Comparing multidimensional sensor data from vehicle fleets with methods of sequential data mining

Julia Hartung¹ · Gabriele Gühring² · Valentin Licht³ · Alexander Warta³

Received: 5 November 2019 / Accepted: 10 March 2020 / Published online: 19 March 2020
© The Author(s) 2020

Abstract

Reading and understanding large amounts of sensor data from vehicle test drives becomes more and more important. In order to test vehicle components or analyze exhaust emissions in real test drives, the sensor data obtained from these test drives have to be comparable. Otherwise components or exhaust emissions are tested and analyzed under false conditions. The sensor data obtained during test drives are highly multidimensional which makes it even more complicated to identify recurring patterns. We present a process model to compare different test drives according to their sensor data and so give an answer to the question whether or not test drives in different cities, locations and environments are representative to real driving scenarios. The algorithms we use focus on segmentation of the individual multivariate test drive data and on clustering of the segments according to different methods. We present several segmentation and cluster methods and compare which of them is best suited for comparing test drives. The segmentation method we identified as best suited is based on principal component analysis. As cluster methods we examine hierarchical, partitioning and density-based clustering in detail.

Keywords Representativeness of time series · Principal component analysis · Segmentation · Multivariate time series · Clustering

Mathematical Subject Classification 68T10 · 62H30 · 62-02

1 Introduction

Analyzing and comparing multivariate vehicle sensor data has many applications. In order to compare test drives in various cities or environments according to the interaction of the different sensor data, recurring patterns have to be found, measured and in the best of all cases classified [1]. The aim of this paper is to present and evaluate several algorithms for this task. Based on cluster analysis the algorithm verifies whether the representativeness of certain test drives is provided. This is important for evaluating signs of wear and tear of certain components of specific

vehicles, as well as exhaust emission values. Without being representative of all urban test drives the obtained values would not be comparable. Finally our analysis helps in simulating vehicle sensor data as in [13], since it gives knowledge about vehicle dynamics and the generation of common and anomalous driving scenarios.

In order to find patterns in sequential data, different approaches can be found in the literature depending on whether the data in use is a time series or a sequence. Whereas a time series is an ordered list of numbers, a sequence is an ordered list of symbols [8]. For sequences or for time series, which are transformed to sequences,

✉ Gabriele Gühring, gabriele.guehring@hs-esslingen.de; Julia Hartung, julia.hartung@trumpf.com; Valentin Licht, valentin.licht@de.bosch.com; Alexander Warta, alexander.warta@de.bosch.com | ¹TRUMPF Laser GmbH, Schramberg, Germany. ²Hochschule Esslingen, Esslingen, Germany. ³Robert Bosch GmbH, Stuttgart, Germany.

sequence mining is used i.e. in [8, 23]. However, transforming a multivariate time series into a sequence is difficult. When a time series is transformed into a sequence a SAX algorithm is used (i.e. [20]), which works only on a time series in one dimension. In [34] therefore a principal component analysis is used to transform multi-dimensional time series data into one-dimensional time series data. They applied their algorithm to a three-dimensional times series data set of human motion. We take a different route based on the work by [1, 32] in order to find relevant patterns in multivariate sensor data. In this approach, a multivariate time series is divided in different subseries by the use of principal component analysis (PCA). We use PCA specifically for segmentation and not for dimensionality reduction. In order to compare the different data, after segmentation a cluster algorithm must be used. In general, time series clustering methods can be divided in time series clustering by features, model based time series clustering and time series clustering by dependence, see for example in [1, 2, 19], where clustering algorithms have been extensively studied. In our paper we investigate time series clustering by features and time series clustering by dependence and compare different cluster algorithms with different input data for the use of showing the representativeness or non-representativeness of certain test drives. The goal of our work is application-driven and specifically geared towards vehicle sensor data. In several papers vehicle sensor data is used to identify drivers i.e. [10]. However, our focus is not on identifying drivers, rather we want to find out about different types of segmentation and subsequent clustering, so that a comparison of different test drives is possible. We claim that a *real drive* is representative to the *test drive* if each cluster combines almost equally many segments of the *test drive* in Stuttgart as well as segments of the *real drives* in other cities. For a definition of *test drives* and *real drives* we refer to chapter 3, where the used data is explained in detail.

2 Related work

Finding representative time series or representative time series patterns has been a topic in many papers. Most of the time, this is done in order to classify a time series, i.e. class-specific representative patterns have to be found [9, 12, 27, 36]. Another motive to discover representative patterns in time series is the detection of anomalies or the forecasting of trends [11]. These patterns are sometimes called *motifs* and are detected by transforming the time series into a sequence of symbols. This can even be done in the multidimensional case [17, 34]. Finding ordinal patterns in one- or multidimensional time series data is another possibility to determine representative patterns in times series, see [30].

