Classification of Car Classes Based on Combustion Engine Sound

Students Project

Seo Jiwon, BSc

Supervisor: Dr. Alois Sontacchi

Graz, October 18, 2017

institut für elektronische musik und akustik

Abstract

Since the launch of electric vehicles, the sound design for those silent vehicles has become a highly important issue nowadays, as it is required to have a suitable exterior sound to alert pedestrians to approaching vehicles as well as an interior one for driver's recognition of the driving situation.
One widely applicable method for the engine sound design is to utilise the knowledge of conventional combustion engine sounds. A engine of a car also generates an appropriate sound corresponding to its specific car class. Is it therefore possible to distinguish different car classes by their engine sound characteristics? The aim of this work is to answer this question, i.e. to classify the car classes based on the engine sounds specified by relevant psycho-acoustic parameters.
After pre-processing the given data, including normalisation and unnecessary parameter removal, the parameters with similar behaviour are grouped by means of PCA. The feature subset is then generated in such a way that one parameter is selected in each group, and then the classifier is constructed after applying LDA. The classifiers that distinguish specific classes are then evaluated according to their performance and robustness. Lastly, potential future works are proposed based on limitations of this work.

Zusammenfassung

Seit der Markteinführung von Elektrofahrzeugen ist das Sound-Design für diese leisen Fahrzeuge zu einer wichtigen Thematik geworden, weil es notwendig ist, dass sowohl ein passender äußerer Klang, zur Warnung von Fußgängern über sich nähernde Fahrzeuge, als auch ein passender innerer Klang, um den Fahrer über die Fahrsituation zu informieren, vorhanden ist.
Eine weitgehend anwendbare Methode für das Motorklang-Design ist es, das Wissen über den Klang der konventionellen Verbrennungsmotoren zu verwenden. Der Motor eines Fahrzeuges erzeugt einen passenden Klang entsprechend seiner Fahrzeugklasse. Ist es dann möglich, die verschiedenen Fahrzeugklassen durch ihre klanglichen Merkmale zu unterscheiden? Ziel dieser Arbeit ist es, diese Frage zu beantworten, d.h. die Fahrzeugklassen anhand der Motorklänge, die durch relevante psychoakustische Parameter spezifiziert sind, zu klassifizieren.
Nach der Vorverarbeitung der vorgegebenen Daten, einschließlich der Normalisierung und der Parameterentfernung, werden die Parameter mit ähnlichem Verhalten mittels PCA gruppiert. Die Merkmalsuntermenge wird dann so erzeugt, dass in jeder Gruppe ein Parameter ausgewählt wird und dann der Klassifikator nach LDA aufgebaut wird. Die Klassifikatoren, die bestimmte Klassen unterscheiden, werden dann nach ihrer Leistung und Robustheit ausgewertet. Schließlich werden mögliche zukünftige Arbeiten auf der Grundlage von Einschränkungen dieser Arbeit vorgeschlagen.

Contents

1 Introduction

Since the launch of electric vehicles into the automotive market, diverse advantages have been emphasised in using them. In addition to well-known benefits such as being environment-friendly and the light body weight, one of the most important advantages is that it rarely generates loud noise, i.e. it is highly silent compared to the conventional combustion engine car. However, too quiet cars are not always desirable. The low exterior sound level can put pedestrians in danger when driving at low speeds [GM14]. Actually, the U.S. Department of Transportation National Highway Traffic Safety Administration specified the minimum sound power for hybrid and electric cars [oTNHTSA13]. Besides, the electric vehicle also needs to have an appropriate interior sound during driving. The interior sound affects loudness adaptation according to the driving situation, such as load-dependency [KSFFG14]. Additionally, the perception of acceleration situation is related with pitch shifting [YF14]. According to [SKvdPW14], the interior sound of the combustion engine is more sporty compared to the sound of an electric engine. Thus, it is necessary to generate a sporty interior sound for an electric car which is supposed to sound sporty. Therefore, the above-mentioned claims justify that the proper engine sound should be artificially generated by using active sound generator.
However, since there is not enough experience yet in designing engine sounds in electric vehicles [GF14], it is still unclear how the sound is supposed to be. It could be a sound of a conventional combustion engine or a futuristic sound synthesised by a novel way [GYS14], [CP16]. One thing is certain that we already have been familiar with the sound of combustion engines. Therefore, we are willing to utilise the knowledge about combustion engine sounds and also to apply it for generating the electric vehicle sound [FSBH14].
When adopting the combustion engine sound for engine sound design of the electric cars, following questions are raised; *How do cars sound corresponding to a belonging car class? Are there any distinctive characteristics of the engine sound in a specific car class? Is it then possible to distinguish the classes by these characteristics?* The aim of this work is to answer these questions, i.e. to classify different car classes according to their combustion engine sounds.
In this chapter, keywords, which are necessary for the understanding of this paper, will be explained. In the first section, principles of the engine sound generation will be described, which are summarised by the combustion noise and the mechanical noise. In the second section, the car classification scheme exploited in this work will be introduced. Lastly, the psycho-acoustic parameters as well as other relevant parameters used in this work will be explained.

1.1 Combustion engine sound

The engine sound, in addition to the sounds from the gearbox and the intake- and exhaust-system, is one of key elements to form the entire sound characteristics of a car. This sound is also needed to fulfil acoustical requirements in a passenger car, for instance, to seek a comfortable sound in a luxury car or to generate a unique sound corresponding to a sports

car brand. The optimisation of the combustion engine sound has been an important issue in the automotive industry since the car operated by the internal combustion engine was developed. This task, however, becomes increasingly challenged by the trend of the light and complex aggregate [Zel12].
The combustion engine sound consists of two parts: the direct combustion noise and the mechanical noise. The direct combustion noise is caused by the pressure pulsation in the combustion chamber as a result of the combustion of the air-fuel-mixture. The generating process of the direct combustion noise is described in Figure 1, where the spectrum from the pressure pulsation is summed up with the spectrum of a transmission path to a receiver.

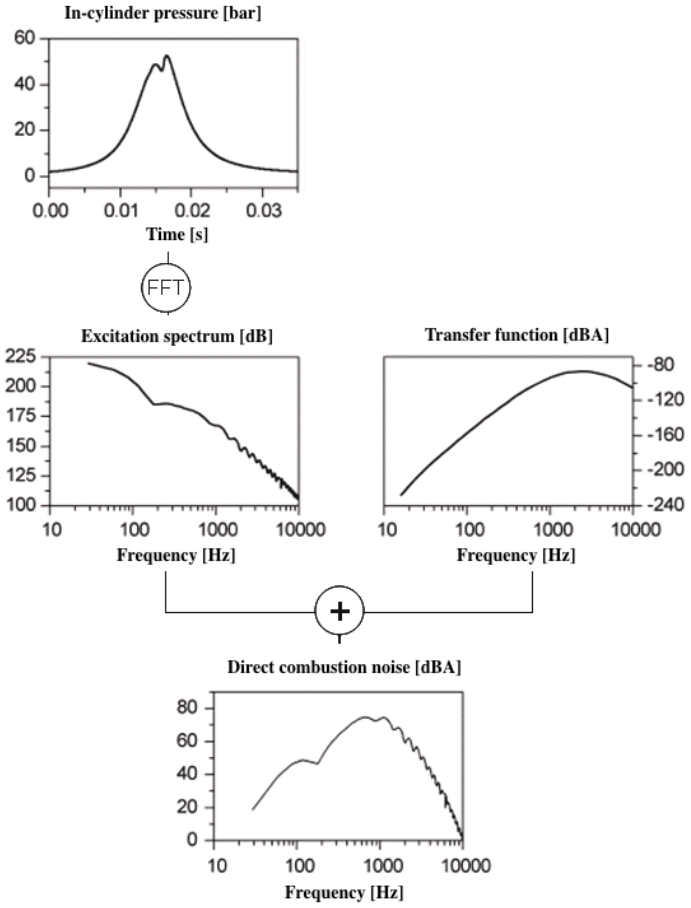


Figure 1 – *Cause-and-effect relationship of the direct combustion noise [Bar01]*

On the other hand, the mechanical noise which essentially contributes to the sound distinction of a car, results from the crankshaft motion. The dynamical characteristics of the crankshaft is influenced by various physical elements, such as cylinder order, balance weights, mount concept of the crankshaft, etc. The periodic excitation moment of the crankshaft can be described by Fourier-series:

$$M(t) = M_0 + \sum_{\nu} \hat{M}_\nu \sin(\nu \cdot \omega \cdot t + \delta_\mu), \tag{1}$$

where ω is the constant fundamental angular frequency and ν is the order of excitation. Although undesirable harmonics of some specific orders can be compensated by the cylinder arrangement and the balancing weight in the crankshaft, it is only partially successful, i.e. other undesirable harmonics remain uncompensated. As an example, an in-line 4 cylinders engine is shown in figure 2. The inertia force F_2 corresponding to the moment of second order has periodic of 180 [deg], which is not compensated in the system, while other moments are eliminated successfully.

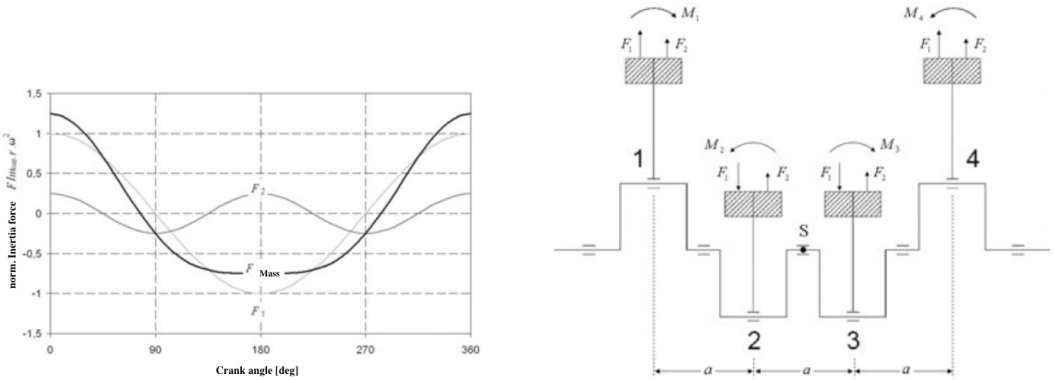


Figure 2 – *(left) inertia force caused by moments of 1. and 2. order. (right) excitation moments in in-line 4 cylinders engine [Zel12]*

Consequently, these two mechanisms of noise generation contribute to producing the characteristics of the engine sound.

1.2 Car classification

There are enormous different schemes for car classification worldwide that are used for various purposes, including regulation, description, categorisation, etc [CCW17]. There are also a lot of classification methods, such as, by means of the body style, number of doors, number of seats, weight, etc. In this work, a scheme called *Euro Car Segment* defined by *European Commission* is used, which is based on a number of objective criteria like engine size and length of cars. However, this is not formal and somehow vague [Car99]. The classes called segments are categorised into 9 segments as follows:

— A-segment : mini cars
— B-segment : small cars
— C-segment : medium cars
— D-segment : large cars
— E-segment : executive cars

— F-segment : luxury cars
— S-segment : sport cars
— M-segment : multi-purpose cars
— J-segment : sport utility cars

1.3 Psycho-acoustic parameters

It is usual to evaluate the engine sound by psycho-acoustic parameters, since the subjective impression in various recognition-dimensions, such as loudness, sharpness, roughness, etc., is the key factor to understand how the sound is recognised by humans. In this section, the important psycho-acoustic parameters are introduced, for instance the *Zwicker-parameter* and other relevant parameters which are often used in practice. These parameters are also used in this work. Detailed description of the *Zwicker-parameter* is represented in [FZ07].

