Analysis and Waveforms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-2(4):301–312, 1980.

[13] T. Pavlidis and S. L. Horowitz. Segmentation of Plane Curves. *IEEE Transactions on Computers*, C-23(8):860–870, 1974.

[14] P. Prandoni, M. Goodwin, and M. Vetterli. Optimal Time Segmentation for Signal Modeling and Compression. In *Proceedings of IEEE International Conference on Acoustics , Speech and Signal Processing 1997*, volume 3, pages 2029–2032, 1997.

[15] A. Schmidt, K. Aidoo, A. Takaluoma, U. Tuomela, K. V. Laerhoven, and W. V. de Velde. Advanced Interaction in Context. In *Hand Held and Ubiquitous Computing*, number 1707 in Lecture Notes in Computer Science, pages 89–101. Springer-Verlag, 1999.

[16] H. Shatkay and S. B. Zdonik. Approximate queries and representations for large data sequences. In *Proceedings of the 12th International Conference on Data Engineering*, pages 536–545. IEEE, 1996.

[17] T. Starner, B. Schiele, and A. Pentland. Visual Contextual Awareness in Wearable Computing. In *Second International Symposium on Wearable Computers, Digest of Papers*, pages 50–57. IEEE, 1998.

[18] C. Wang and X. S. Wang. Supporting Content-based Searches on Time Series via Approximation. In *Proceedings of the 12th International Conference on Scientific and Statistical Database Management*, pages 69–81. IEEE, 2000.

[19] L.-D. Wu. A Piecewise Linear Approximation Based on a Statistical Model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(1):41–45, 1984.

[20] Z. Xiong, C. Herly, K. Ramchandran, and M. T. Orchard. Flexible Time Segmentations for Time-Varying Wavelet Packets. In *IEEE Proc. Intl. Symp on Time-Frequency and Time-Scale Analysis*, pages 9–12, 1994.