With our application scenario we are committed to vehicle sensor data. However, we do not want to identify drivers or a certain driving behaviour, such as it is done in [10, 18]. For our purpose it is not important how a certain pattern looks like, rather that it appears in a similar way in all of the reference data sets. The identification of certain driving scenarios is rather a by-product, i.e. Table 2, for our analysis. We use the three clustering algorithms agglomerative hierarchical clustering, *k*-means++ and Density Based Spatial Clustering of Applications with Noise (DBSCAN) as described in [22, 33] and [6] respectively and feed them with different input data, we obtained by segmentation of the test drives and the real drives. For the segmentation of the time series of the test drive and the real drives we use the method of Krzanowski in [16] as it was already done in [1, 32] or in a recent work for visual localization in [21].

3 Data and data preparation

The time series database we analyze, contains a large number of different vehicle sensor data of several plug-in hybrid vans. In the first step, we select a subspace of relevant signals based on expert knowledge and statistical properties as for example the correlation coefficient between individual sensor data time series. In this way, nine signals are selected and used for the analysis. For data pre-processing, among other things, a Savitzky–Golay filter is used [28]. It removes the noise by polynomial regression and tries to keep the characteristics of the distribution of the time series like minima, maxima and scatter as best as possible [4, 28]. In addition, pre-processing steps such as resampling, linear interpolation and z-score normalization are performed.

For an efficient runtime and a better overview only 4 h and 12 min of driving time are analyzed with a sampling rate of 0.1 s. Half of the data is taken from measurements on test vehicles, which are equipped with extensive measurement technology and driven by specially trained test drivers on a defined test track in public road traffic in the city of Stuttgart. This data is called *test drives*. In contrast, there are measurements of vehicle fleets in real operation in another large city. These measured journeys are carried out by normal drivers in everyday operation, we call this data *real drives*. In both cases the same plug-in hybrid vehicle type is used.

4 Segmentation of input data

In this and the following section the concept of pattern identification on input data and the aggregation of the data are investigated. The data analysis algorithm we perform focuses on the segmentation of the individual

multivariate driving data and is followed by the clustering of the segments according to different clustering methods. The multivariate time series are divided into smaller sections with a focus on identifying homogeneous time intervals and recurring segments. We use a bottom-up approach as in [15], therefore we start with a very fine segmentation of the multivariate time series, from which we fuse neighboring segments according to a prespecified rule.

In the bottom segmentation, the segments are separated based on extreme values of each individual signal. Other critical points could also be used for fine segmentation i.e. inflection points or saddle points. However, due to the number of dimensions and many extreme values of the signals, we believe that extreme values provide a bottom segmentation which is fine enough to start with (see also [32]). An exemplary signal with the marked extreme values is shown in Fig. 1. To prevent small fluctuations and residual noise from being evaluated as extreme points, the search algorithm is set to a span of 5 s. This means that a segment of an individual signal can not be smaller than 5 s. The value is based on the duration of typical change in a driving situation and was determined by try and error. The segment limits of individual signals are transferred to all signals afterwards. This results in bottom segments with an average duration of 0.8 s in our use case.

When combining the individual bottom segments, the effects of all dimensions as well as their correlations must be taken into account. Only by considering the overall condition of the vehicle, meaningful segments can be formed [1]. In order to consider all dimensions, a principal component analysis (PCA) is carried out on each segment resulting from the fine segmentation. The reconstruction error of the principal component projection as in (1) is used as the basis for the cost function of the segment fusion.

The bottom segmentation results in n non-overlapping time intervals. A segment of the data is defined by $s_i(a_i, b_i)$ with $1 \leq i \leq n$, where $a_1 = 1, a_i = b_{i-1} + 1$ and $b_n = N$ with

N being the total number of all data points in the examined vehicle data. The principal component projection p_j transfers a D dimensional data vector x_j into a m -dimensional subspace. The cost function $error(s_i(a_i, b_i))$ is calculated as the reconstruction error, which is defined as the square distance between the original values of the time series and the projected values.

$$error(s_i(a_i, b_i)) = \frac{1}{b_i - a_i + 1} \sum_{j=a_i}^{b_i} (x_j - p_j)^2 \quad (1)$$

By comparing the error of the individual segments $s_i(a_i, b_i)$ and $s_{i+1}(a_{i+1}, b_{i+1})$ with the error of the larger segment $s_{i_v}(a_i, b_{i+1})$ containing the two neighboring segments with starting point a_i and endpoint b_{i+1} , information about the homogeneity of neighboring time series segments can be obtained. Combining the segments, in cases where the error in the linked segments is smaller than the sum of the errors of the individual segments, will merge very similar segments [32]. This way a fusion of the segments takes place, whenever

$$error(s_{i_v}(a_i, b_{i+1})) < (error(s_i(a_i, b_i)) + error(s_{i+1}(a_{i+1}, b_{i+1}))) \cdot M. \quad (2)$$

M denotes a threshold which can also be allowed to be greater than 1, i.e. $M > 1$. Since the bottom segmentation already forms homogeneous segments, it is sufficient to use only a few principal components (PC) for a meaningful projection of the individual segments. With the vehicle data we used, a sufficient cumulative variance of more than 97% is already achieved with $m = 2$ principal components.