— The most important part of the psycho-acoustical evaluation is to estimate how loud a sound is. The representative parameter is *Loudness [sone]* defined in *ISO532B*. There is also various *weighted sound pressure level [dB( )]* group defined in *ICE 61672*. The weighted sound pressure levels are obtained by applying different weight curves in the frequency domain. Moreover, other specified parameters used in practice, such as *Annoyance, Sportiness, Powerfulness, etc*, are derived by combining basic parameters for the loudness. On the other hand, a parameter *Articulation Index* is adopted in this work in order to evaluate quietness as opposed to the loudness.
— *Sharpness [acum]* is one of the important parameters in determining the sensory pleasantness. It indicates a measure of the high frequency content of a signal. The sensory pleasantness decreases with increasing the sharpness.
— A feature distinguishing the tonal quality from the noise component of sounds is *Tonality*, i.e. it indicates how many tonal components are contained in a signal. The more tonal components the sound has, the more pleasant it sounds.
— A parameter *Roughness [asper]* is based on the amplitude-modulation. At very low modulation frequency up to $15Hz$, the loudness changes up and down slowly and this sensation is represented as parameter *Fluctuation Strength*. On the other hand, at the modulation frequency range from 15 to $300Hz$, the sensation roughness is dominant. In other words, the roughness is created by the relatively quick changes of the loudness. The parameter *Impulsiveness*, similar with the roughness, also indicates the sensation of quick level changes, however, it contains aperiodic as well as periodic modulation excitation.

2 Data Analysis

In this chapter, the procedure for data analysis is described. At the first section, given parameters and the number of samples in each class are introduced. After that, feature vectors are constructed by averaging values of each parameter within a certain engine speed range and they are normalised as a next step. Consequently, some redundant parameters are removed from the set of feature and the rest are grouped by means of *Principal Component Analysis (PCA)*. Lastly, the most discriminant subset of features is selected and then a classifier is built by this subset, with applying *Linear Discriminant Analysis (LDA)*.

2.1 Data structure

The set of samples used in this work is given by *AVL List GmbH*. It consists of engine sounds of 261 different passenger cars. In each sample, the engine speed [rpm] increases gradually as function of time, which means the acceleration situation, and 29 psychoacoustic parameters are calculated corresponding to the engine speed. Table 1 shows the name of the given parameters. The given data includes not only the basic parameters such as loudness, sharpness, tonality and roughness (see section 1.3) but also some combined parameters such as annoyance, sportiness, luxury, etc. Although information about measurement process or definition of these practical parameters are not provided, it is possible to deduce the similarity with the known parameters by PCA (see section 2.3.2).

Linear Level	Roughness (AVL)	Sportiness (Japan)
A-weight. Level	Articulation Index	DOC Power
B-weight. Level	Ext. Articulation Index	RE0 Power
C-weight. Level	Low frequency content	Powerfulness
D-weight. Level	Standard Deviation	Luxury
AD-weight. Level	Impulsiveness (Kurtosis)	Evenness
Loudness (ISO532B)	Annoyance (Europe)	unweighted Sharpness
Sharpness (Aures)	Engine Speed Change	Annoyance (Japan)
Sharpness (Zwicker)	DOC Sport	CKI
Tonality	Sportiness (Europe)	

Table 1 – 29 given parameters

The number of samples in each class is shown in table 2. The scheme of car classification used for this work is *Euro Car Segment* defined by European Commission (section 1.2). Since the data set was not collected for the purpose of this work, there is a lack of samples in some specific classes when compared to other classes, such as A-, F- and M-segment, which makes it difficult to generalise classifiers for these classes (see section 3.5).

Class	#Samples
A-segment	9
B-segment	44
C-segment	55
D-segment	59
E-segment	23
F-segment	12
S-segment	20
M-segment	11
J-segment	28

Table 2 – The number of samples in each class

2.2 Data pre-processing

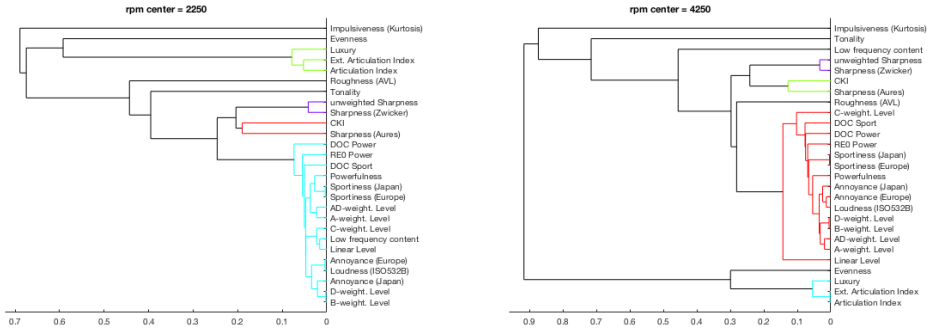
Before implementing classifiers, it is necessary to refine the given data set in so-called pre-processing stage. This stage consists of two parts. At first, the engine speed is segmented into five ranges and each parameter, called feature in the machine-learning sense, is averaged over each range, which eventually yields a 29-dimensional feature vector in each range. In other words, each sample keeps five feature vectors corresponding to the engine speed range. The engine speed range is divided as follows:

$$@rpm_{center} = [2250, 2750, 3250, 3750, 4250], \pm 250, \tag{2}$$

where $@rpm_{center}$ indicates the centre of each rpm-range and ± 250 is the width of each range. The range division was specified so that it includes all samples without discarding any samples, since some samples have no measurement out of the proposed rpm-range.
The second step is to normalise the feature vectors. It is required as the values of each feature are distributed with a different statistics, e.g. different mean and variance values, thus each distribution has to be in one normalised range. The following normalisation

$$z_{ij} = \frac{x_{ij} - \mu_j}{|\sigma|} \tag{3}$$

yields feature values z_{ij} with $\mu_j = 0$ and $|\sigma| = 1$, where i is the sample index, j is the feature index, μ_j is the mean, and $|\sigma|$ is the average absolute deviation. $|\sigma|$ was applied to suppress the influence of outliers under the assumption that the data set has a certain amount of outliers.
After the pre-processing, the dimension of the data is $261 \times 29 \times 5$.



(a) $rpm_{center} = 2250$ (b) $rpm_{center} = 4250$

Figure 3 – *Dendrogram plot of 27 features; coloured groups represent groups under the threshold 0.2*

2.3 Parameter grouping

2.3.1 Removal of unnecessary features

The dimension of the feature vectors should be reduced before the actual classification by removing unnecessary features. First, two features that are not acoustic parameters were removed from the feature set: *Standard Deviation* and *Engine Speed Change*.
Thereafter, the correlation between the parameters represented in the form of the dendrogram analysed (see figure 3). Figure 3 shows only the dendrograms of 2250 and 4250 rpm-range as the behaviour of the other ranges is highly similar with that of these two ranges. According to the correlation analysis, there are several highly correlated parameter pairs. One of the two features of each pair, that is, a total of 7 features was eliminated from the feature set:

— *Linear Level*
— *B-weight. Level*
— *AD-weight. Level*
— *Ext. Articulation Index*
— *Annoyance (Europe)*
— *Sportiness (Japan)*
— *Unweighted Sharpness*.

After removing the redundant features, the dimension of the data was reduced intermediately to $261 \times 20 \times 5$.

2.3.2 Principal Component Analysis (PCA)

After the removal of the redundant features, parameters which contribute to the data distribution should be examined. The information of the data distribution was investigated by *Principal Component Analysis (PCA)*. The PCA is a linear transformation which is described as follows:

$$\mathbf{Y} = \mathbf{AX}, \tag{4}$$

where $\mathbf{X}$ is the matrix of input, $\mathbf{Y}$ is the output data, and $\mathbf{A}$ is the transformation matrix. $\mathbf{C}$ is the covariance matrix which is calculated from the data matrix $\mathbf{X}$ as follows:

$$\mathbf{C} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x_n} - \mathbf{m_x})(\mathbf{x_n} - \mathbf{m_x})^T, \tag{5}$$

where $\mathbf{x_n}$ is the feature vector of *nth* sample, $\mathbf{m_x}$ is the mean vector, and N is the number of samples. The eigenvalues corresponding to the eigenvectors of $\mathbf{C}$ represent the transformed variances which are equal to the information of the data distribution. The eigenvalues are sorted in ascending order and their cumulative summation is shown in Figure 4.

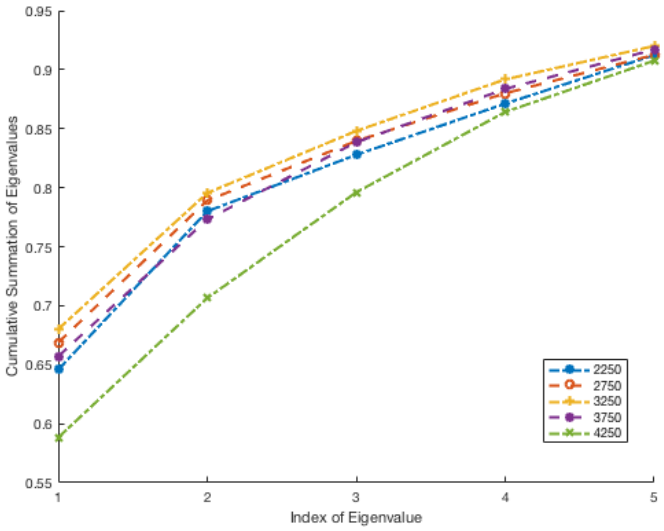


Figure 4 – *Cumulative summation of eigenvalues of all rpm-ranges*

As seen in Figure 4, the cumulative summation of the eigenvalues already reaches 0.9 at the index of 5 in all engine speed ranges. In other words, the transformation from *20D* to *5D* holds almost 90% of the distribution information. Assuming that the amount of this information is sufficient to express the degree of the dispersion of the data, only up to the fifth principal component is considered. Interestingly, the rpm-ranges of 2250, 2750, 3250 and 3750 have a similar behaviour, where the first eigenvalue has about 65% of the information amount, while the behaviour of the range 4250 is different from the others, where the first eigenvalue contains less than 60%. Appendix A shows 2D-biplots of both

data distribution and contributing parameters represented by points and arrows, respectively. It contains only of 2250, 3250, and 4250 rpm-range. Due to the similarities of the plots of 2750, 3250 and 3750 rpm-range, only the plot of 3250 rpm-range is represented. Since the fifth component of all rpm-ranges does not represent a specific parameter contribution, it was eventually excluded. According to the biplots and the dendrogram plots (see figure 3), the features were grouped by the PCA and the correlation analysis. Due to the different feature contributions among the rpm-ranges: [2250], [2750, 3250, 3750], [4250], parameter grouping was carried out in three different ranges. Table 3 shows the parameter contribution to the each component. In 2250 rpm-range, each group named as a representative parameter contains similar parameters as follows:

— *Luxury*: Articulation Index, Luxury
— *Loudness*: A-weight. Level, C-weight. Level, D-weight. Level, Loudness (ISO532B), Low frequency content, DOC Sport, Sportiness (Europe), DOC Power, RE0 Power, Powerfulness, Annoyance (Japan)
— *Tonality*: Tonality
— *Sharpness*: Sharpness (Aures), Sharpness (Zwicker)
— *Evenness*: Evenness
— *Impulsiveness*: Impulsiveness,

In 3250 and 4250 rpm-range, Low frequency content is separated from the *Loudness* as an independent group and in 4250 rpm range *Evenness* is merged to the group *Luxury*. It implies that some parameters behave differently depending on the rpm-range. There are 6 or 7 groups with regard to the rpm-range. Additionally, two extra parameters were removed from the feature set, which do not contribute to the PCA: *Roughness (AVL)* and *CKI*. Ultimately, the dimension of the data was reduced to $261 \times 18 \times 5$.

	negative	positive
1.PCA	Luxury	Loudness
2.PCA	Tonality	Sharpness
3.PCA	Tonality	Evenness
4.PCA	Evenness	Impulsiveness

	negative	positive
1.PCA	Luxury	Loudness
2.PCA	Tonality	Sharpness
3.PCA	LFC	Impulsiveness
4.PCA	Evenness	Tonality

	negative	positive
1.PCA	Luxury	Loudness
2.PCA	Sharpness	LFC
3.PCA	Tonality	Sharpness
4.PCA	LFC	Impulsiveness

Table 3 – *Parameter contribution; top-left corner: rpm-range = 2250, top-right corner: 3250, bottom: 4250*

