Pattern Recognition and Classification for Multivariate Time Series

Stephan Spiegel, Julia Gaebler, Andreas Lommatzsch
Ernesto De Luca, Sahin Albayrak
DAI-Labor
Technische Universitaet Berlin
Ernst-Reuter-Platz 7
10587 Berlin, Germany
{stephan.spiegel,julia.gaebler,andreas.lommatzsch,ernesto.deluca,sahin.albayrak}
@dai-labor.de

ABSTRACT

Nowadays we are faced with fast growing and permanently evolving data, including social networks and sensor data recorded from smart phones or vehicles. Temporally evolving data brings a lot of new challenges to the data mining and machine learning community. This paper is concerned with the recognition of recurring patterns within multivariate time series, which capture the evolution of multiple parameters over a certain period of time.

Our approach first separates a time series into segments that can be considered as situations, and then clusters the recognized segments into groups of similar context. The time series segmentation is established in a bottom-up manner according the correlation of the individual signals. Recognized segments are grouped in terms of statistical features using agglomerative hierarchical clustering.

The proposed approach is evaluated on the basis of real-life sensor data from different vehicles recorded during car drives. According to our evaluation it is feasible to recognize recurring patterns in time series by means of bottom-up segmentation and hierarchical clustering.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information Filtering, Clustering*; G.3 [**Mathematics of Computing**]: Probability and Statistics—*Time Series Analysis*

General Terms

Algorithm, Design, Experimentation

Keywords

Time Series Segmentation, Singular Value Decomposition, Pattern Recognition, Agglomerative Hierarchical Clustering

SensorKDD '11, August 21, 2011, San Diego, CA, USA.

1. INTRODUCTION

Since most electronic components have gotten substantially smaller and cheaper to produce, an increasing number of everyday electronic devices are equipped with smart sensors. Recent mobile phones have integrated location and acceleration sensors, and most modern vehicles are able to measure ambient temperature and average fuel consumption. There is a significant trend in using these integrated sensors to simplify human-computer interaction by means of automatic adaptation of user interfaces or even machine behaviour according to the recognized context [3, 4, 8]. With regard to sensors; context or situations are usually described as segments of multivariate time series, which are internally homogeneous. In the field of time series segmentation the term *homogeneous* refers to an unchanging correlation between the observed variables over a specified period of time. Due to the fact that most situations have different length and are often overlapping, it is a non-trivial task to determine clear borders between individual time series segments [1]. Figure 1 illustrates the necessity of machine learning techniques for automatic recognition and classification of time series segments or situations respectively.

Our segmentation approach employs *Singular Value Decomposition* model [2, 11, 12] to find segment borders at those points in time where the correlation structure of the observed variables fulfil a significant change. We rely on the fact that SVD gives a close approximation of our time series data, whenever the observed variables exhibit a linear correlation. In other words, the SVD model is expected to give a high reconstruction error, if and only if the decomposed time series data fulfils a significant change in correlation structure [2]. The proposed bottom-up segmentation algorithm is based on the assumption that two adjacent time series segments can only be merged in case of a low reconstruction error, implying that the correlation structure among the observed variables does not considerably change within the two contiguous segments (refer to Figure 2).

The proposed SVD-based time series segmentation is mainly inspired by the sensor fusion algorithm developed by Abonyi [2]. However, we extended the original sensor fusion algorithm by several features, such as automatic determination of model rank and merge threshold, as well as initial fine-grain segmentation according to critical points of individual signals [9]. Furthermore, we evaluate the extended segmentation algorithm on both synthetic time series data and real-

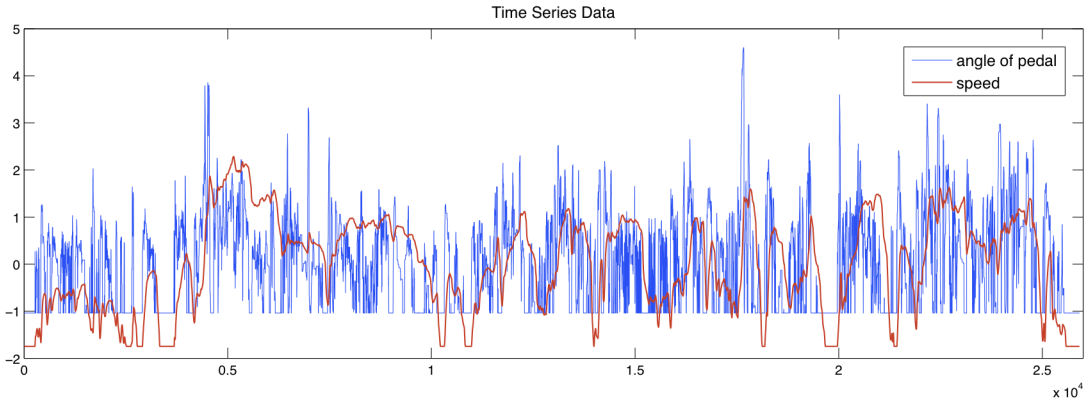


Figure 1: Progression of speed and accelerator signal during a car drive

In order to compare signals against each other the time series data was normalized via z-score (the original scale of the speed signal ranges from 0 to 130 km/h, and the accelerator measures vary between 0 to 100 degrees). Although speed and accelerator signal are highly correlated, it is not visible to the naked eye at which point in time a specific situation starts or ends (e.g. stop and go at a traffic light). Due to the fact that we normally consider multiple signals at the same time, it is advisable to apply machine learning techniques for automatic recognition and classification of recurring patterns [6].

life multivariate sensor signals which give information about the progression of motor parameters during car drive. In case of motor signals, a time series segment could represent a critical situation that appeared during acceleration or loss of speed. Figure 1 illustrates a test drive in which speed and accelerator angle were recorded.