Different approaches are possible to compare and summarize the neighboring segments of the fine segmentation with the error function defined in (1). On the one hand it is possible to use an iterative bottom-up algorithm which compares the neighboring segments in iterative passages and fuses them if necessary. On the other hand

Fig. 1 Small section of one-dimensional signal with marked extreme values after pre-processing

an algorithm based on the sliding-window approach was tested in our work, which, starting with the first segment, enlarges a segment by further segments until a threshold limit is exceeded. If this occurs, the process repeats with the next segment that is not contained in the newly combined segment. When a segment is not combined with the following segment, it is assumed that there is a boundary between the two segments that does not have homogeneity of signals and signal correlations (see also [1] and [15]). An advantage of this algorithm is that it runs through all of the time series only once from left to right and therefore the entire time series does not already have to be considered at the beginning of the segment fusion. This makes it an algorithm which is also online-capable, as it is stressed out by [15]. In addition, our tests have shown that a better homogeneity of the segments is achieved, which is why the non-iterative algorithm based on sliding-window approach is used for the fusion of our segments. For a better understanding, we refer to the pseudo-code in Algorithm 1.

Algorithm 1: Segment Fusion

1. **Input:** $S = [s_1; s_2; \dots; s_n]$ stream of input segments, with each $s_i = [a_i; b_i; \text{error}(a_i; b_i; m)]$ where a_i is the starting time and b_i the end time of segment s_i and $\text{error}(a_i; b_i; m)$ the reconstruction error of s_i when projected to the first m principal components
2. **Constants:**
 - $m = 2$ number of principal components
 - $M = 1.03$ threshold for segment fusion
3. **Initialize** $i = 1$
4. **for** $j = 1$ **to** $n - 1$ **do**

$\text{loss_fusion} = \text{error}(a_i; b_{j+1}; m)$

if $(\text{error}(a_i; b_i; m) + \text{error}(a_{j+1}; b_{j+1}; m)) \cdot M \geq \text{loss_fusion}$

$b_i = b_j$

$\text{error}(a_i; b_i; m) = \text{loss_fusion}$

$\tilde{s}_i = [a_i; b_i; \text{error}(a_i; b_i; m)]$

else

$\tilde{s}_i = s_i$

$i = j$

5. **Output** $\tilde{S} = [\tilde{s}_1; \tilde{s}_2; \dots; \tilde{s}_k]$ stream of output segments, where again each $\tilde{s}_i$ is defined of three components $\tilde{s} = [a_i; b_i; \text{error}(a_i; b_i; m)]$

For the segment fusion a threshold value of $M = 1 + 3\%$ is defined, such that the reconstruction error of the merged segment may exceed the sum of the errors of the individual segments by 3%. The threshold value is determined by comparing the results of different threshold values. In our case the number of segments is reduced to about one third compared to the segment division after fine segmentation. Figure 2 shows sections of exemplary signals with their defined segment boundaries.

5 Input parameter for clustering

After segmentation of the vehicle sensor data for the test drives and the real drives we use clustering in order to find patterns, which can be identified with a certain driving manoeuvre. Thus, the previously aggregated segments are regarded as input parameters for the various cluster procedures, which cluster the segments of the test drives as well as the segments of the real drives in one algorithm. A distinction must be made between similarity-based and

Fig. 2 Sections of exemplary pre-processed signals with segment boundaries (yellow) after segment fusion

feature-based clustering. Similarity-based clustering, i.e. hierarchical clustering, requires a distance matrix as input parameter. The $n \times n$ -distance matrix, which is also called independence matrix, contains the distances between the individual data segments. The parameter n reflects the number of data segments, which are already fused, of the data sets [14]. In feature-based clustering, an $n \times D$ -feature matrix, also known as design matrix, is used. The parameter n represents again the number of data segments and D represents the dimensionality of each data segment, as is pointed out in [14]. Since different cluster methods are compared in Sect. 6 of this article, the input parameters for the different cluster methods must be created, which is described in more detail in Sects. 5.1 and 5.2.

5.1 Distance matrix

Similarity-based clustering requires a distance matrix. The distances between the individual segments can be calculated in several ways.

A method developed by [16] for measuring the distance of two segments uses again principal component analysis for a holistic approach. The PC reflect the data alignment and thus enable a similarity comparison of the segments. The data sets to be compared must contain the same number of D input dimensions but are allowed to have different lengths. They can then be compared using the similarity factor S_{PCA} calculated as in [31].

$$S_{PCA} = \frac{1}{m} \text{trace}(L^T M M^T L) \quad (3)$$

where m is the number of the relevant principal components in both segments and the $(D \times m)$ -matrices L and M contain the PC as column vectors for the comparing segments s_i and s_{i+1} respectively. Since the PC are normalized to the length of one, the scalar product $L_r \cdot M_s$ of their column vectors L_r and M_s can always be simplified to

$$L_r \cdot M_s = \cos(\theta_{L_r M_s}) \quad (4)$$

where $\theta_{L_r M_s}$ denotes the angle between the PC vectors L_r and M_s . Thus the formula of S_{PCA} can be written as

$$S_{PCA} = \frac{1}{m} \sum_{r=1}^m \sum_{s=1}^m \cos^2(\theta_{L_r M_s}) \quad (5)$$

In our application we use $m = 2$, so the first two PC of each segment are used for our calculations.