2.4 Feature selection

It is necessary to select a subset of features rather than using the entire set, since the full set can cause a potential risk of overfitting, that is, it can build an unstable classifier. Furthermore, a classifier might be physically contradictory e.g. multiple physically similar features could contribute in the opposite direction of a projection vector in LDA (more information in section 2.5.1). In the subset of features, the parameters are sorted in ascending order according to their discriminant performance. Only one feature per group is selected to prevent physically contradictory classifier formation, i.e. once a parameter is selected in one group, the remaining parameters of that group are not selected in the next selection. The used selection algorithm is called *Sequential Forward Selection*. The algorithm starts from an empty feature subset and adds a single feature which yields the maximum value of an objective function J in order:

$$\begin{aligned} x^+ &= \underset{x \in X - Y_k}{argmax}[J(Y_k + x)] \\ Y_{k+1} &= Y_k + x^+, \end{aligned} \tag{6}$$

where Y_k is the subset of features and x^+ is the selected feature. The applied objective function is the discriminant potential (DP) which is also used for LDA, later. The DP is defined as follows:

$$DP = \frac{Tr\{\mathbf{S_b}\}}{Tr\{\mathbf{S_w}\}}, \tag{7}$$

where $\mathbf{S_b}$ is the between-class scatter matrix, $\mathbf{S_w}$ the within-class scatter matrix, and $Tr\{\cdot\}$ is the trace operator. In Eq. (8), these scatter matrices are derived:

$$\begin{aligned} \mathbf{S_b} &= \sum_{k=1}^{K} \frac{L_k}{L}(\mu_k - \mu)(\mu_k - \mu)^T \\ \mathbf{S_w} &= \sum_{k=1}^{K} \frac{L_k}{L}\mathbf{C_k}, \end{aligned} \tag{8}$$

where K is the number of classes, L_k is the number of samples of class k, L is the number of all samples, μ_k is the mean vector of class k, μ is the global mean vector, and $\mathbf{C_k}$ is the covariance matrix of class k (cf. definition in Eq. (5)). To simply describe the DP, it means maximising the distance between classes while minimising the variance within each class. Consequently, the subset includes 6 or 7 features with regard to the rpm-range and lower-priority features such as the 6th and 7th feature, can be removed from the subset if they do not support the classification.

2.5 Classification

Eventually, the car classes are distinguished by the selected feature subset. Linear Discriminant Analysis was applied to make a classifier. The performance of the classifier is represented by the confusion matrix. Also, the robustness is evaluated by N-folds cross validation.

2.5.1 Linear Discriminant Analysis (LDA)

Likewise for PCA, *Linear Discriminant Analysis (LDA)* is also a method used for feature extraction. Unlike PCA, LDA additionally considers the class information to find optimal components contributing to the class separation. The method is realised by the between-class scatter matrix $\mathbf{S_b}$ and the within-class scatter matrix $\mathbf{S_w}$ defined in Eq. (8). These matrices are computed in each class and summed up at the end. Consequently,

$$\mathbf{S_w^{-1}S_b} \tag{9}$$

is diagonalised, i.e. the eigenvectors and its eigenvalues of Eq. (9) are computed. The eigenvector with the largest eigenvalue is equal to the first projection vector. In this manner, up to $K-1$ projection vectors are obtained, where K is the number of classes. Eventually, the data is transformed by the projection vectors and the transformed data is classified by applying to a linear classifier.

2.5.2 Confusion matrix

The performance of a classifier is evaluated by the distance-based *Confusion Matrix (CM)*. The distance between each sample and the mean vector of each class is calculated. Therefore, each sample belongs to a class which yields the shortest distance to the sample. In this way, every sample belongs to one of the classes. Then, the assigned class is compared with the actual class of the sample to determine whether the classification is correct. The number of correctly categorised samples is represented by the diagonal elements of CM, while the misclassified samples occupy other elements in CM corresponding to the inaccurately categorised classes. For instance, in

$$CM = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, \tag{10}$$

a_{11} occupies the rate of accurate classified samples to class 1 while a_{12} the rate of the misclassified samples from class 1 to class 2.

2.5.3 Cross validation

In order to evaluate the robustness of the classifier, *4-folds Cross Validation method* was used in this work. The sample set in each class is divided into 4 segments, three of them are used for training while the other for testing. Likewise, the training and the testing processes are repeated four times with different training and test set combinations and confusion matrices are calculated by the test set in each run. Furthermore, a projection vector is also calculated in each run and 4 projection vectors are then compared to each other to evaluate how consistent they are. The consistency of the projection vectors is evaluated quantitatively by the projection error in (11):

$$e_{i,j} = 1 - \left| \frac{\mathbf{w_i}^T \mathbf{w_j}}{|\mathbf{w_i}||\mathbf{w_j}|} \right|, \tag{11}$$

where $\mathbf{w}$ is the projection vector and $\mathbf{i}, \mathbf{j}$ is the index of run.

3 Results of Classification

In this chapter, the classification results are shown including target classes, a feature subset, an estimate of misclassification rate, projection error averaged by run, a confusion matrix, and a plot of a classifier. Only one classifier is adopted in the rpm-range with the highest discrimination performance among the five ranges. The type of used classifier is *"Linear"* to exclude any complexity of the classifier itself and to focus on only the performance of the selected feature subset. Before implementing the actual classifier, the outliers in all (upper-)classes are removed to prevent the distortion of the classifier by the outliers which lie outside of a specific range:

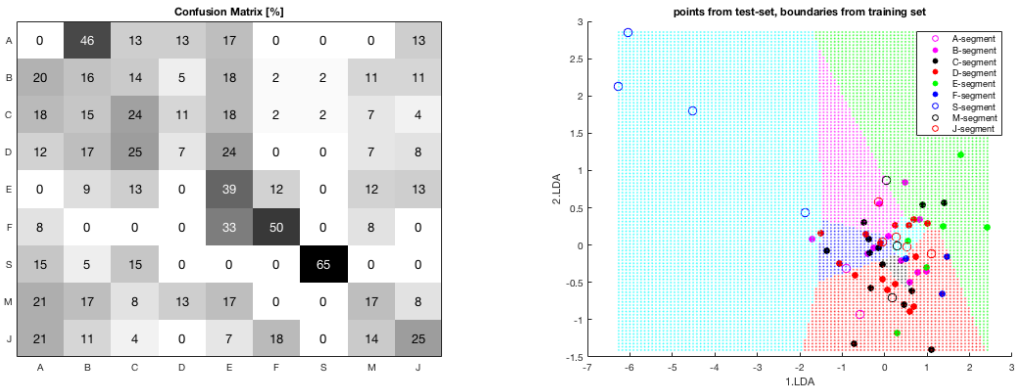
$$\{\mathbf{x_c} > 3\sigma_c \| \mathbf{x_c} < 3\sigma_c\}, \forall \mathbf{x_c}, \tag{12}$$

where $\mathbf{x_c}$ is a single sample in the (upper-)class $\mathbf{c}$ and σ_c is the standard deviation of the class.

3.1 Classifier for all classes

First of all, a classifier to distinguish all classes was investigated:

- target classes: {**A**} ‖ {**B**} ‖ {**C**} ‖ {**D**} ‖ {**E**} ‖ {**F**} ‖ {**S**} ‖ {**M**} ‖ {**J**}
- engine speed range = 2250
- feature subset = *{Loudness (ISO532B), Articulation Index, Sharpness (Aures), Tonality, Impulsiveness (Kurtosis), Evenness }*
- estimate of misclassification rate $\approx 66\%$
- projection error $\approx 0\%$
- classifier plot and confusion matrix in figure 5.



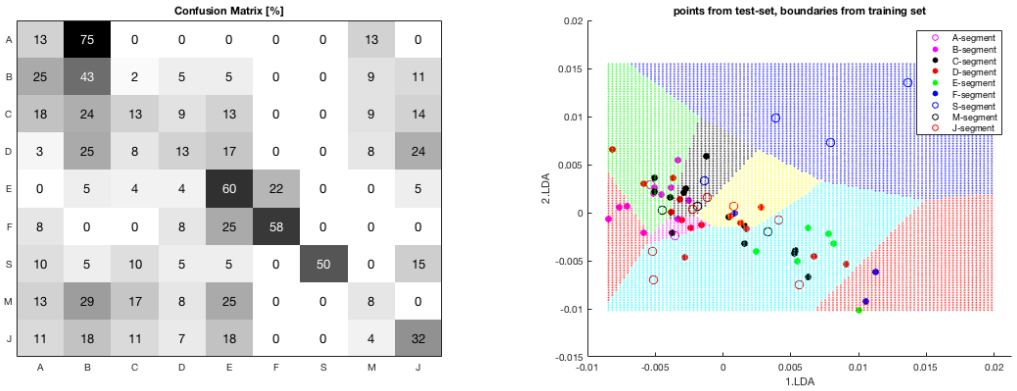
(a) Confusion matrix

(b) Classifier plot in 2D

Figure 5 – *Classifier for all classes in rpm-range = 2250*

The subset of features was selected the same in all rpm-ranges, i.e. the subset is independent of the rpm-range. The classifier, however, did not represent a sufficient classification performance except for the *F- and S-segment*. In order to seek a better performance, the full set of 18 features was used instead of the subset:

— target classes: {**A**} ‖ {**B**} ‖ {**C**} ‖ {**D**} ‖ {**E**} ‖ {**F**} ‖ {**S**} ‖ {**M**} ‖ {**J**}
— engine speed range = 2250
— full set of 18 features (see Table 1)
— estimate of misclassification rate $\approx 56\%$
— projection error $\approx 0\%$
— classifier plot and confusion matrix in figure 6,



(a) Confusion matrix (b) Classifier plot in 2D

Figure 6 – *Classifier for all classes with full set of 18 features in rpm-range = 2250*

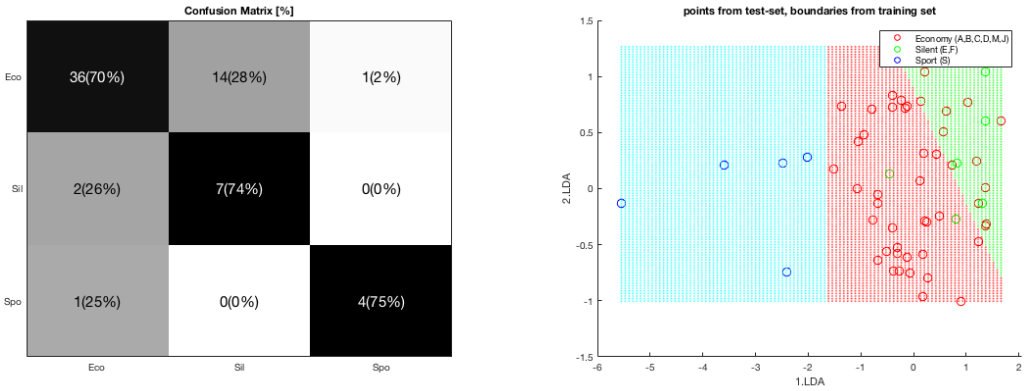
Although the performance was slightly improved in this case, i.e. the misclassification rate when using the entire set is slightly lower than when using the previous subset, 66% to 56%, the size of the entire set is three times larger than the size of the previous subset and the performance is still poor. As a result, it is nearly impossible to distinguish all classes by a single classifier, using the given set of features.

3.2 Economy- vs. Silent- vs. Sport-class

Since it is difficult to discriminate all classes at once, the existing classes are grouped into upper-classes. The upper-classes are conventionally divided according to [YVFR14]: *Economy-*, *Silent-*, and *Sport-class*. The *Silent-class* includes the *E- and F-segments* and the *Sport-class* consists solely of the *S-segment* while the *Economy-class* includes all the rest segments. The classifier is represented as follows:

— target classes: {**A, B, C, D, M, J**} ‖ {**E, F**} ‖ {**S**}
— engine speed range = 2250
— feature subset = *{Loudness (ISO532B), Articulation Index, Sharpness (Aures), Tonality}*

— estimate of misclassification rate: $\approx 24\%$
— projection error: $\approx 0\%$
— classifier plot and confusion matrix in figure 7.



(a) Confusion matrix (b) Classifier plot in 2D

Figure 7 – *Classifier for economy | silent | sports in rpm-range = 2250*

The classifier is highly robust independent of rpm-ranges. It also yields a satisfactory performance, even though there are some misclassifications between the *Economy-class* and the *Silent-class* while there is no misclassification between the *Economy-class* and *Sports-class* and between the *Silent-class* and *Sports-class*. The selected feature subset includes only those features that contribute to the first and second PCA, which means that these features is sufficient for classification.