Our main interest is the recognition of patterns, or rather situations, in sensor data recorded during a car drive. To identify recurring situations we need to group the identified segments according their similarity. For the grouping of time series segments we propose agglomerative hierarchical clustering, which has the distinct advantage that any valid measure of distance or similarity can be used. Although the proposed time series segmentation and segment clustering is discussed in terms of vehicular sensors, it is also applicable to time series data of other domains.

The rest of the paper is organized as follows. Section 2 gives a general introduction to time series segmentation and furthermore discusses several different segmentation techniques. Section 3 describes the automatic classification of identified time series segments. The experimental results of both time series segmentation and clustering are presented in Section 4. Related work can be found in Section 5, and finally Section 6 concludes this paper.

2. TIME SERIES SEGMENTATION

Segmentation is the most frequently used subroutine in both clustering and classification of time series. It is often used to locate stable periods of time or alternatively to identify change points [2]. There already exist different heuristic approaches to extract internally homogeneous segments. Some primitive algorithms search for inflection points to locate episodes [13]. Other algorithms determine segment borders by the Sliding Window technique, were a segment is grown until it exceeds some error bound [6]. The contributed approach uses the bottom-up method, which begins creating a fine approximation of the time series, and iteratively merges the lowest cost pair of segments until some stopping criteria is met [6]. The cost of merging two adjacent segments is evaluated by the Singular Value Decomposition (SVD) model of the new segment. SVD-based algorithms are able to detect changes in the mean, variance and correlation structure among several variables. The proposed approach can be considered as an extension of the sensor fusion algorithm developed by Abonyi [2]. In the following, we describe the mathematical foundations of time series segmentation in more detail.

For further problem discussion we define a multivariate time series $T = \{x_k = [x_{1,k}, x_{2,k}, \ldots, x_{m,k}] | 1 \leq k \leq N\}$ as a finite set of N samples with m measurements labelled by time points $t_1, \ldots, t_N$ [2]. Consequently, a segment of T is defined as a set of consecutive time points $S(a,b) = a \leq k \leq b$, $x_a, x_{a+1}, \ldots, x_b$. The segmentation of time series T into c non-overlapping time intervals can thus be formulated as $S_T^c = \{S_e(a_e, b_e) | 1 \leq e \leq c\}$, were $a_1 = 1$, $b_c = N$ and $a_e = b_{e-1} + 1$.

For the purpose of finding internally homogeneous segments from a given time series, we need to formalize the cost function for the individual time intervals. In most cases, the cost function $cost(S(a,b))$ is based on the distance between the actual values of the time series and a simple function (linear function, or polynome of higher degree) fitted to the data of each segment [2].

Since our approach aims at detecting changes in the correlation structure among several variables, the cost function of the segmentation is based on the Singular Value Decomposition of the segment matrices, where each row is an observation, and each column is a variable. Before applying SVD, the observed variables need to be centred and scaled, in such a way as to make them comparable. Hence, we compute the z-scores using mean and standard deviation along each individual variable of the time series.

The singular value decomposition of a segment matrix X can be described as an factorization $X \approx U\Sigma V^T$ into a matrix Σ which includes the singular values of X in its diagonal in decreasing order, and into the matrices U and V which include the corresponding left-singular and right-singular vectors. With the use of the first few non-zero singular val-

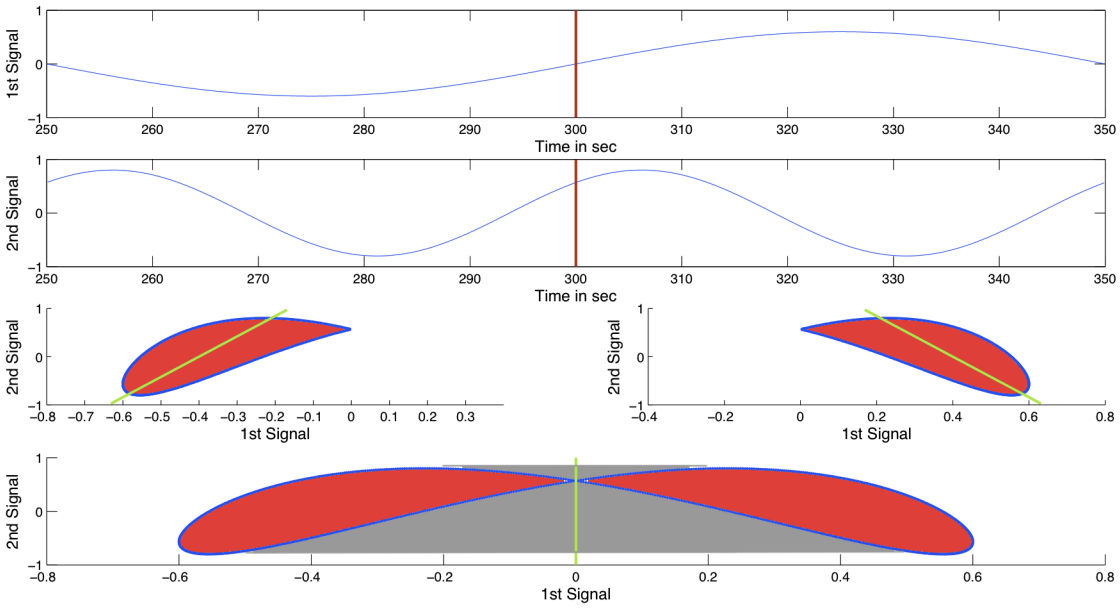


Figure 2: Segmentation of Multivariate Time Series via Singular Value Decomposition
The first two plots illustrate the progression of two synthetic signals which are linearly independent. Assuming that the time series was initially segmented at predetermined inflection points of the first signal (segment borders are highlighted in red color), we are faced with the decision whether to merge the depicted segments or not. Our approach relies on the fact that we only merge contiguous segments, if the correlation structure among the signals does **not** fulfil a significant change. Meaning that the reconstruction error of the employed SVD model does not exceed a predefined threshold. In case of two observed parameters the correlation structure of a segment can be expressed as a position vector (two-dimensional hyperplane) which gives an approximation of the original segment data. The scatter plots of the individual segments show the distribution of the data points (marked in blue color) together with the largest singular-vector (highlighted in green color), which was determined via singular value decomposition and describes the correlation structure among the examined signals. Furthermore, the area highlighted in red color can be considered as the reconstruction error, which is the distance between the original segment data and the approximation given by singular value decomposition. Note that the scatter plots of the individual segments illustrate the data points situated in time interval [250,300] and [300,350] respectively. In the last plot, we can see that the reconstruction error grows if we merge the examined segments. Beside the error of the individual segments (red color), we get an additional error highlighted in gray color. The growth of the reconstruction error accounts for the fact that the SVD model gives a less precise approximation of the merged segments, implying a significant change in correlation structure among the examined signals.