The advantage of Krzanowski's methodology is that the similarity between the data segments is described by a single number in which all dimensions as well as all data points are taken into account. In the following, this kind of input parameter is referred to as *distance matrix PCA*.

An alternative way is to build the distance matrix on the basis of feature vectors. There the distances are determined by the euclidean distance between the vectors f_i containing several features of segment s_i , i.e. mean values, standard deviations, minimum values, maximum values, etc. of each of the D dimensions, and the feature vector f_j containing the same features of segment s_j .

$$d_{eucl.} = ||f_i - f_j||_2 = \sqrt{\sum_k (f_{ik} - f_{jk})^2} \quad (6)$$

In the following, this input parameter is called *distance matrix dimension features*.

5.2 Feature matrix

In contrast the feature-based clustering e.g. k -means and DBSCAN is based on a feature matrix [14]. This matrix is created by linking the feature vectors of the segments. For a better illustration and interpretation of the cluster divisions, only two features for each dimension based on statistical values are calculated and linked in a feature vector in our use case. Two different definitions for the feature vectors are tested in this paper and the results are compared after clustering the feature vectors. One definition contains the statistical values mean and standard deviation of each of the D different dimensions for each segment s_i . This input parameter is called *feature matrix dimension* in the following.

Alternatively, a feature vector definition based on the principal components of the segments is used. For each segment the PC are calculated first. Statistical values are then formed based on the first PC and linked in a feature vector. In this case the statistical values median and kurtosis are used, because the PC were formed on standardized values and thus the mean value always assumes the value zero. It results in a two-dimensional feature-vector. This input parameter is named *feature matrix PCA*.

6 Clustering

Clustering comprises the grouping of similar objects which require a high interclass similarity and a low intraclass similarity, see for example [29]. There are different types of cluster paradigms, i.e. hierarchical, partitioning and density-based clustering as it is described in [32]. In this paper we investigate in detail the following clustering methods

- agglomerative hierarchical clustering [22],
- k -means++ [33] and
- Density Based Spatial Clustering of Applications with Noise, called DBSCAN [6].

They are based on the various distance or feature matrices explained in Sect. 5. In the end, it must be decided which cluster method in combination with the proposed distance or feature matrices is best suited for comparing test drives and measure their representativeness in different cities [35]. Thus each of the three clustering methods works on the respective distance matrices or on the respective feature matrices which are calculated out of segments from the test drives and the real drives. We claim that a real drive is representative to the test drive if each of the obtained clusters contains segments from the test drives from Stuttgart and the real drives from other cities.

Depending on each of the three cluster methods, hyperparameters must be defined. For example, in agglomerative hierarchical clustering or k -means++ the number of clusters must be defined in advance. We will apply necessary algorithms for the selection of optimal hyperparameters. The various cluster methods are then compared and evaluated on the basis of statistical cluster values, graphical cluster visualization and intra- and inter-polar cluster distances in Sect. 7.

6.1 Hierarchical clustering

Hierarchical clustering is divided into the two essential approaches of divisive and agglomerative clustering [14]. Since, in our case, smaller clusters are intended, agglomerative clustering is chosen. In hierarchical clustering, the linkage method for calculating the cluster distance as well as the number of clusters to be formed must be specified as hyperparameters.

Agglomerative clustering acts according to the “rich get richer” behaviour, which has the consequence that cluster sizes are often unbalanced. This effect is additionally favored by the cluster distance calculation according to the average method and is best avoided by the ward linkage method, see for example [37]. In addition, the clustering algorithm operates according to the greedy principle, which means that only the next step is considered when forming clusters. Within the framework of cluster analysis, clusters should be as compact and meaningful as possible. We follow therefore the investigations of [26], where the two linkage methods average and ward are preferred. The average method is well suited for non-euclidean distances and for data sets containing outliers. The calculation is based on the average distance $d_{av}(u, v)$ of all element pairs of two clusters u and v with accordingly n_u and n_v elements [14].

$$d_{av}(u, v) = \frac{1}{n_u n_v} \sum_{i=1}^{n_u} \sum_{j=1}^{n_v} d(u_i, v_j) \quad (7)$$

With the ward method, also called the minimum variance method, the distance between two clusters u and v is defined as the increase of the error sum of squares (ESS) by merging the two clusters u and v [37].