3.3 Classifier for Sport Utility Vehicle class

Figure 8 shows a boxplot of 18 features in *J-segment* called sport utility vehicle (SUV) class in 2250 rpm-range. It indicates that the *J-segment* has a distinctive feature *Sharpness*. Therefore, it is necessary to examine whether the feature supports to distinguish the *J-segment* from other classes.

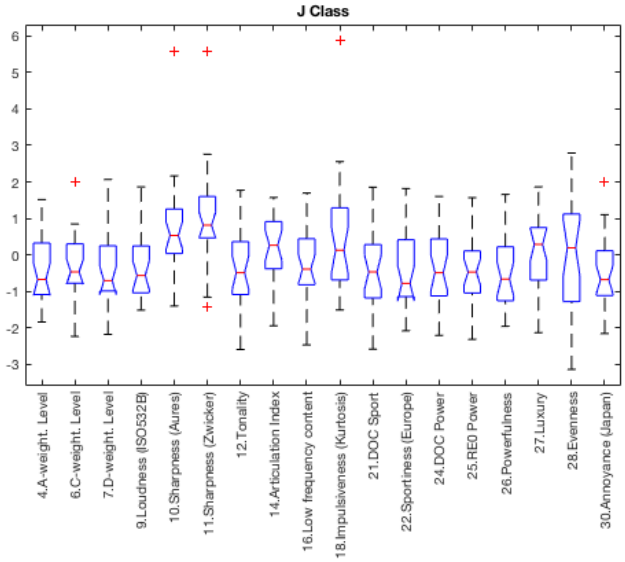
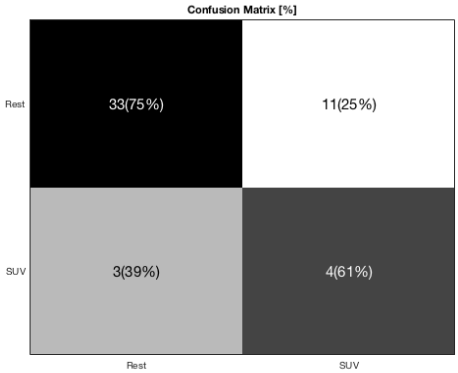


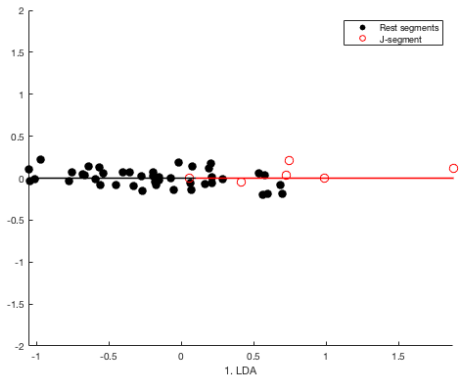
Figure 8 – *Boxplot of 18 features in J-segment in 2250 rpm-range*

Considering only the segments in the *Economy-class*, a classifier for the *J-segment* was implemented by using only *Sharpness (Zwicker)* in 2250 rpm-range:

- target classes: {**A, B, C, D, E, M**} ‖ {**J**}
- engine speed range = 2250
- feature subset = *{Sharpness (Zwicker)}*
- estimate of misclassification rate $\approx 24\%$
- projection error: no exists
- classifier plot and confusion matrix in figure 9.



(a) Confusion matrix



(b) Classifier plot in 1D

Figure 9 – *Classifier for rest | J in rpm-range = 2250, vertical axis of plot is meaningless i.e. the samples are randomly distributed in order to be presented clearly.*

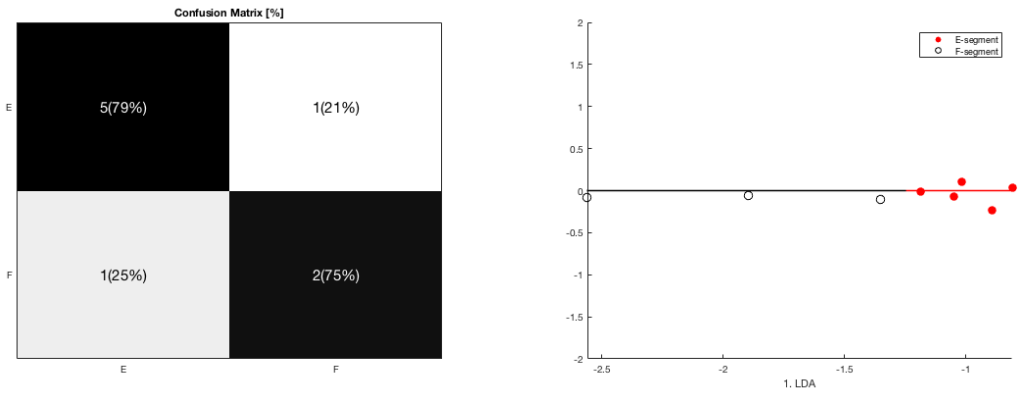
It is clear that the binary classifier between the *J-segment* and the rest yields a single projection vector, therefore, the classifier is one dimensional. In fact, the feature *Sharpness*

(Zwicker) successfully discriminated the *J-segment* from the rest. Although at a different rpm-ranges the feature showed slightly lower DP, the performance itself was not poor compared to the 2250 range. One interesting observation is that the performance of the classifier has degenerated when other features selected by the selection algorithm belong to the subset, which means that only *Sharpness (Zwicker)* is helpful for classification.

3.4 Executive- vs. Luxury-class

In the previous section 3.2, the *E-segment (Executive class)* and *F-segment (Luxury class)* were grouped into the upper-class *Silent-class*. To prove that the two classes are distinguishable, the following classifier has been constructed:

- target classes: {**E**} || {**F**}
- engine speed range = 2250
- feature subset = *{DOC Power, Articulation Index}*
- estimate of misclassification rate $\approx 20\%$
- projection error $\approx 0\%$
- classifier plot and confusion matrix in figure 10.



(a) Confusion matrix (b) Classifier plot in 1D

Figure 10 – *Classifier for E | F in rpm-range = 2250, vertical axis of plot is meaningless i.e. the samples are randomly distributed in order to be presented clearly.*

As a result, both classes were separated by a subset of features that contains only two parameters of the first PCA, even though they have similar characteristics of the feature distribution (see Figure 11).

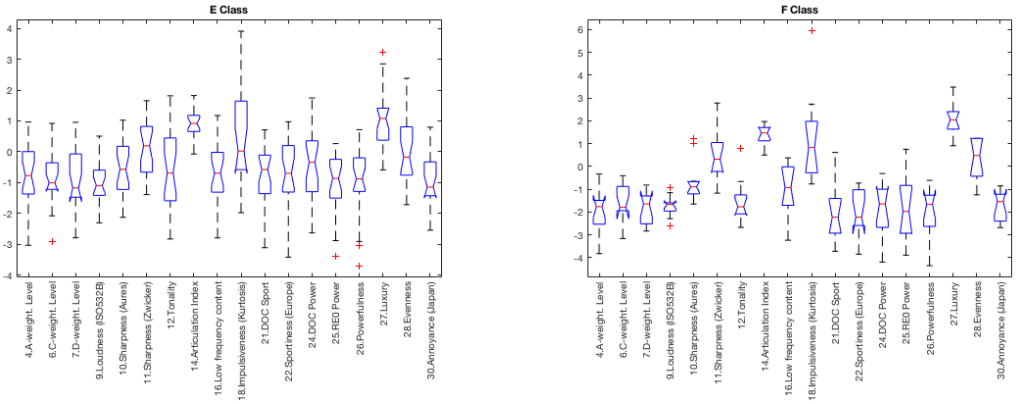
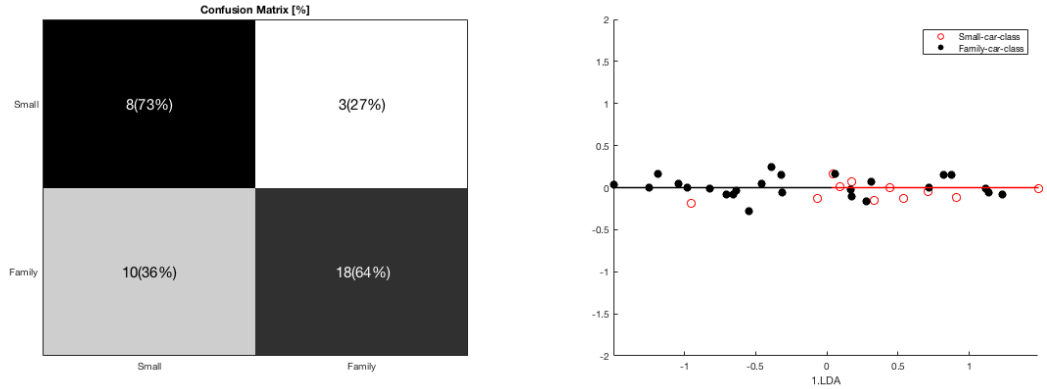


Figure 11 – *Boxplot of 18 features in E- (left) and F-segment (right) in 2250 rpm-range*

3.5 Small-car- vs. Family-car-class

Due to insufficient numbers of samples in *A*- and *M-segment*, it is difficult to construct a classifier for these classes. For example, it is difficult to produce a consistent projection vector with a small number of samples, and it is easy to show a very large misclassification rate with only a small number of false classified samples during the test. For these reasons it is difficult to generalise the classifier for these classes. Hence, it was abandoned to build classifiers for *A*- and *M-segment*. Instead, a classifier has been built to discriminate between *B-segment*, called *Small-car-class*, and an upper-class called *Family-class* which contains both *C*- and *D-segments*:

— target classes: {**B**} || {**C, D**}
— engine speed range = 4250
— feature subset = *{Sharpness (Aures), Luxury, Low frequency content, Loudness(ISO532B)}*
— estimate of misclassification rate $\approx 25\%$
— projection error $\approx 6\%$
— classifier plot and confusion matrix in figure 12.



(a) Confusion matrix

(b) Classifier plot in 1D

Figure 12 – *Classifier for small-car-class | family-car-class in rpm-range = 4250, vertical axis of plot is meaningless i.e. the samples are randomly distributed in order to be presented clearly.*

Although it yields a satisfactory separation performance, there are a few misclassifications in this case. The classifier is robust independent of rpm-ranges, i.e. the selected subset as well as the performance are quite similar in all ranges.

4 Conclusion

This work started with a simple question: Could different car classes be classified by their engine sounds? Although it failed to realise an ideal classifier due to the limitations of the given data information, several interesting results were observed.

First, it was not possible to distinguish all classes at once. There are two reasons for this; one is that the given set of parameters may not have enough information to distinguish all classes, the other is that some of the samples and classes may have already been overlaid and so the classification itself was impossible.

The second interesting observation is that the classifications performed in this work were usually independent of the engine speed range. In fact, according to the PCA, the parameters are grouped slightly differently over the speed range, nevertheless, most classifiers had a consistent feature subset regardless of the rpm range. Moreover, the performance difference across all ranges was not significant.

The third interesting point is that even among classes that did not expect classification to be successful, some classifiers performed very well, for example, the *J-segment* actually contains various types of vehicles, e.g. compact, compact-luxury and luxury SUVs, which correspond to *C-*, *D-*, and *E- or F-segment*, respectively, the class was separated even by only a single feature *Sharpness*.

Finally, it is observed that only the parameters corresponding to the first and secont PCA were only selected as a subset of features. In other words, the parameters corresponding to the third PCA: *Evenness* and *Impulsiveness (Kurtosis)*, did not affect classification. This means that about 80% of the distribution information occupying the first and second PCAs has the most impact on classification.

In conclusion, the classification of car classes by their engine sounds is somewhat possible by replacing the existing classes with the upper-classes, by using the parameters corresponding to the first and second PCA, regardless of the divided speed range.

5 Future Works

The samples are not perfectly balanced because the given data is originally measured for other research purposes. For this reason, the number of samples in some classes is absolutely insufficient. To construct a classifier that can be applied to all classes, each class is required to have the balanced number of samples. Thus, if each class has a sufficient amount of samples, it would be possible to construct a reliable classifier for all classes.