ues and the corresponding singular vectors, the SVD model projects the correlated high-dimensional data onto a hyperplane which is useful for the analysis of multivariate data. When the SVD model has an adequate number of dimensions, the distance of the original data from the hyperplane is a significant indicator for anomalies or unsteadiness in the progression of the observed variables. Hence, it is useful to analyse the reconstruction error of the SVD model to get information about the homogeneity of the factorized time series segments. In the proposed approach the reconstruction error is determined by the Q-measure [2], which is commonly used for the monitoring of multivariate systems and for the exploration of the errors. The Q-measure is defined as the mean squared deviation of projection P, which transfers our original segment data X into the p-dimensional subspace given by the Singular Value Decomposition. Due to the fact that we are mainly interested in the correlation structure among the observed variables, the projection P just considers the singular-vectors given by factor matrix V and neglect the hidden structure among the observations held in factor matrix U. The crux of the proposed time series segmentation is to use the Q-measure as an indicator for the homogeneity of individual or rather merged segments. Equation 1 to 3 formulate the derivation of the cost function:

$$P_{m,n} = \sum_{k=1}^{p} X_{m,n} V_{n,k} V_{k,n}^{T} \tag{1}$$

$$Q = \frac{1}{m \cdot n} \sum_{i=1}^{m} \sum_{j=1}^{n} P_{i,j}^{2} \tag{2}$$

$$cost_Q(S_e(a_e, b_e)) = \frac{1}{b_e - a_e + 1} \sum_{k=a_e}^{b_e} Q_{e,k} \tag{3}$$

The introduced bottom-up segmentation approach iteratively merges the cheapest pair of adjacent segments until a predetermined number of segments is reached or a specific thresh-

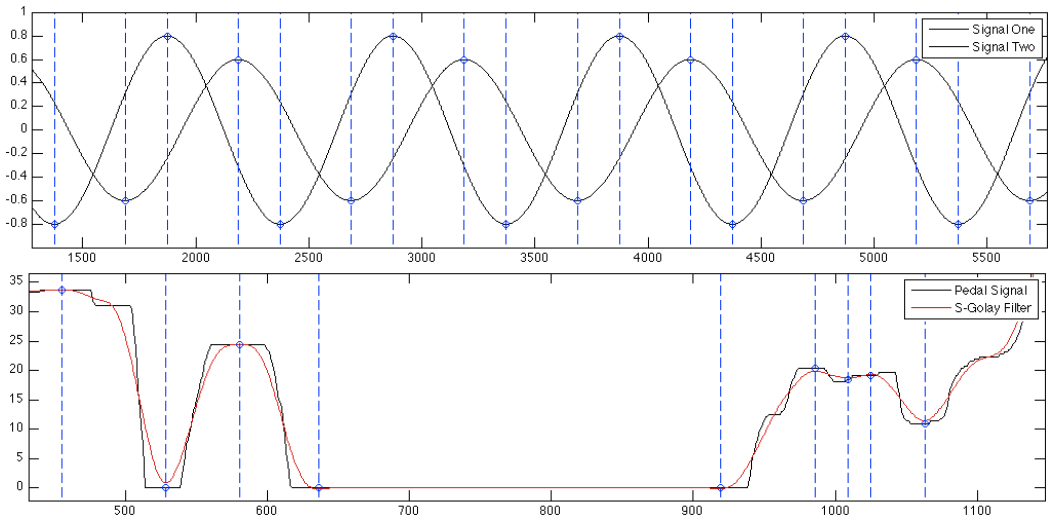


Figure 3: Time Series Segmentation based on Critical Points of Synthetic and Real-Life Signals
The first plot shows two synthetic signals, which are segmented according to their local extrema. Here the blue dots illustrate extrema and the blue dotted lines depict segment borders. In contrast to the common bottom-up segmentation approach, the critical point (CP) algorithm is able to find reappearing time intervals for initial segmentation in periodic signals. Consequently all merged segments will also contain reappearing sub patterns, which is absolutely necessary for segment classification or clustering. As illustrated in the second plot the critical point algorithm also gives satisfying results for real-life data. The plot shows the progression of the accelerator during a car drive as unvarnished and idealized signal (function plotted in black and red color respectively). For the smoothing of the accelerator signal we used the Savitzky-Golay filter with a second-order polynomial and a frame size of 30 data points. To eliminate residual noise we furthermore applied a moving average low-pass filter with the same frame size. For most signals the frame size has to be adjusted according the degree of noise and variance.

old is exceeded. By the time the proposed bottom-up algorithm terminates we expect that an optimal time series segmentation was found. Figure 2 illustrates the growth of the reconstruction error, when two contiguous segments are merged. One possible way to determine the merge threshold can be inferred from Figure 5(a).

The accuracy of the SVD model strongly depends on the rank of the decomposition. However, in most cases it is hard to determine the optimal rank in terms of model accuracy and computational complexity. Abonyi et al. [2] suggest to infer an appropriate rank from the scree-plot of the eigenvalues. Since this method is very imprecise and only works for variables with the same scale, we suggest to choose the model rank according the desired accuracy. In other words, we consider the k-largest singular values of matrix $\Sigma_{n\times n}$ that hold the required energy $[E_k = (sum_{i=1}^{k}\Sigma_{i,i})/(sum_{i=1}^{n}\Sigma_{i,i})]$. Note, that the calculation of the required model rank is based on the size of the final coarse segmentation. This derives from the fact that the approximation of large segments usually requires a high model rank, which most certainly also fits smaller segments.