$$d_{ward}(u, v) = ESS_{uv} - ESS_u - ESS_v \quad (8)$$

where the respective ESS for the single cluster with $c = u$ and $c = v$ and the merged cluster with $c = uv$ is defined by

$$ESS_c = \sum_{i=1}^{n_c} x_i^2 - \frac{1}{n_c} \left(\sum_{i=1}^{n_c} x_i \right)^2 \quad (9)$$

for x_i being an element of cluster c . The ward method is mainly used to form small and compact clusters. Because it is not clear which of the two linkage methods is best suited for comparing the test drives of different cities, both, the average and the ward method are used in this paper. The distance calculation according to the ward method can only be applied if the segment distance is calculated as euclidean distance. Therefore, only the distance matrix calculated on the basis of the dimension feature vectors using the euclidean distance is used as the input parameter for the method with ward linkage. For average linkage, both distance matrices presented in Sect. 5.1 are used. In chapter 8 their results are compared.

To determine the optimum number of clusters, the cost function of the dendrogram (Fig. 3) is used. It subtracts the interclass similarity of each cluster from the intraclass similarity of all clusters, thus making the costs for combining two clusters visible. The limit for a certain number of clusters is set, whenever there is a noticeable jump from small intraclass distances to large intraclass distances. This point can be recognized graphically as an elbow in the cost function, shown in Fig. 4 for the hierarchical clustering with average linkage on the first two PC at a value of 5 clusters. Thus the cut is selected in such a way that within cluster-like segments can be grouped, which have a small distance to each other. Between the clusters a high dissimilarity and thus large distances is aimed at.

6.2 K-means++

The k -means algorithm has prevailed in various studies against the other participating cluster methods, see for example [26]. However, the most common disadvantage of this algorithm is the randomly chosen starting point,

Fig. 3 Dendrogram, hierarchical clustering, average linkage, distance matrix

Fig. 4 Costfunction dendrogram, hierarchical clustering, average linkage, distance matrix PCA

which is improved by the *k*-means++ algorithm. The *k*-means++ algorithm optimizes the selection of starting points by increasing the probability of distant data points being initial cluster centers [3]. For this reason *k*-means++ is used for our use case.

The optimal number of clusters can be determined using the scree test, also called elbow test, which compares the variability of the clusters with the number of clusters. The cluster variability is defined by the average distance of the cluster values to the cluster center [25]. The optimal number of clusters exists when the variability has decreased and no longer changes significantly when more clusters are formed. This position is characterized by an elbow in the scree plot. In our case the cluster variability is calculated for 1, 2, ..., 9 clusters. Then the values are plotted

Fig. 5 Scree plot with optimal number of clusters, *k*-means++, feature matrix PCA

in the scree plot and the elbow is located automatically via the maximum point of the 2nd derivation of the variability of the clusters as in Fig. 4.

In Fig. 5 for the *k*-means++ on PC values, the optimal number of clusters would be $k = 2$, because there a clear kink in the function is recognizable. However, since 2 clusters represent a very small cluster division and the scattering within a cluster continues to decrease, a further bend is searched for. It is located at $k = 4$ clusters.

6.3 DBSCAN

Density-based clustering is suitable for applications where the clusters cannot be described by a slight dissimilarity within the cluster [7]. In DBSCAN two data points are considered to be connected in one cluster if there is a chain of

dense data points connecting them. The DBSCAN cluster method requires the parameters Eps , a distance for the neighborhood of datapoints and $MinPts$, which specifies the number of datapoints in a neighborhood of maximum distance Eps [6]. A data point is called dense if there exist at least $MinPts$ in its Eps neighborhood. DBSCAN often requires the knowledge of domain experts for a good parameter selection, since the parameters depend on the investigated database [5]. We use a heuristic approach to calculate these hyperparameters.

The parameter Eps is defined according to the k -nearest neighbor methodology. The distances of the data points to the k -nearest neighbors are calculated, sorted by size and plotted in a diagram. The optimal value for Eps is located at the position of the elbow of the kNN distance function, shown in Fig. 6, where the DBSCAN algorithm is applied to the *feature matrix PCA*. There the elbow is located at the 19th segment, which leads to a distance value for $Eps = 2.58$. We use the same method when applying the DBSCAN algorithm to the *feature matrix dimension*.

The hyperparameter $MinPts$ can be set to the value 4 as in [6]. Another approach to calculate the parameter is using the formula $\ln(n)$ where n reflects the number of

segments to be grouped [5], which are in our case 11,357 segments. The defined value for $MinPts$ is also used for the parameter k of the kNN algorithm to determine the parameter Eps . The definition of $MinPts = 4$ according to [6] was originally done for the two-dimensional space. Since our feature vector on a segment signal basis has far more than two dimensions, we use $MinPts = \ln(n)$, which leads to a value of $MinPts = \ln(11,357) \approx 9$.