In this work, there were only 6 or 7 parameter groups, of which only four were used for classification. Given the parameters, only partial classifications were successful, and no classifier could be made for all classes at once. This results from the limitations of the given set of parameters. Therefore, as part of future work, new features, such as a pitch, musical interval, musical chord as well as energetic distribution of deterministic and stochastic components, formants position/shift, etc., need to be added to the parameter set.

Another thing to be done in future work is to validate the classifiers constructed in this work through a psycho-acoustic test. This will verify that these classifiers made from data analysis are actually useful.

List of Figures

List of Tables

References

[Bar01] B. Barba, "Erarbeitung von verbrennungskennwerten aus indizierdaten zur verbesserten prognose und rechnerischen simulation des verbrennungsablaufes bei pkw-de-dieselmotoren mit common-rail-einspritzung," Ph.D. dissertation, ETH Zürich, 2001.

[Car99] "Case no comp/m.1406 - hyundai / kia, regulation (eec) no 4064/89 merger procedure," *Commission of the european communities*, March 1999.

[CCW17] July 2017. [Online]. Available: https://en.wikipedia.org/wiki/Car_classification#cite_note-16

[CP16] K. Chang and D. Park, "Technology of an emotional engine sound designing for active sound control using order balance and musical instrument sound," *SAE Technical Paper 2016-01-1782*, 2016.

[FSBH14] M. Frank, A. Sontacchi, S. Brandl, and R. Höldrich, "Sound design for electric cars: What can we learn from combustion engines?" in *Proceedings of 6th Congress of the Alps Adria Acoustics Association (6th Congress of the Alps Adria Acoustics Association)*, 2014.

[FZ07] H. Fastl and E. Zwicker, *Psychoacoustics Facts and Models*. Springer, 2007.

[GF14] K. Genuit and A. Fiebig, "Sound design of electric vehicles - challenges and risks," *Inter-noise*, 2014.

[GM14] K.-P. Glaeser and T. Marx, "Testing the sound detection of electric vehicles by blind or visually impaired persons," *Forum Acusticum*, 2014.

[GYS14] D. Gwak, K. Yoon, and Y. Seong, "Application of subharmonics for active sound design of electric vehicles," *Acoustical Society of America*, 2014.

[KSFFG14] J. Kerkmann, B. Schulte-Fortkamp, A. Fiebig, and K. Genuit, "Acceptance of synthetic driving noises in electric vehicles," *Forum Acusticum*, 2014.

[oTNHTSA13] U. S. D. of Transportation National Highway Traffic Safety Administration, "Minimum sound requirements for hybrid and electric vehicles," *NHTSA-2011-0100*, 2013. [Online]. Available: https://www.nhtsa.gov/press-releases/nhtsa-sets-quiet-car-safety-standard-protect-pedestrians

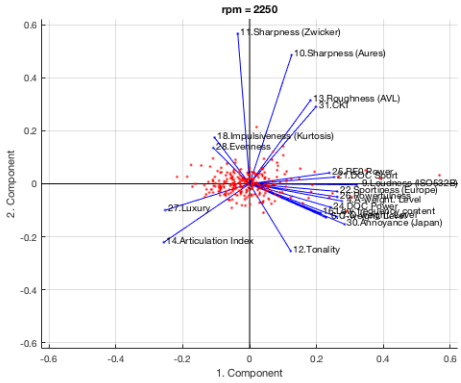
[SKvdPW14] H. Sukowski, R. Kühler, S. van de Par, and R. Weber, "Developement and application of a new adjective list for the assessment of interior sounds of electric vehicles," *Forum Acusticum*, 2014.

[YF14] K. Yamauchi and J. Feng, "Effect of frequency shifting of additional sound for electric vehicles on acceleration impression," *Forum Acusticum*, 2014.

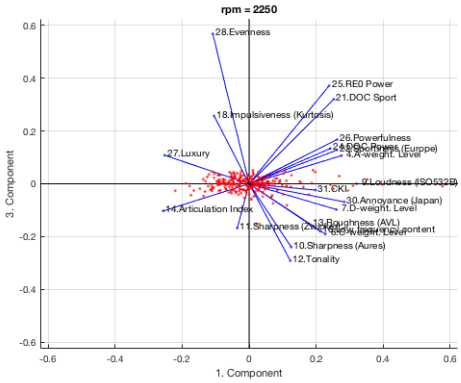
[YVFR14] J. Yoshida, F. Völk, H. Fastl, and G. Rigoll, "Influences of vehicle exterior design on loudness ratings by german versus japanese drivers." *The Acoustical Society of Japan*, vol. 35, no. 6, pp. 327–329, 2014.

[Zel12] P. Zeller, *Handbuch Fahrzeugakustik: Grundlage, Auslegung, Berechnung, Versuch*. Vieweg Teubner Verlag, 2012.

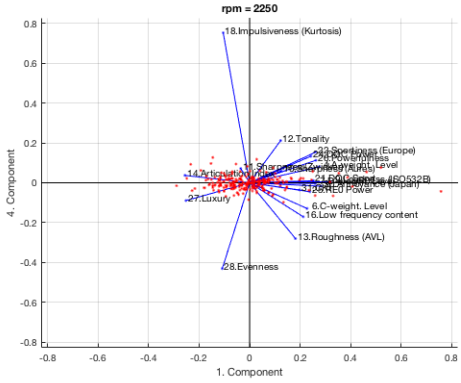
A Appendix 1: PCA-Biplot from 1. to 4. Component



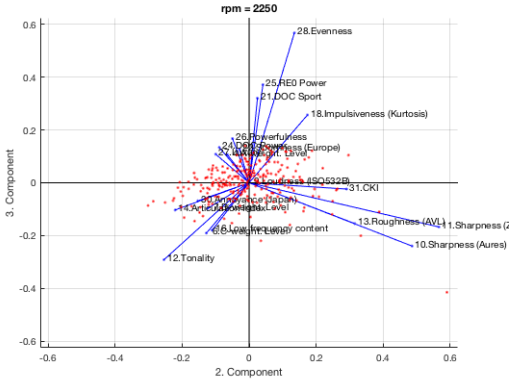
(a) 1. and 2. component



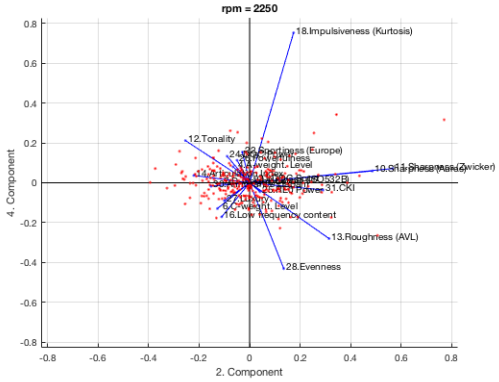
(b) 1. and 3. component



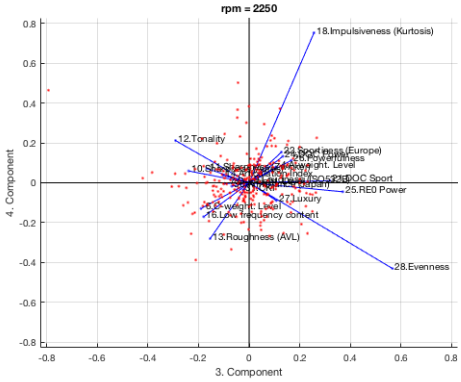
(c) 1. and 4. component



(d) 2. and 3. component

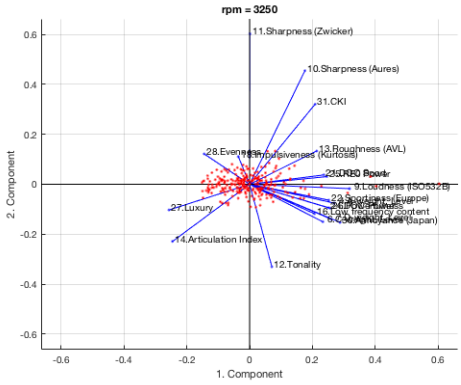


(e) 2. and 4. component

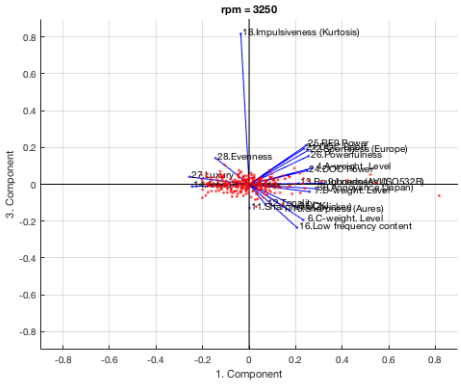


(f) 3. and 4. component

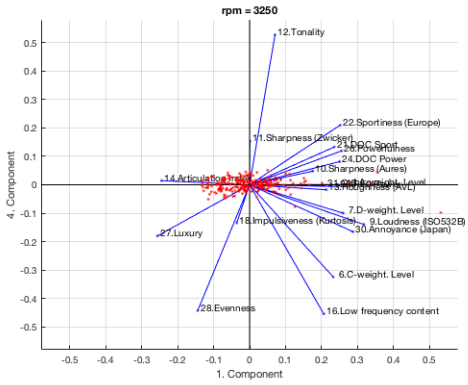
2D-biplots from 1. to 4. component of 2250 rpm-range



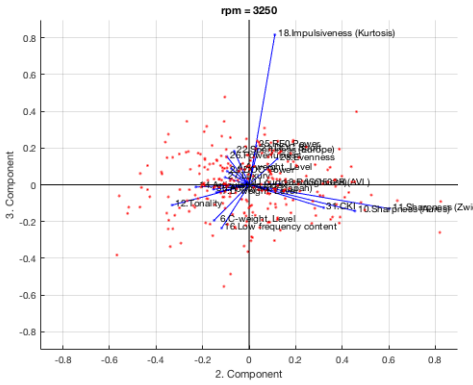
(a) 1. and 2. component



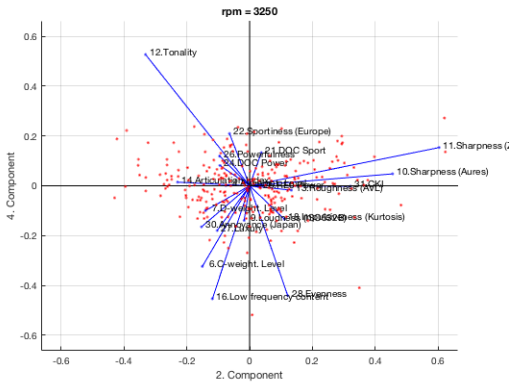
(b) 1. and 3. component



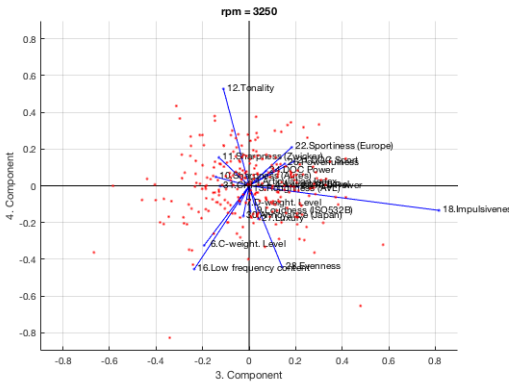
(c) 1. and 4. component



(d) 2. and 3. component



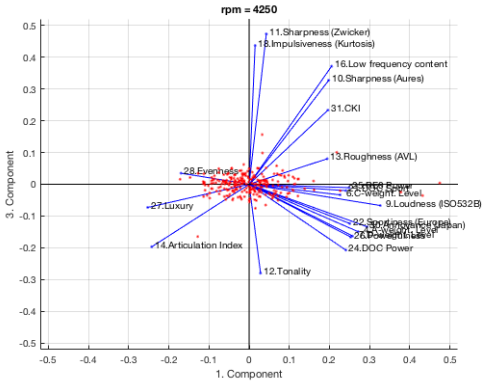
(e) 2. and 4. component



(f) 3. and 4. component

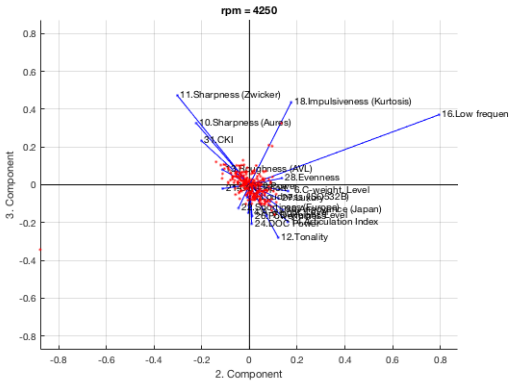
2D-biplots from 1. to 4. component of 3250 rpm-range