The sensor fusion algorithm developed by Abonyi [2] merges segments bottom-up, whereas the initial fine-grain segmentation of the time series is determined manually. In case of a periodic signal and an odd initial segmentation it is highly probable that the bottom-up algorithm will not find reappearing segments, because the time series was partitioned into internal inhomogeneous time intervals. Therefore, we propose to perform the initial fine-grain segmentation of a time series according to the critical points of the individual signals. Critical points of great interest are extrema as well as inflection and saddle points. In order to calculate the critical points of a curve or signal we need to calculate the first, second or third derivative respectively.

Due to the fact that sensors often produce extremely noisy and fluctuate signals, it is very likely that the examined time series exhibit dozens of critical points. Hence, we need to smooth the signal with a low-pass or base-band-pass filter that reduces the number of critical points, but this does not substantially change the coordinates of the retrieved segmentation borders (refer to Figure 3).

In our critical point (CP) approach, we employ the Savitzky-Golay filter, which is typically used to smooth out a noisy signal whose frequency span is large. Savitzky-Golay smoothing filters perform much better than standard averaging FIR (Finite Impulse Respond) filters, which tend to filter out a significant portion of the signal's high frequency content along with the noise. Although Savitzky-Golay filters are more effective at preserving the pertinent high frequency components of the signal, they are less successful than standard averaging FIR filters at rejecting noise. Savitzky-Golay filters are optimal in the sense that they minimize the least-squares error in fitting a higher-order polynomial to frames of noisy data [9].

The introduced critical point (CP) approach can be employed as a preprocessing step for the proposed bottom-up segmentation (BU), but can also be considered as a segmentation technique by its own. In Section 4 we evaluate the mixed approach (BUCP) as well as both segmentation techniques separately.

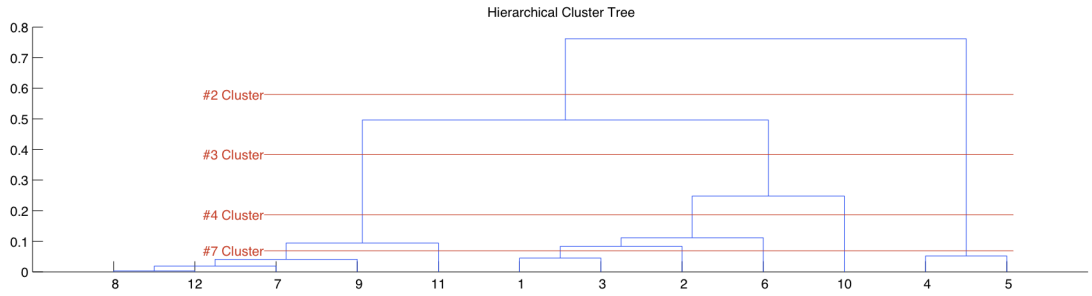


Figure 4: Dendrogram for Cluster Analysis of Time Series Segments

In general, the x-axis of a dendrogram identifies the indexes of the items (e.g. time series segments) that were grouped together, and the y-axis denotes the distance at which the single items were merged. Cutting a hierarchical cluster tree (dendrogram) at a specific height will give a clustering (highlighted in red color) for the selected precision or rather distance.

3. PATTERN CLASSIFICATION

One major advantage of the proposed segmentation algorithm is that it allows one to reuse the SVD model to define a distance measure to compare multivariate time series segments [17,18]. Consider two segments S_i and S_j with data matrices X_i and X_j that hold the same n variables, and whose singular value decomposition transforms the original data into a k-dimensional subspace, denoted by $U_{i,k}\Sigma_{i,k}V_{i,k}^T$ and $U_{j,k}\Sigma_{j,k}V_{j,k}^T$ respectively. Then the similarity between these subspaces can be defined as the sum of the squares of the cosines of the angles between each vector of matrix $W_{i,k}$ and $W_{j,k}$, which are composed by multiplying the respective singular values and right-singular vectors ($W_{i,k} = \Sigma_{i,k}V_{i,k}$ and $W_{j,k} = \Sigma_{j,k}V_{j,k}$). This similarity measure is formalized by the following equation:

$$S_{SVD} = \frac{1}{k} trace((W_i^T W_j)(W_j^T W_i)) \tag{4}$$

Due to the fact that subspaces W_i and W_j contain the k most important singular vectors that account for most of the variance in their corresponding data set, S_{SVD} is also a measure of similarity between the segments S_i and S_j.

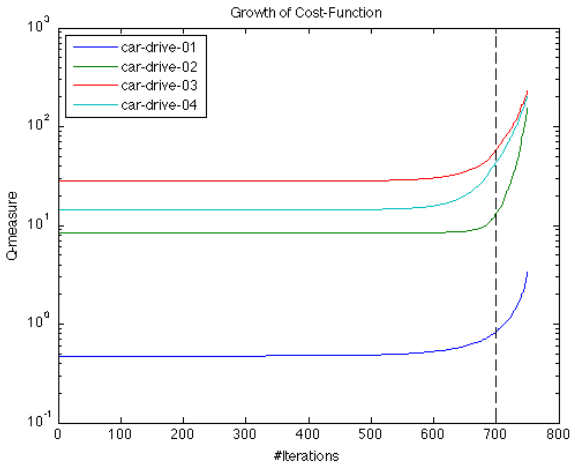
Cluster analysis is the assignment of a set of observations into subsets (called clusters) so that observations in the same cluster are similar in some sense. It is a method of unsupervised learning, and a common technique for statistical data analysis used in fields like machine learning, data mining and information retrieval. Hierarchical methods find successive clusters using previously established ones. Agglomerative hierarchical clustering algorithms begin with each element as a separate cluster and merge them into successively larger clusters. The merges and splits are determined in a greedy manner. The results of hierarchical clustering are usually presented as a dendrogram. Figure 4 illustrates the dendrogram for a sample of time series segments.