6.4 Cluster models

Within the scope of this work, different models are created and their cluster results are compared. The models differ in the choice of the cluster method and the type of segment description. The different combinations are shown in Table 1. The definition of the hyperparameters is implemented for all models according to the same procedure depending on the cluster method presented above. The number of clusters definition is exemplary shown for the hierarchical clustering with average linkage on the distance matrix PCA in Fig. 4 and for the k -means++ on the feature matrix PCA in Fig. 5. DBSCAN calculates the number of clusters during clustering based on the hyperparameters Eps and $MinPts$. The definition of Eps is shown in Fig. 6 for the algorithm on the feature matrix PCA. The number of clusters is not specified in any method by the user.

7 Data interpretation

The formed clusters and their characteristics are visualized with different methods, e.g. histogram and box plot.

When interpreting the cluster results of the different models, the clusters formed on the basis of a distance matrix with dimensional features can more easily be described by known driving manoeuvre than the clusters which were calculated by models based on PCA. In order to establish a direct relationship between the PC and the original signals, the composition of the PC would have to be recalculated. However, the PC describe the segments more precisely than the dimension feature vectors.

Fig. 6 kNN distance function with optimal Eps , DBSCAN, feature matrix PCA

Table 1 Cluster model combinations

Method	Distance calc.	Input parameter	Num clust.
Aggl. hierarchical	Average	Distance matrix PCA	5
	Average	Distance matrix dimension features	4
	Ward	Distance matrix dimension features	6
k -means++		Feature matrix PCA	4
		Feature matrix dimension	5
DBSCAN		Feature matrix PCA	1
		Feature matrix dimension	1

Fig. 7 Hierarchical, average linkage, distance matrix PCA

The model of hierarchical clustering with average linkage on the signal features and both DBSCAN models all form only one main cluster which contains most of the data. The models may show that all data are very similar, but they are useless overall for showing the representativeness of certain test drives or driving characteristics. We therefore suggest that models with more balanced cluster ratios should be used, as is the case in any of the other models. The cluster shares of the test drive data and the fleet data are shown in Figs. 7, 8, 9, 10, 11, 12, 13. On the x-axis, a distinction is made between the internal test drives and the fleet data, while the proportion of segments per cluster is represented by the bars on the y-axis.

While all of the clusters are constructed only with analytical methods of segmentation and clustering, they can be characterized by their different signal behavior. When interpreting for example the division of the torque signal, statements can be made about the vehicle acceleration within the clusters. So each cluster shows different driving characteristics and behaviors. As an example we present in Figs. 14, 15, 16, 17, 18, 19 box plot diagrams of different signals divided into the clusters according to *k*-means++ based on dimension features. This algorithm is user-friendly because driving situations can be easily allocated.

The signal values are divided into clusters 1–5 on the x-axis of the box plot diagrams. In Fig. 14 one can see that, while acceleration is strong in the second cluster, it is usually zero in the first and third clusters. The zero values indicate a vehicle standstill or a constant speed. In Fig. 15 the negative torque values in cluster 4 indicate regenerative braking processes. Similar to these analysis, further statements about the driving situations could be obtained on the basis of the box plot diagrams of the other signals. Table 2 contains the summarized cluster interpretation based on the box plot diagrams of the different signals.

Fig. 8 Hierarchical, average linkage, distance matrix dimension features

Fig. 9 Hierarchical, ward linkage, distance matrix dimension features

Fig. 10 *k*-means++, feature matrix PCA

Fig. 11 *k*-means++, feature matrix dimension

Fig. 12 DBSCAN, feature matrix PCA

Fig. 13 DBSCAN, feature matrix dimension

Fig. 14 Acceleration, clustering with *k*-means++, feature matrix dimension

Fig. 15 Mechanical torque, clustering with *k*-means++, feature matrix dimension

Fig. 16 Voltage, clustering with *k*-means++, feature matrix dimension

8 Results and evaluation

Our results show that the representativeness of a particular test drive compared to fleet data can be determined by using the following cluster methods based on a distance or feature matrix of constructed segments:

- aggl. hierarchical clustering, average linkage, distance matrix calculated on PCA values

Fig. 17 Filtered effective current, clustering with *k*-means++, feature matrix dimension

Fig. 18 Rotational speed, clustering with *k*-means++, feature matrix dimension

Fig. 19 Current, clustering with *k*-means++, feature matrix dimension

- aggl. hierarchical clustering, ward linkage, distance matrix calculated on dimension values
- *k*-means++, feature matrix calculated on PCA values
- *k*-means++, feature matrix calculated on dimension values

If each cluster consists of similar parts of the test drives and the fleet data, the representativeness is confirmed. Otherwise, it is shown which cluster is not present in a certain time series or test drive.

The runtime of hierarchical clustering is much worse than that of the other applied cluster methods. Its greatest strength, which involves working out hierarchical structures of the data, finds little usage in our use case. Although the developed methodology does not represent a real-time-critical application, it should be able to analyze large amounts of data. Therefore, the performance of the cluster methods must not be ignored.