(a) 1. and 2. component

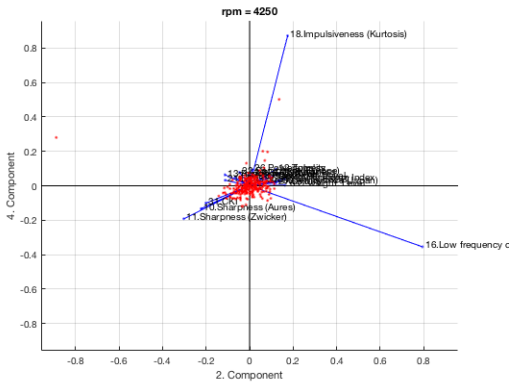


(b) 1. and 3. component

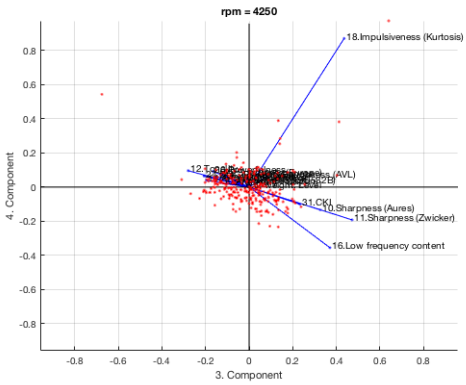
(c) 1. and 4. component



(d) 2. and 3. component



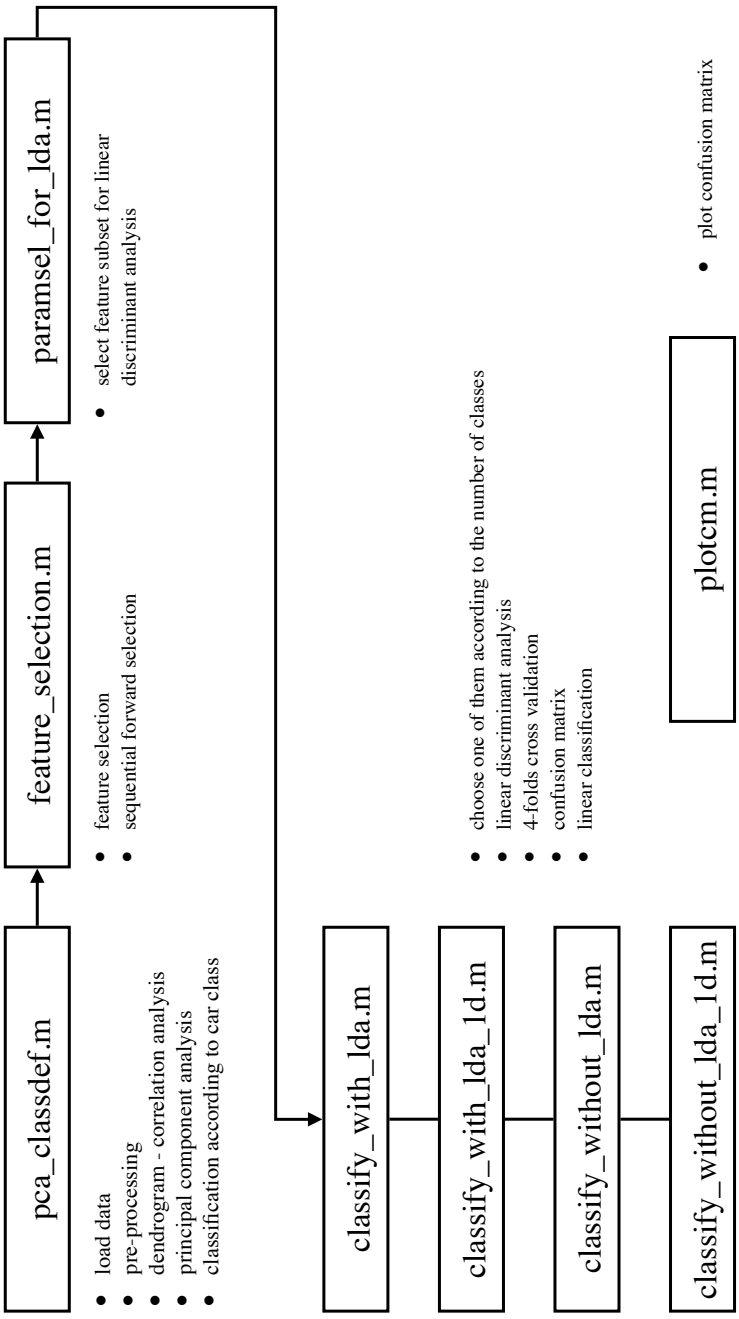
(e) 2. and 4. component



(f) 3. and 4. component

2D-biplots from 1. to 4. component of 4250 rpm-range

B Appendix 2: Overview of Matlab Structure



C Appendix 3: pca_classdef.m

```
%%% Toningenieur-Projekt, Students Project
%%% Title : Classification of Car Classes Based on Combustion Engine Sound
%%% University of Music and Performing Arts Graz, Insititue of Electronic
%%% Music and Acoustics
%%% 1173081, Jiwon Seo
%%% Supervisor : Dr. Alois Sontacchi
%%% Matlab-Implementation Part 1 : PCA & Class Definition

clc
clear all;
close all;

% load data set and lable
load('/Users/jiwonseo/Documents/MATLAB/TI_code/tables.mat')
load('/Users/jiwonseo/Documents/MATLAB/TI_code/class.mat')

% define parameter name with numbering
par_name = char(tables(1).parametername(1:31));
par_name = char(par_name);

parn = [1:length(par_name)]';
po(1:31,1) = '.';

par_n = [num2str(parn) po];
par_name = cellstr([par_n par_name]);

% initialisation
rpm_center = [2250 2750 3250 3750 4250];
len_rpm = length(rpm_center);
len_sam = length(tables);
len_par = length(par_name);

%% calculate averaged parameter-vectors according to engine speeds [rpm]
% eliminate samples if the number of samples in a certain range is smaller
% than 5.

for i = 1 : len_rpm
    for j = 1 : len_sam
        par = tables(j).parameter;
        rpm_ranged = par(par(:,2) >= rpm_center(i)-250 & par(:,2) <= rpm_center(i)+250, :);
        size_sam(j,i) = size(rpm_ranged,1);
        if size_sam(j,i) <= 5
            rpm_ranged = NaN;
        end
        par_m(j,:,i) = mean(rpm_ranged,1);
    end
```

```
47 end
48
49 %% Normalisation
50
51 for i = 1 : len_rpm
52     for j = 1 : len_par
53
54         parmean(j,i) = nanmean(par_m(:,j,i))';
55         parstd(j,i) = nanstd(par_m(:,j,i))';
56         parmad(j,i) = mad(par_m(:,j,i))';
57         par_m_norm(:,j,i) = (par_m(:,j,i) - parmean(j,i)) ./ parmad
               (j,i);
58
59     end
60 end
61
62 %% Selection of a subset of features
63 par_m_norm_29 = par_m_norm(:,[3:16 18:19 21:31],:);
64 par_name_29 = par_name([3:16 18:19 21:31]);
65 par_m_norm_20 = par_m_norm(:,[4 6 7 9 10 11 12 13 14 16 18 21 22 24
       25 26 27 28 30 31],:);
66 par_name_20 = par_name([4 6 7 9 10 11 12 13 14 16 18 21 22 24 25 26
       27 28 30 31]);
67 param_18 = [4 6 7 9 10 11 12 14 16 18 21 22 24 25 26 27 28 30];
68 par_m_norm_18 = par_m_norm(:,[4 6 7 9 10 11 12 14 16 18 21 22 24 25
       26 27 28 30],:);
69 par_name_18 = par_name([4 6 7 9 10 11 12 14 16 18 21 22 24 25 26 27
       28 30]);
70
71
72 %% Dendrogram
73
74 % for i = 1 : len_rpm
75 %
76 %     C(:,:,i) = corr(par_m_norm_29(:,:,i),'rows','complete');
77 %
78 %     figure
79 %     %imagesc(C(:,:,i)), colorbar
80 %     %title(rpm_center(i));
81 %     %hold on;
82 %     D=1-C(:,:,i);
83 %     y=squareform(D);
84 %     z=linkage(y);
85 %     [H, T] = dendrogram(z,'labels',par_name_29,'orientation','
       left','ColorThreshold',0.2);
86 %     title(['rpm center = ', num2str(rpm_center(i))]);
87 %
88 % end
89
90 %% PCA
91
92 % initialisation
93 rpm = 3; % select rpm-range for PCA e.g. 3 = [3250]
94 par_pca = par_m_norm_20;
```

```
95  par_pca_name = par_name_20;
96  len_fea_pca = size(par_pca,2);
97
98  for i = 1 : len_rpm
99      [coeff(:,:,i),score(:,:,i),latent(:,:,i)] = pca(par_pca(:,:,i))
            ;
100     sum_c(:,i) = cumsum(latent(:,:,i))/sum(latent(:,:,i));
101 end
102
103 % plot cumulative summation of eigenvalues
104 % figure
105 % hold on
106 % plot(sum_c(1:5,1),'-.*','LineWidth',2)
107 % plot(sum_c(1:5,2),'--o','LineWidth',2)
108 % plot(sum_c(1:5,3),'-.+','LineWidth',2)
109 % plot(sum_c(1:5,4),'--*','LineWidth',2)
110 % plot(sum_c(1:5,5),'-.x','LineWidth',2)
111 % xticks(1:5);
112 % xlabel('Index of Eigenvalue'); ylabel('Cumulative Summation of
      Eigenvalues');
113 % title('Cumulative Summation of Eigenvalues');
114 % legend('2250','2750','3250','3750','4250');
115
116 % 1D plot
117 % for i = 1 : 10
118 %     figure
119 %     biplot([zeros(len_fea_pca,1) coeff(:,i,rpm)],'varlabels',
      par_pca_name)
120 %     xlabel(' ');
121 %     ylabel([num2str(i) '. Component']);
122 %     %title(['rpm = ' num2str(rpm_center(i)) '+-250']);
123 % end
124
125 % 2D plot
126 % for i = 1 : 3
127 %     for j = i+1 : 4
128 %         figure
129 %         parbi(:,i,j) = biplot(coeff(:,[i j],rpm), ...
130 %         'scores',score(:,[i j],rpm),'varlabels',par_pca_name);
131 %         xlabel([num2str(i) '. Component']);
132 %         ylabel([num2str(j) '. Component']);
133 %         %legend('A','B','C','D','E','F','S','M','J');
134 %         title(['rpm = ' num2str(rpm_center(rpm))]);
135 %     end
136 % end
137
138 % 3D plot
139 % biplot(coeff(:,[1 2 3],rpm),'scores',score(:,[1 2 3],rpm),'
      varlabels',par_pca_name);
140 % xlabel('1.PCA'); ylabel('2.PCA'); zlabel('3.PCA');
141 % title(['rpm = ' num2str(rpm_center(rpm))]);
142
143 %% select a feature subset for LDA
144 EU_Com_init = ['A','B','C','D','E','F','S','M','J'];
```

```
145 par_lda = par_m_norm_18;
146 par_lda_name = par_name_18;
147 len_feature = size(par_lda,2);
148
149 %% Classification of data according to car class (9 classes)
150 for i = 1 : len_rpm
151     for j = 1 : length(EU_Com_init)
152         cc = find(class == j);
153         eval(['par_lda_' num2str(EU_Com_init(j)) '(:,:,i) = par_lda
                (cc,:,i);']);
154
155     end
156 end
157
158 %% class merge
159
160 % par_lda_A = [par_lda_A; par_lda_B; par_lda_C; par_lda_D;
      par_lda_M; par_lda_J];
161 % par_lda_E = [par_lda_E; par_lda_F];
162 % par_lda_B = [par_lda_B; par_lda_C; par_lda_D; par_lda_M;
      par_lda_J];
163 % par_lda_C = [par_lda_C; par_lda_D; par_lda_M];
164 % par_lda_A = [par_lda_A; par_lda_B; par_lda_C; par_lda_D;
      par_lda_M];
165 % par_lda_D = [par_lda_D; par_lda_E];
166 % par_lda_C = [par_lda_C; par_lda_D];
167 % par_lda_B = [par_lda_B; par_lda_J];
168 % par_lda_C = [par_lda_C; par_lda_D; par_lda_E];
169
170 % new class definition, len_class is newly defined
171 EU_Com = ['A','B','C','D','E','F','S','M','J'];
172 len_class = length(EU_Com);
173 len_coeff = len_class - 1;
174
175
176 %% calculate the number of samples for each class
177 n_all = 0;
178 for i = 1 : len_class
179     eval(['n_' num2str(EU_Com(i)) '= size(par_lda_' num2str(EU_Com(
          i)) '(:,:,1), 1);']);
180     n_all = n_all + eval(['n_' num2str(EU_Com(i))]);
181 end
182
183 % count the number of NaN-samples of each class
184 % for i = 1 : len_rpm
185 %     for j = 1 : len_class
186 %       eval(['nonnan_n_' num2str(EU_Com(j)) '(i)= sum(isnan(
      par_lda_' ...
187 %           num2str(EU_Com(j)) '(:,1,i)));']);
188 %     end
189 % end
190
191 %% calculate mean- and variance-vectors
192 m_ff_u = zeros(1,len_feature,len_rpm);
```

```
193 for i = 1 : len_rpm
194     for j = 1 : len_class
195         eval(['m_ff_' num2str(EU_Com(j)) '(:,:,i)= nanmean(par_lda_
                ' num2str(EU_Com(j)) ...
196         '(:,:,i),1);']);
197         eval(['var_ff_' num2str(EU_Com(j)) '(:,:,i)= nanvar(
                par_lda_' num2str(EU_Com(j)) ...
198         '(:,:,i),1);']);
199         eval(['m_ff_u(:,:,i) = m_ff_u(:,:,i) + m_ff_' num2str(
                EU_Com(j)) '(:,:,i);']);
200     end
201 end
202 m_ff_u = m_ff_u / len_class;
203
204 %% eliminate deviated samples
205 for i = 1 : len_rpm
206     for j = 1 : len_feature
207         for k = 1 : len_class
208             for l = 1 : eval(['n_' num2str(EU_Com(k)) ';']);
209                 if eval(['par_lda_' num2str(EU_Com(k)) '(l,j,i) >'
                        ...
210                         'm_ff_' num2str(EU_Com(k)) '(1,j,i) + 3*sqrt
                            (var_ff_' ...
211                         num2str(EU_Com(k)) '(1,j,i)) || par_lda_'
                            num2str(EU_Com(k)) ...
212                         '(l,j,i) < m_ff_' num2str(EU_Com(k)) '(1,j,i
                            ) - 3*sqrt(var_ff_' ...
213                         num2str(EU_Com(k)) '(1,j,i))']);
214                 eval(['par_lda_' num2str(EU_Com(k)) '(l,:,i) = NaN;
                        ']);
215                 end
216             end
217         end
218     end
219 end
```