Hierarchical clustering has the distinct advantage that any valid measure of distance can be used. In fact, the observations themselves are not required; all that is used is a matrix of distances. The choice of an appropriate metric does influence the shape of the clusters, as some elements may be close to one another according to one distance and farther away according to another. Some commonly used metrics for hierarchical clustering are Euclidean distance, cosine similarity, Mahalanobis distance (covariance matrix) and Pearson's correlation. For text or other non-numeric data, metrics such as the Hamming distance or Levenshtein distance are often used. Given an appropriate metric, an agglomerative hierarchical cluster tree can be created by several different methods, which differ from one another in how they measure the distance between the clusters. Available methods are single and complete linkage, also known as shortest and furthest distance, as well as average, centroid and inner-squared distance. In our proposed approach we employ cosine distance (refer to Equation 5) to calculate the similarities between feature vectors and average linkage to compute average distance between all pairs of objects in any two clusters (see Equation 4).

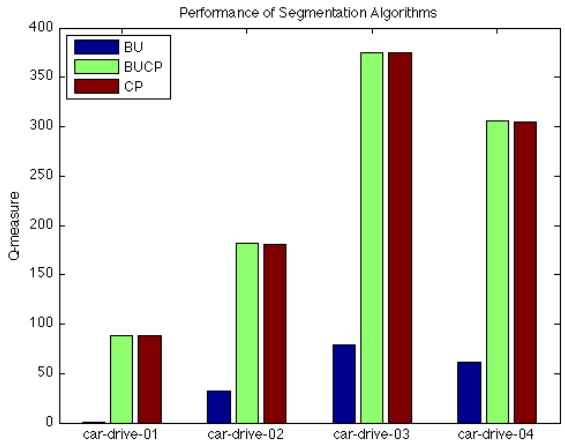
$$d(r,s) = \frac{1}{n_r n_s} \sum_{i=1}^{n_r} \sum_{j=1}^{n_s} dist(x_{ri}, x_{sj}) \tag{5}$$

$$dist(x_{ri}, x_{sj}) = 1 - \frac{x_{ri}x_{sj}^T}{\sqrt{(x_{ri}x_{ri}^T)(x_{sj}x_{sj}^T)}} \tag{6}$$

Although distance metric and cluster measure directly influence the grouping of items or rather segments, meaningful clustering always depends on the selection of expressive features. In machine learning and statistics, feature selection is the technique of selecting a subset of relevant characteristics for building robust learning models. From a theoretical perspective, it can be shown that optimal feature selection for (un)-supervised learning problems requires an exhaustive search of all possible subsets of features. For practical learning algorithms, the search is for a satisfactory set of features instead of an optimal set. Beside the previously introduced feature extraction via SVD, we furthermore examine common signal processing features like standard deviation, local maxima and minima as well as the increase or decrease of the individual signals within a segment. The evaluation of the proposed time series segmentation and segment clustering is discussed in the following section.



(a) Growth of Cost-Function



(b) Performance of Algorithms (lower Q-measure is better)

Figure 5: Evaluation of Time Series Segmentation

(a) The Q-measure indicates how well our SVD-based model fits the segmentation of a time series. In the case that the segmentation cost (sum of Q-measures over all segments) exceeds a certain limit or the cost function exhibits a steep ascent, the bottom-up algorithm should stop to merge adjacent segments. The above plot shows the evolution of the segmentation cost for four different car drives of same length (8000 ms). All time series were initially partitioned into 800 segments of equal length, which then were iteratively merged to a final number of 50 segments by our bottom-up algorithm (doing 750 merge operations). The above plot illustrates that for all car drives the segmentation costs increases rapidly after about 700 iterations. Under these conditions, it would be advisable to terminate the bottom-up algorithm after a total number of 100 segments were established, which is furthermore used as a threshold for the following performance evaluation.

(b) This plot shows the performance of three different segmentation algorithms in terms of the employed cost-function. For our evaluation we assume that the total cost of time series segmentation is the sum of Q-measures over all segments. Due to the fact that the Q-measure is an indicator for the homogeneity of individual segments, it also gives us an idea about how accurate our SVD-based model fits the underlying structure of the examined time series. As a general rule, the lower the sum of Q-measures for a given segmentation the better. In order to compare the bottom-up (BU), critical point (CP) and mixed BUCP approach, we evaluate all algorithms with the same parameter settings and on the same time series datasets. The comparison shows that for all examined car drives the straightforward bottom-up algorithm performed best. To our surprise, both CP algorithm and mixed BUCP approach achieved similar results. This can be explained by the fact that the CP algorithm segments a time series according local extrema of individual signals, which might not have a noticeable effect on the Q-measure of the overall model (considering all examined signals), even if the established segments are merged by means of the introduced BU approach in a second step. The performance of (BU-)CP might vary slightly for each individual signal.

4. EVALUATION

In supervised learning, one usually has a training data set of input and target values which is used to adjust the parameters of the learning algorithm until it is able to model the underlying structure of the presented data. The trained model can then be used to predict target values for previously unseen input. However, in unsupervised learning one aims to discover the underlying structure of the presented data without having any target values for validation.

In case of our time series segmentation, we do not know what makes up a "good" segment and at which point of our bottom-up algorithm we should stop to merge adjacent segments. Nonetheless we are able to determine a merge threshold by means of the applied cost function. Figure 5(a) illustrates the growth of the Q-measure (sum over all segments) for each iteration of our bottom-up algorithm. A possible termination criteria for merging time series segments could be a steep ascent of the cost function. Beside automatically determining the number of segments (merge threshold), we furthermore compare different approaches of time series segmentation by evaluating the overall cost of the established segments. Figure 5(b) shows a comparison of our proposed bottom-up segmentation (BU), the introduced critical point (CP) approach and a mixed BUCP segmentation for several different car drives.