The *k*-means++ algorithm based on feature vectors is strongly oriented to the individual signal values when clustering. This makes the algorithm very user-friendly, since known driving situations can be easily recognized and abstracted by the user. The *k*-means++ algorithm based on the features of the principal components has abstracted the segment description by extracting and aggregating the data to such an extent that a meaningful interpretation and assignment to driving manoeuvre is more difficult. In addition, the interaction between the individual signals is taken into account and the cluster classifications cannot be easily traced back to individual signals. Due to the holistic approach this method is best suited for comparing the test drives and the fleet data and makes the most exact statement about their similarity.

9 Conclusion

The results show that the representativeness of our test drive data was confirmed by all examined cluster models except for the two DBSCAN models and hierarchical clustering with average linkage on the signal features, which

Table 2 Cluster interpretation *k*-means++, feature matrix dimension

Cluster	Characteristics	Interpretation
Cluster 1	Decreasing/lower voltage values Often approximately zero in torque, acceleration, speed, filtered effective current	Stand, roll, slow driving
Cluster 2	Torque, acceleration and current in the higher value range	Drive, acceleration
Cluster 3	Decreasing/lower voltage values Often approximately zero in torque, acceleration, speed, filtered effective current	Stand, roll, slow driving
Cluster 4	Higher speed Low/negative values in current, torque, acceleration	Regenerative braking, braking
Cluster 5	Torque and current distributed over the full value range	Constant travel, propulsion

we called *distance matrix dimension features*. These three models can not be used for an analysis since all of them result in only one cluster, which could not be associated with a driving scenario. In our use case the user's easiest to interpret description was achieved with *k*-means++ algorithm on the *feature matrix dimension*, whereas the most comprehensive analysis is achieved with *k*-means++ based on *feature matrix PCA*.

In this paper, the description of the segments was implemented using two alternative approaches. On the one hand, the segments were described on the basis of signal features. In order to make the dimensions smaller and more descriptive and to make the result more interpretable, the feature selection was limited to two features. In a further analysis, other or more features could also be considered. For further literature see [24]. The second approach in this paper is to reduce the signal dimensions by PCA. Subsequently, the PC are described by statistical quantities. Here, too, further quantities can be used for a more precise description.

By analyzing the cluster characteristics different groupings within the time series data can be identified. The developed methodology in this paper is heuristic and can therefore be used for further analysis. It is suitable for the verification of representativeness and for a general comparison of multivariate time series data. It is also possible to compare more than two different input data sources. By grouping the data into different clusters, the procedure also provides information about different characteristics of the time series segments. Finally, our analysis helps simulating vehicle sensor data since it gives knowledge about vehicle dynamics and the generation of common and anomalous driving scenarios.

Acknowledgements Open Access funding provided by Projekt DEAL. This research was supported by Robert-Bosch GmbH. We thank all of the colleagues from there who provided insight and expertise that greatly assisted the research.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Abonyi J, Feil B (2007) Cluster analysis for data mining and system identification. Birkhäuser, Basel
2. Aghabozorgi S, Shirkhorshidi A, Wah T (2015) Time-series clustering—a decade review. *Inf Syst* 53:16–38. <https://doi.org/10.1016/j.is.2015.04.007>
3. Arthur D, Vassilvitskii S (2007) K-means++: the advantages of careful seeding. In: Proceedings of the eighteenth annual ACM-SIAM symposium on discrete algorithms, Society for Industrial and Applied Mathematics, Philadelphia, pp 1027–1035
4. Azami H, Saraf M, Mohammadi K (2014) A new signal segmentation approach based on singular value decomposition and intelligent Savitzky–Golay filter. In: Movaghar A, Jamzad M, Asadi H (eds) Artificial intelligence and signal processing. Springer, Cham, pp 212–224
5. Birant D, Kut A (2007) ST-DBSCAN: an algorithm for clustering spatial temporal data. *Data Knowl Eng* 60:208–221. <https://doi.org/10.1016/j.datak.2006.01.013>
6. Ester M, Krieger H-P, Sander J, Xu X (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the second international conference on knowledge discovery and data mining, pp 226–231
7. Ester M, Sander J (2000) Knowledge discovery in databases—Techniken und Anwendungen. Springer, Berlin
8. Fournier-Viger P, Lin C, Kiran R, Koh Y, Thomas R (2017) A survey of sequential pattern mining. *Data Sci Pattern Recognit* 1:54–77
9. Grabocka J, Schilling N, Wistuba M, Schmidt-Thieme L (2014) Learning time-series shapelets. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining, pp 392–401
10. Hallac D, Sharang A, Stahlmann R, Lamprecht A, Huber M, Roehder M, Sosic R, Leskovec J (2017) Driver identification using automobile sensor data from a single turn. In: IEEE 19th international conference on intelligent transportation systems (ITSC). <https://doi.org/10.1109/ITSC.2016.7795670>
11. Hao M, Marwah M, Janetzko H, Sharma R, Keim D, Dayal U, Patnaik D, Ramakrishnan N, (2011) Visualizing frequent patterns in large multivariate time series. In: Wong PC (ed) Visualization and data analysis 2011. IST/SPIE electronic imaging. San Francisco, CA, SPIE, pp 78680J–78680J-10. <https://doi.org/10.1117/12.872169>
12. Hills J, Lines E, Baranauskas E, Mapp J, Bagnall A (2014) Classification of time-series by shapelet transformation. *Data Min Knowl Disc* 28:851–881
13. Hirsenkorn N, Hanke T, Rauch A, Dehlink B, Raschhofer R, Biehl E (2016) Virtual sensor models for real-time applications. *Adv Radio Sci* 14:31–37. <https://doi.org/10.5194/ars-14-31-2016>
14. Iguar L, Seguí S (2017) Introduction to data science—a python approach to concepts, techniques and applications. Springer, Cham
15. Keogh E, Kasetty S (2003) On the need for time series data mining benchmarks: a survey and empirical demonstration. *Data Min Knowl Disc* 7:349–371. <https://doi.org/10.1023/A:1024988512476>
16. Krzanowski W (1979) Between-group comparison of principal components. *J Am Stat Assoc* 74:703–707. <https://doi.org/10.1080/01621459.1979.10481674>
17. Laftchiev E, Yuchao L (2018) Finding multidimensional patterns in multidimensional time series. In: KDD workshop on MiLeTS'18, London
18. Lestyán S, Acs G, Biczók G, Szalay Z (2019) Extracting vehicle sensor signals from CAN logs for driver re-identification. *ICI SSP*