D Appendix 4: feature_selection.m

```
%%% Toningenieur-Projekt, Students Project
%%% Title : Classification of Car Classes Based on Combustion Engine Sound
%%% University of Music and Performing Arts Graz, Insititue of Electronic
%%% Music and Acoustics
%%% 1173081, Jiwon Seo
%%% Supervisor : Dr. Alois Sontacchi
%%% Matlab-Implementation Part 2 : Feature Selection

% initialize parameter group
rpm = 1; % select rpm-range for FS
clear feat_sel_value

if rpm == 5
    pargrp = [2 2 2 2 4 4 3 1 5 6 2 2 2 2 2 1 1 2];
    E = 6;
elseif rpm == 1
    pargrp = [2 2 2 2 4 4 3 1 2 6 2 2 2 2 2 1 5 2];
    E = 6;
else
    pargrp = [2 2 2 2 4 4 3 1 7 6 2 2 2 2 2 1 5 2];
    E = 7;
end

%% calculate optimal feature subset
% sequential forward algorithm
for h = 1 : E

    if h == 1
        feat_sel = [];
    end
    clear d_p
    comp_par = find(~isnan(pargrp));

    for i = 1 : length(comp_par)
        S_w_fs = zeros(h); S_b_fs = zeros(h);
        for j = 1 : len_class
            % calculate S_w and S_b
            eval(['S_w' num2str(EU_Com(j)) '_fs = n_' num2str(EU_Com(j)) ...
            '/n_all * nancov(par_lda_' num2str(EU_Com(j)) ...
            '(:,[feat_sel comp_par(i)],rpm));']);
            S_w_fs = S_w_fs + eval(['S_w' num2str(EU_Com(j)) '_fs;']);

            eval(['S_b' num2str(EU_Com(j)) '_fs= n_' num2str(EU_Com(j)) '/n_all * transpose(m_ff_' ...
            num2str(EU_Com(j)) '(1,[feat_sel comp_par(i)],rpm) ' ...
            '- m_ff_u(1,[feat_sel comp_par(i)],rpm))' ...
```

```
46              '* (m_ff_' num2str(EU_Com(j)) '(1,[feat_sel comp_par(i)
                    ],rpm)' ...
47              ' - m_ff_u(1,[feat_sel comp_par(i)],rpm));']);
48              S_b_fs = S_b_fs + eval(['S_b' num2str(EU_Com(j)) '_fs;'
                    ]);
49          end
50          d_p(i) = sum(diag(S_b_fs)) / sum(diag(S_w_fs)); %
                discriminant potential
51      end
52
53      [max_param, max_param_index] = max(d_p);
54      feat_sel_value(h) = max_param;
55      feat_sel(h) = comp_par(max_param_index);
56
57      subt_pargrp = (pargrp(feat_sel(h)) == pargrp);
58      for i = 1 : length(pargrp)
59          if subt_pargrp(i)
60              pargrp(i) = NaN;
61          end
62      end
63
64  end
65
66  % parameter name and dp-value of selected subset
67  par_lda_name(feat_sel)'
68  feat_sel_value
```

E Appendix 5: paramsel_for_lda.m

```
%%% Toningenieur-Projekt, Students Project
%%% Title : Classification of Car Classes Based on Combustion
    Engine Sound
%%% University of Music and Performing Arts Graz, Insititue of
    Electronic
%%% Music and Acoustics
%%% 1173081, Jiwon Seo
%%% Supervisor : Dr. Alois Sontacchi
%%% Matlab-Implementation Part 3 : Parameter Selection for LDA

%% Selection of a subset of features
subset = [10 21 14 12];
par_m_norm_subset = par_m_norm(:,subset,:);
par_name_subset = par_name(subset);
for i = 1 : length(subset)
     subset_ind(i) = find(param_18 == subset(i));
end

%% select a feature subset for LDA
par_lda = par_m_norm_subset;
par_lda_name = par_name_subset;

len_feature = size(par_lda,2);
if len_feature < len_class - 1
    len_coeff = len_feature;
end
if len_class == 2 && len_feature == 2
    len_coeff = 2;
end

%% only selected parameters for the selected classes
for i = 1 : len_class
    eval(['par_lda_' num2str(EU_Com(i)) '= par_lda_' num2str(EU_Com
        (i)) '(:,subset_ind,:);']);
end

par_lda_merged = [];
class_merged = [];
for i = 1 : length(EU_Com)
    par_lda_merged = eval(['[par_lda_merged; par_lda_' EU_Com(i) '
        ];']);
    eval(['class_merged = [class_merged; repmat(EU_Com(i),n_'
        EU_Com(i) ',1)];']);
end
class_merged = cellstr(class_merged);
```