Clustering falls into the category of unsupervised learning algorithm as well, and therefore exhibits the common problem of validation. Similar to the segmentation task we do not know for which threshold we should stop to merge single items or rather clusters. However, we believe that it is possible to determine the number of clusters based on the internal homogeneity and external heterogeneity. In other words, similar to community detection in complex networks [7], we want to find groups of nodes or items that are more densely connected internally then with the rest of the network. Due to the fact that we employ average linkage as cluster measure, both internal and external item distance grow monotonically with an increasing number of clusters. For the evaluation we take account of the external cluster distance, which can be retrieved from the hierarchical clus-

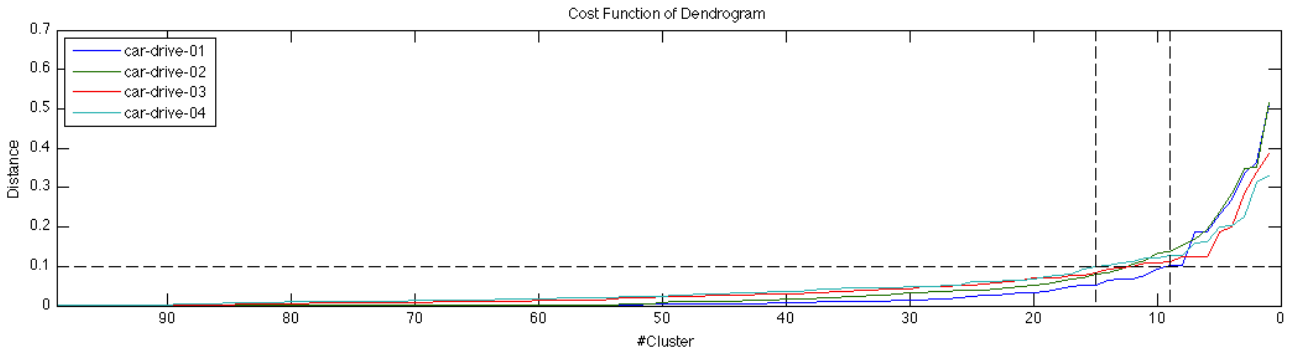


Figure 6: Cost Function of Hierarchical Cluster Tree

This plot illustrates the growth of distance between clusters for a decreasing number of groups. In other words, the graphs show at which cost the groups are merged by the agglomerative hierarchical cluster algorithm. Due to the fact that the cost function grows exponentially, the agglomerative hierarchical clustering should terminate at a certain point. If we assume a threshold of 0.1 distance, a number of 9 to 15 clusters would be appropriate to group the segments of the examined data sets. A threshold could be either predefined by an expert or learned by the cluster algorithm. For unsupervised learning the slope of the cost function is a useful indicator for determining a reasonable number of clusters with sufficient discrimination.

ter tree generated during segment grouping (also refer to the dendrogram in Figure 4). The cost function for several different sample datasets and the determination of a reasonable number of clusters can be inferred from Figure 6. However, cluster quantity always depends on the underlying structure of the time series and might vary from dataset to dataset.

For a better understanding of time series segmentation and clustering, we present an evaluated time series dataset in Figure 7. As mentioned previously, we are mainly interested in the recognition of complex drive manoeuvres in sensor data recorded from vehicles. To identify complex drive manoeuvres, we first of all employ time series segmentation and clustering as introduced before, and subsequently retrieve recurring patterns of segment sequences. In our case, sequences are regarded as sets of grouped segments, which exhibit a distinct order of cluster labels. If the determined time series segments are considered as individual situations in car driving, then segment sequences can be viewed as complex drive manoeuvres, like stop and go at a traffic light. Figure 7 illustrates several recurring sequences, which denote complex drive manoeuvres of a predefined length.

Drive manoeuvres might differ from person to person, and also depend on the motorization of the vehicles. Therefore it is interesting to analyse the patterns that characterize individuals or vehicles respectively. In our future work, we plan to identify sequences that describe normal drives or rather drivers, as well as sequences that can be regarded as critical drive manoeuvres or abnormalities. This knowledge is especially valuable for the motor vehicle industry, because it allows for performance optimization of an engine based on the understanding of frequent or abnormal situations. Although this paper discusses context recognition in terms of sensor data recorded from vehicles, the proposed time series analysis is also applicable to datasets from other domains. The following section gives some examples of related work.

5. RELATED WORK

The sensor fusion algorithm developed by Abonyi et al. [2] is able to segment a multivariate time series in a bottom-up manner. Adjacent segments are merged iteratively if their combined model does not exceed a predefined reconstruction error, which is computed by means of principal component analysis. The PCA model is used to measure the homogeneity of the segment according the change of correlation among the variables. Since the PCA model defines linear hyperplanes, the proposed segmentation algorithm can be considered as the multivariate extension of piecewise linear approximation. Our own work can be considered as an extension of the sensor fusion algorithm and has been evaluated on several real-life datasets.

In a following paper [1] Abonyi introduces a modified time series segmentation approach, which is able to group overlapping and vague segments by means of fuzzy clustering. The proposed algorithm favours contiguous clusters in time and is capable to detect changes in the hidden structure of multivariate time-series, where local probabilistic PCA models are used to represent the segments. The evaluation results suggest that the proposed approach can be applied to extract useful information from temporal databases. In our future work we plan to employ fuzzy clustering to model overlapping situation in car driving.

A group of researchers from the University of California (Keogh et al. [6]) reviewed the three major segmentation approaches in the literature (*Sliding Windows, Top-Down & Bottom-Up*) and provided an extensive empirical evaluation on a heterogeneous collection of datasets. In order to evaluate the segmentation algorithms, all datasets are represented as piecewise linear approximation (PLA), which refers to an approximation of a time series. Among all segmentation algorithms the bottom-up approach performed best, which corresponds to our own evaluation. Furthermore, they introduce a novel on-line algorithm for segmenting time series that scales linearly to the size of the data and produces high quality approximations.