19. Liao WT (2005) Clustering of time series data—a survey. *Pattern Recogn* 38:1857–1874. <https://doi.org/10.1016/j.patcog.2005.01.025>
20. Lin J, Keogh E, Wei L, Lonardi S (2007) Experiencing SAX: a novel symbolic representation of time series. *Data Min Knowl Disc* 15:107–144
21. Lowry S (2019) Similarity criteria: evaluating perceptual change for visual localization. In: European conference on mobile robots (ECMR), Prague, Czech Republic, pp 1–6
22. Müllner D (2011) Modern hierarchical, agglomerative clustering algorithms, 1–29. arXiv e-prints [arXiv:1109.2378](https://arxiv.org/abs/1109.2378)
23. Mooney C, Roddick J (2013) Sequential pattern mining—approaches and algorithms. *ACM Comput Surv* 5:1–46
24. Nanopoulos A, Alcock R, Manolopoulos Y (2001) Features-based classification of time-series data. *Int J Comput Res* 10:49–61
25. Ng A, Soo K (2017) *Data science—was ist das eigentlich?!*. Springer, Berlin
26. Punj G, Stewart D (1983) Cluster analysis in marketing research: review and suggestion for application. *J Mark Res* 20:134–148. <https://doi.org/10.1177/002224378302000204>
27. Rakthanmanon T, Keogh E (2013) Fast shapelets: a scalable algorithm for discovering time series shapelets. In: *Proceedings of SDM*
28. Savitzky A, Golay M (1964) Smoothing and differentiation of data by simplified least squares procedures. *Anal Chem* 36:1627–1639
29. Shearer C (2000) The CRISP-DM model: the new blueprint of data mining. *J Data Warehous* 5:13–22
30. Shekar A, Pappik M, Iglesias S, Müller E (2018) Selection of relevant and non-redundant multivariate ordinal patterns for times series classification. In: Soldotova L, Vanschoren J, Papadopoulos G, Ceci M (eds) *Discovery science, Lecture Notes in Computer Science*, Springer, Cham pp 224–240
31. Singhal A, Seborg D (2001) Matching patterns from historical data using PCA and distance similarity factors. In: *IEEE proceedings of the 2001 American control conference*, pp 224–240
32. Spiegel S, Gaebler J, Lommatzsch A, De Luca E, Albayrak S (2011) Pattern recognition and classification for multivariate time series. In: *Proceedings of the fifth international workshop on knowledge discovery from sensor data*, ACM, New York, NY, USA pp 34–42
33. Stuart PL (1982) Least squares quantization in PCM. *IEEE Trans Inf Theory* 28:129–136
34. Tanaka Y, Iwamoto K, Uehara K (2005) Discovery of time-series motif from multi-dimensional data based on MDL principle. *Mach Learn* 58:269–300
35. Wagner M, Mohammed Z (2014) *Data mining and analysis—fundamental concepts and algorithms*. Cambridge University Press, New York
36. Wang X, Lin J, Senin P, Oates T, Gandhi S, Boedihardjo A, Chen C, Frankenstein S (2016) RPM: representative pattern mining for efficient time series classification. In: *Proceedings of the 19th international conference on extending database technology (EDBT)*, March 15–18, Bordeaux, France. <https://doi.org/10.5441/002/edbt.2016.19>
37. Ward J Jr (1963) Hierarchical grouping to optimize an objective function. *J Am Stat Assoc* 58:236–244

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.