F Appendix 6: classify_with_lda.m

Since classify_with_lda_1d.m, classify_without_lda.m, and classify_without_lda_1d.m are slightly modified version of the routine classify_with_lda.m. Therefore, these three routines are omitted due to the simplification.

```
%%% Toningenieur-Projekt, Students Project
%%% Title : Classification of Car Classes Based on Combustion Engine Sound
%%% University of Music and Performing Arts Graz, Insititue of Electronic
%%% Music and Acoustics
%%% 1173081, Jiwon Seo
%%% Supervisor : Dr. Alois Sontacchi
%%% Matlab-Implementation Part 4 : Classification & Evaluation

clc
close all

%% divide samples in each class by 4 groups for 4-fold cross validation
for j = 1 : len_class
    n_sam = eval(['n_' num2str(EU_Com(j)) ';']);
    eval(['par_lda_cv_' num2str(EU_Com(j)) ' = par_lda_' num2str(EU_Com(j)) '(:,:,rpm);']);
    eval(['par_lda_cv_' num2str(EU_Com(j)) '= par_lda_cv_' num2str(EU_Com(j)) ...
        '(randperm(n_sam)'',:);']);

    nseg = floor(n_sam/4);
    if n_sam == 4*nseg
        bbb = [nseg nseg nseg nseg];
    elseif n_sam == 4*nseg+1
        bbb = [nseg+1 nseg nseg nseg];
    elseif n_sam == 4*nseg+2
        bbb = [nseg+1 nseg+1 nseg nseg];
    elseif n_sam == 4*nseg+3
        bbb = [nseg+1 nseg+1 nseg+1 nseg];
    end

    b1_c = [0 cumsum(bbb)];

    for i = 1 : 4
    eval(['par_lda_' num2str(EU_Com(j)) num2str(i) '= par_lda_cv_' num2str(EU_Com(j)) ...
        '(b1_c(i)+1:b1_c(i+1),:);']);
    end

end

for i = 1 : len_class
    eval(['par_lda_' num2str(EU_Com(i)) '_train1 = [par_lda_' num2str(EU_Com(i)) ...
```

```
41      '2 ; par_lda_' num2str(EU_Com(i)) '3 ; par_lda_' num2str(EU_Com
            (i)) '4];']);
42      eval(['par_lda_' num2str(EU_Com(i)) '_train2 = [par_lda_'
            num2str(EU_Com(i)) ...
43      '1 ; par_lda_' num2str(EU_Com(i)) '3 ; par_lda_' num2str(EU_Com
            (i)) '4];']);
44      eval(['par_lda_' num2str(EU_Com(i)) '_train3 = [par_lda_'
            num2str(EU_Com(i)) ...
45      '1 ; par_lda_' num2str(EU_Com(i)) '2 ; par_lda_' num2str(EU_Com
            (i)) '4];']);
46      eval(['par_lda_' num2str(EU_Com(i)) '_train4 = [par_lda_'
            num2str(EU_Com(i)) ...
47      '1 ; par_lda_' num2str(EU_Com(i)) '2 ; par_lda_' num2str(EU_Com
            (i)) '3];']);
48  end
49
50  %% LDA for 4 training/test sets
51
52  for t = 1 : 4
53
54  % Clear previous training/test set
55  for i = 1 : len_class
56      eval(['clear par_lda_train_' num2str(EU_Com(i))]);
57      eval(['clear par_lda_test_' num2str(EU_Com(i))]);
58  end
59
60  % define training/test set
61  for i = 1 : len_class
62      eval(['par_lda_train_' num2str(EU_Com(i)) '= par_lda_' num2str(
            EU_Com(i)) ...
63          '_train' num2str(t) ';']);
64      eval(['par_lda_test_' num2str(EU_Com(i)) '= par_lda_' num2str(
            EU_Com(i)) num2str(t) ';']);
65  end
66
67  % Calculate mean-vectors
68  for i = 1 : len_class
69       eval(['m_' num2str(EU_Com(i)) '= nanmean(par_lda_train_'
             num2str(EU_Com(i)) ...
70      ',1);']);
71  end
72
73  % Calculate global mean vector
74  m_u = 0;
75  for i = 1 : len_class
76      m_u = eval(['m_' num2str(EU_Com(i)) '+ m_u;']);
77  end
78  m_u = m_u / len_class;
79
80  % Calculate S_w and S_b
81  S_w = zeros(len_feature, len_feature);
82  S_b = zeros(len_feature, len_feature);
83
84  for j = 1 : len_class
```

```
85      eval(['S_w' num2str(EU_Com(j)) ' = n_' num2str(EU_Com(j)) ...
86          '/n_all * nancov(par_lda_train_' num2str(EU_Com(j)) ');']);
87      S_w = S_w + eval(['S_w' num2str(EU_Com(j)) '']);
88
89      eval(['S_b = S_b + n_' num2str(EU_Com(j)) '/n_all * transpose(
            m_' ...
90      num2str(EU_Com(j)) ' - m_u) * (m_' num2str(EU_Com(j)) ...
91      ' - m_u);']);
92  end
93
94  % Calculate projection vectors
95  [EV, E] = eig(S_w \ S_b);
96  lambda = abs(diag(E));
97  [~,indx] = sort(-abs(diag(E)));
98  for j = 1 : len_coeff
99      eval(['w_' num2str(j) '(:,t) = EV(:,indx(j,:));' ]);
100 end
101
102 % Calculate transformed vectors for test
103 for j = 1 : len_class
104     for k = 1 : len_coeff
105         eval(['y_test_' num2str(EU_Com(j)) num2str(k) num2str(t) '=
                transpose(w_' ...
106         num2str(k) '(:,t)) * transpose(par_lda_test_' ...
107         num2str(EU_Com(j)) ');']);
108     end
109 end
110
111 for j = 1 : len_class
112     for k = 1 : len_coeff
113             eval(['y_test_' num2str(EU_Com(j)) num2str(k) num2str(t
                ) '= transpose(y_test_' ...
114             num2str(EU_Com(j)) num2str(k) num2str(t) ');']);
115     end
116 end
117
118 % Calculate transformed vectors for training
119 for j = 1 : len_class
120     for k = 1 : len_coeff
121         eval(['y_train_' num2str(EU_Com(j)) num2str(k) num2str(t) '
              = transpose(w_' ...
122         num2str(k) '(:,t)) * transpose(par_lda_train_' ...
123         num2str(EU_Com(j)) ');']);
124     end
125 end
126
127 for j = 1 : len_class
128     for k = 1 : len_coeff
129             eval(['y_train_' num2str(EU_Com(j)) num2str(k) num2str(
                t) '= transpose(y_train_' ...
130             num2str(EU_Com(j)) num2str(k) num2str(t) ');']);
131     end
132 end
133
```

```
134
135 %% Confusion Matrix
136
137 for j = 1 : len_class
138     W = [w_1(:,t) w_2(:,t)]; %%%%%%%%
139     eval(['tp_' num2str(EU_Com(j)) ' = m_' num2str(EU_Com(j)) ' * W
        ;']);
140 end
141
142 for j = 1 : len_class
143     eval(['td_' num2str(EU_Com(j)) ' = par_lda_' num2str(EU_Com(j))
        num2str(t) ...
144     ' * W;']);
145 end
146
147 for i = 1 : len_class
148     eval(['clear D' num2str(EU_Com(i))]);
149 end
150
151 for j = 1 : len_class
152     for k = 1 : len_class
153         eval(['D' num2str(EU_Com(j)) '(k,:) = sum((td_' num2str(
            EU_Com(j)) ...
154             '- tp_' num2str(EU_Com(k)) ').^2,2);']);
155     end
156 end
157
158 for i = 1 : len_class
159     eval(['clear cD' num2str(EU_Com(i))]);
160 end
161
162 for j = 1 : len_class
163     eval(['[~,cD' num2str(EU_Com(j)) ']=min(D' num2str(EU_Com(j)) '
        ,[],1);']);
164 end
165
166 for i = 1 : len_class
167     for j = 1 : len_class
168         eval(['CM(i,j,t) = length(find(cD' num2str(EU_Com(i)) ...
169             '==j)) / length(cD' num2str(EU_Com(i)) ')*100;']);
170         eval(['CM_c(i,j,t) = length(cD' num2str(EU_Com(i)) ...
171             '(cD' num2str(EU_Com(i)) '==j));']);
172     end
173 end
174
175
176 %% Classify
177
178 for i = 1 : len_class
179     for j = 1 : len_coeff
180         eval(['nonnan_1 = y_train_' num2str(EU_Com(i)) num2str(j)
            num2str(t) ';']);
181         eval(['nonnan_1(isnan(nonnan_1)) = [];']);
```

```
182         eval(['nony_train_' num2str(EU_Com(i)) num2str(j) '=
                nonnan_1;']);
183         eval(['nonnan_2 = y_test_' num2str(EU_Com(i)) num2str(j)
                num2str(t) ';']);
184         eval(['nonnan_2(isnan(nonnan_2)) = [];']);
185         eval(['nony_test_' num2str(EU_Com(i)) num2str(j) '=
                nonnan_2;']);
186     end
187 end
188
189 % for i = 1 : len_class
190 %     eval(['num_nan(:,i) = length(y_train_' num2str(EU_Com(i)) ...
191 %         num2str() '1) - length(nony_' num2str(EU_Com(i)) '1);']);
192 % end
193
194 % define train
195 for i = 1 : len_class
196     eval(['len_non_nan_train(i) = length(nony_train_' num2str(
            EU_Com(i))  '1);']);
197 end
198
199 clear group
200 m = 0;
201 for i = 1 : len_class
202     for j = 1 : len_non_nan_train(i)
203         m = m+1;
204         group(m) = num2str(EU_Com(i));
205     end
206 end
207
208 group = group';
209
210 clear aa
211 clear bb
212
213 l = 0;
214 for i = 1 : len_class
215     eval(['aa(l+1:len_non_nan_train(i)+l,1) = (nony_train_' num2str
            (EU_Com(i)) '1);']);
216     eval(['bb(l+1:len_non_nan_train(i)+l,1) = (nony_train_' num2str
            (EU_Com(i)) '2);']);
217     l = l + len_non_nan_train(i);
218 end
219
220 % define test
221 for i = 1 : len_class
222     eval(['len_non_nan_test(i) = length(nony_test_' num2str(EU_Com(
            i))  '1);']);
223 end
224
225 l = 0;
226 for i = 1 : len_class
227     eval(['aa_t(l+1:len_non_nan_test(i)+l,1) = (nony_test_' num2str
            (EU_Com(i)) '1);']);
```

```
228     eval(['bb_t(l+1:len_non_nan_test(i)+l,1) = (nony_test_' num2str
            (EU_Com(i)) '2);']);
229     l = l + len_non_nan_test(i);
230 end
231
232 [X,Y] = meshgrid(linspace(min(aa_t),max(aa_t)),linspace(min(bb_t),
        max(bb_t)));
233 X = X(:); Y = Y(:);
234
235 [C,err(:,t),P,logp,coeff] = classify([X Y],[aa bb],group,'Linear');
236
237
238 %%%%%%% SCATTER PLOT OF ONLY CHOSEN CLASSES %%%%%%%
239
240 % AFS
241 %  figure
242 %  hold on
243 %  scatter(nony_test_F1,nony_test_F2,75,'k','filled');
244 %  %scatter(nony_test_F1,nony_test_F2,75,'r','filled');
245 %  scatter(nony_test_S1,nony_test_S2,75,'b','filled');
246 %  scatter(nony_test_A1,nony_test_A2,50,'r');
247 %  gscatter(X,Y,C,'kby','.',1,'off');
248 %  legend('F-segment','S-segment','Rest segments');
249 %  xlabel('1.LDA'); ylabel('2.LDA');
250 %  title('points from test-set, boundaries from training set');
251
252 % AJ
253 %  figure
254 %  hold on
255 %  scatter(nony_test_B1,nony_test_B2,50,'k','filled');
256 %  scatter(nony_test_C1,nony_test_C2,50,'m','filled');
257 %  scatter(nony_test_D1,nony_test_D2,50,'y','filled');
258 %  %scatter(nony_test_J1,nony_test_E2,80,'r','filled');
259 %  gscatter(X,Y,C,'kbr','.',1,'off');
260 %  legend('B-segment','C-segment','D-segment');
261 %  xlabel('1.LDA'); ylabel('2.LDA');
262 %  title('points from test-set, boundaries from training set');
263
264 % AESJ
265 %  figure
266 %  hold on
267 %  scatter(nony_test_A1,nony_test_A2,50,'g','filled');
268 %  %scatter(nony_test_B1,nony_test_B2,50,'g','filled');
269 %  %scatter(nony_test_C1,nony_test_C2,50,'b','filled');
270 %  %scatter(nony_test_D1,nony_test_D2,50,'m','filled');
271 %  %scatter(nony_test_E1,nony_test_E2,75,'y','filled');
272 %  %scatter(nony_test_F1,nony_test_F2,75,'k','filled');
273 %  scatter(nony_test_S1,nony_test_S2,75,'r','filled');
274 %  %scatter(nony_test_M1,nony_test_M2,50,'g');
275 %  %scatter(nony_test_J1,nony_test_J2,75,'r','filled');
276 %  gscatter(X,Y,C,'kbyg','.',1,'off');
277 %  legend('A','S');
278 %  xlabel('1.LDA'); ylabel('2.LDA');
279 %  title('points from test-set, boundaries from training set');
```

```
280
281 % all classes
282 %  figure
283 %  hold on
284 %  scatter(nony_test_A1,nony_test_A2,75,'m');
285 %  scatter(nony_test_B1,nony_test_B2,50,'m','filled');
286 %  scatter(nony_test_C1,nony_test_C2,50,'k','filled');
287 %  scatter(nony_test_D1,nony_test_D2,50,'r','filled');
288 %  scatter(nony_test_E1,nony_test_E2,50,'g','filled');
289 %  scatter(nony_test_F1,nony_test_F2,50,'b','filled');
290 %  scatter(nony_test_S1,nony_test_S2,75,'b');
291 %  scatter(nony_test_M1,nony_test_M2,75,'k');
292 %  scatter(nony_test_J1,nony_test_J2,75,'r');
293 %  gscatter(X,Y,C,'crgbmky','.',1,'off');
294 %  legend('A-segment','B-segment','C-segment','D-segment', ...
295 %      'E-segment','F-segment','S-segment','M-segment','J-segment')
    ;
296 %  xlabel('1.LDA'); ylabel('2.LDA');
297 %  title('points from test-set, boundaries from training set');]
298
299 % Economy | Slient | Sport
300 %  figure
301 %  hold on
302 %  scatter(nony_test_A1,nony_test_A2,75,'r');
303 %  scatter(nony_test_E1,nony_test_E2,75,'g');
304 %  scatter(nony_test_S1,nony_test_S2,75,'b');
305 %  gscatter(X,Y,C,'crgbmky','.',1,'off');
306 %  legend('Economy (A,B,C,D,M,J)','Silent (E,F)','Sport (S)');
307 %  xlabel('1.LDA'); ylabel('2.LDA');
308 %  title('points from test-set, boundaries from training set');
309
310 end
311
312 %% calculate projection error between 4 training processes
313 m = 1;
314 for i = 1 : 3
315     for j = i+1 : 4
316         err_w(m,1) = 1 - abs(w_1(:,i).' * w_1(:,j) / (abs(w_1(:,i))
            .' * abs(w_1(:,j))));
317         err_w(m,2) = 1 - abs(w_2(:,i).' * w_2(:,j) / (abs(w_2(:,i))
            .' * abs(w_2(:,j))));
318         %err_w(m,3) = 1 - abs(w_3(:,i).' * w_3(:,j) / (abs(w_3(:,i)
            ).' * abs(w_3(:,j))));
319         m = m+1;
320     end
321 end
322
323 % figure
324 % imagesc(err_w)
325 % colorbar
326 % xticks(1:3);
327 % xticklabels({'w_1', 'w_2', 'w_3'});
328 % xlabel('projection vectors');
329 % yticks(1:6);
```

```
% yticklabels({12, 13, 14, 23, 24, 34});
% ylabel('between two folds');
% title('projection vector errors between two folds');

%%

[CM_c sum(CM_c,2)]
round(mean(CM,3))
err
err_w'
%w_1
%w_2
par_lda_name
```