In a survey about time series abstraction methods, Hoeppner [5] discusses data representations that have been used in literature. The author states that an abstracted data representation is necessary to account for the human way of perceiving time series. Moreover, the paper suggests to use multi-scale methods or rather multi-resolution analysis,

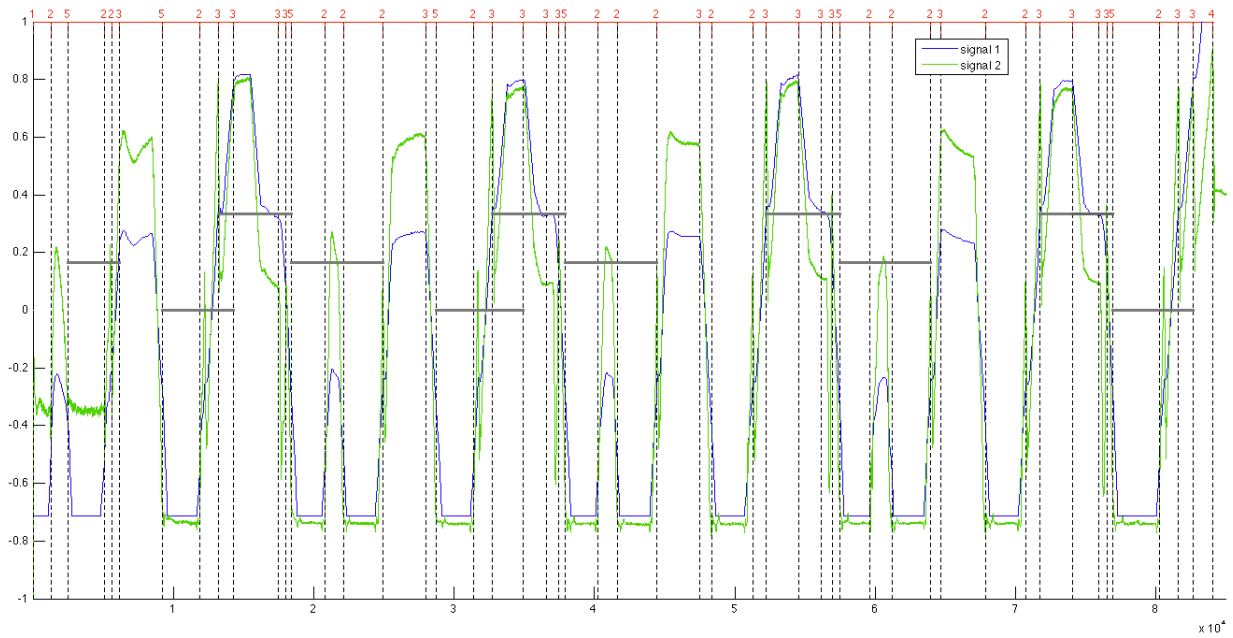


Figure 7: Time Series Segmentation and Clustering of Sensor Data from Vehicles
This plot combines the results of time series segmentation, segment clustering and sequence analysis. In the first step we employed the bottom-up (BU) segmentation approach to identify internally homogeneous time intervals. Whereas the merge threshold was determined according the growth of the segmentation cost (refer to Figure 5(a)). The determined segments are unequally distributed and represent individual situations that occur during a car drive (illustrated by vertical broken lines). In a second step we grouped the identified segments by means of agglomerative hierarchical clustering. To measure the similarity of the individual segments we extracted different signal features like standard deviation, minimum and maximum value, as well as ascent. The established feature vectors were compared by means of cosine similarity. For the examined time series data we decided to group the individual segments into 5 clusters (according to the cost function of the cluster tree; also refer to Figure 6). The cluster labels (highlighted in red color) are shown at the top of the respective segment borders. In the final step we retrieved cluster sequences that occur in regular time intervals. The cluster sequences are illustrated by horizontal bars, whereas identical sequences are plotted at the same height (highlighted in dark gray). In case of vehicular sensor data, sequences of segments can be regarded as complex drive manoeuvres, such as stop and go at traffic lights or overland drives.

which only considers features that persist over a broad range of scale and compensate dislocation effects. We aim to extend our work to multi-scale methods to capture features from segments or situations of different scales.

Most segmentation algorithms belong to the supervised learning family, where a labelled corpus is available to the algorithm in the learning phase. An unsupervised learning algorithm for segmenting sequences of symbols or categorical events is presented by Shani et al. [10], where the algorithm never sees examples of successful segmentation, but still discovers meaningful segments. The proposed algorithm computes a maximum likelihood segmentation, which is most appropriate to hierarchical sequences, where smaller segments are grouped into larger segments. One advantage of this probabilistic approach is that it allows one to suggest conditional entropy as quality measurement of a segmentation in absence of labelled data. The idea of measuring the quality of segmentation inspired our own performance evaluation on the basis of the employed cost function.

In many cases time series segmentation is used to recognize context from sensor data. For instance, recognizing the context of use is important in making mobile devices as simple to use as possible. The awareness of a user's situation can help the device and underlying services in providing an adaptive and personalized user interface. Himberg et al. [4] present a randomized variation of dynamic programming that minimizes the intra-segment variances and gives approximately optimal results. Although their approach is interesting, we are more fascinated by the idea of integrating our own segmentation and clustering algorithm into a real-life context-aware application.

The use of context in mobile devices is receiving increasing attention in mobile and ubiquitous computing research. Gellersen et al. [3] have investigated multi-sensor context-awareness in a series of projects and report experience from development of a number of device prototypes. In order to design context-aware systems, raw sensor data is transformed into 'cues' that incorporate various features based on simple statistics (e.g., standard deviation, quartile distance, etc.). The mapping from cues to context may be explicit, for instance when certain cues are known to be relevant indicators of a specific context, or implicit in the result of a supervised or unsupervised learning technique. Based on several software prototypes, Gellerson et al. [3] have shown that the integration of diverse sensors is a practical approach to obtain context representing real-life situations.

This paper describes an interesting use case of multi-sensor context-awareness and approves the relevance and topicality of our own foundational research.
Another considerable context-aware application is *CenceMe* [8], which infers the presence of individuals using sensor-enabled mobile phones and shares this information through social network applications (such as *Facebook* and *MySpace*). The *CenceMe* framework contributes to the design of light-weight classifiers, running on mobile phones, which realize a split-level classification paradigm. To infer the presence of individuals, audio data is classified using the unsupervised learning technique of discriminant analysis, where the feature vectors are composed of the mean and standard deviation of the DFT (Discrete Fourier Transform) power. In case of context-inference from accelerometer readings the mean, standard deviation, and the number of peaks per unit time are accurate feature vector components, providing high classification accuracy. Classification of accelerometer data is based on a decision tree technique (J48 algorithm of *WEKA* data mining workbench[1]).

6. CONCLUSION

This paper is concerned with time series segmentation and segment clustering. We proposed a SVD-based bottom-up algorithm that identifies internally homogeneous time series segments. To recognize recurring patterns, the established time series segments were grouped via agglomerative hierarchical clustering. Subsequently we retrieved recurring sequences of grouped segments, which can be considered as complex situations or higher-level context.
The proposed bottom-up segmentation extends the introduced sensor fusion algorithm [2] by several features, such as automatic determination of model rank and merge threshold. Furthermore, we evaluated the extended segmentation algorithm on several real-life datasets which comprise the progression of motor parameters during a car drive.
Our evaluation demonstrated that the proposed bottom-up (BU) approach performs better than the straightforward critical point (CP) segmentation. Additionally, we suggested to employ the segmentation cost to determine the merge threshold. Based on further experiments we showed how to choose a suitable number of clusters for grouping established time series segments.
In our future work, we plan to identify segment sequences that describe normal and abnormal drives or rather drivers, and support performance tuning of an engine.
Although this paper discusses time series segmentation and clustering in terms of multivariate sensor data recorded from vehicles, we explained that our approach is suitable for other domains, such as sensor data collected from smart phones. We described different real-life use cases and applications, where time series analysis is successfully applied to recognize higher-level context or complex situations.

7. ACKNOWLEDGEMENTS

The proposed pattern recognition and classification approach for multivariate time series was developed in cooperation with the Volkswagen AG, Wolfsburg. Thanks to Bernd Werther and Matthias Pries for providing sensor data from vehicles, and for their contribution of expert knowledge.

[1] `http://www.cs.waikato.ac.nz/~ml/weka/`

8. REFERENCES

[1] J. Abonyi, B. Feil, S. Nemeth, and P. Arva. Modified gath–geva clustering for fuzzy segmentation of multivariate time-series. *Fuzzy Sets Syst.*, 149:39–56, January 2005.

[2] J. Abonyi, B. Feil, S. Nemeth, and P. Avra. Principal component analysis based time series segmentation: A new sensor fusion algorithm. *Preprint*, 2004.

[3] H. W. Gellersen, A. Schmidt, and M. Beigl. Multi-sensor context-awareness in mobile devices and smart artifacts. *Mob. Netw. Appl.*, 7:341–351, Oct 2002.

[4] J. Himberg, K. Korpiaho, H. Mannila, J. Tikanmäki, and H. Toivonen. Time series segmentation for context recognition in mobile devices. In *Proceedings of the 2001 IEEE International Conference on Data Mining*, ICDM '01, pages 203–210, Washington, DC, USA, 2001. IEEE Computer Society.

[5] F. Höppner. Time series abstraction methods - a survey. In *Informatik beweqt: Informatik 2002 - 32. Jahrestagung der Gesellschaft fuer Informatik E.V. (GI)*, pages 777–786. GI, 2002.

[6] E. J. Keogh, S. Chu, D. Hart, and M. J. Pazzani. An online algorithm for segmenting time series. In *Proceedings of the 2001 IEEE International Conference on Data Mining*, ICDM '01, pages 289–296, Washington, DC, USA, 2001. IEEE.

[7] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney. Statistical properties of community structure in large social and information networks. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 695–704, New York, NY, USA, 2008. ACM.

[8] E. Miluzzo, N. D. Lane, K. Fodor, R. Peterson, H. Lu, M. Musolesi, S. B. Eisenman, X. Zheng, and A. T. Campbell. Sensing meets mobile social networks: the design, implementation and evaluation of the cenceme application. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, SenSys '08, pages 337–350, New York, NY, USA, 2008. ACM.

[9] S. J. Orfanidis. Savitzky-golay filtering. In *Introduction to Signal Processing*, Englewood Cliffs, NJ, 1996. Prentice-Hall.

[10] G. Shani, C. Meek, and A. Gunawardana. Hierarchical probabilistic segmentation of discrete events. In *ICDM'09*, pages 974–979, Washington, DC, USA, 2009.

[11] S. Spiegel, J. Kunegis, J. Clausen, and S. Albayrak. Link prediction on evolving data using tensor factorization. In *PAKDD2011: Proceeding of ACM International Workshop on Behavior Informatics (BI2011)*, New York, NY, USA, 2011. ACM.

[12] S. Spiegel, J. Kunegis, and F. Li. Hydra: a hybrid recommender system [cross-linked rating and content information]. In *CNIKM '09: Proceeding of the 1st ACM international workshop on Complex networks meet information and knowledge management*, pages 75–80, New York, NY, USA, 2009. ACM.

[13] G. Stephanopoulos and C. Han. Intelligent systems in process engineering: A review. In *Computational Chemical Engineering*, volume 20, pages 743–791, Miamisburg, OH, USA, 1996. Elsevier.