Bachelor´s Thesis
Double Degree in Physics and Electronic Engineering
(Electronic Engineering Degree)

Analysis of the relationship between the signaling obtained from CAN-BUS and comfort assessment variables based on artificial intelligence

Author:
Jon Wigerfelt Holgado
Advisors:
Asua Uriarte, Estibaliz

Leioa, 22 June 2022 / Leioa, 2022ko Ekainaren 22a / Leioa, 22 de Junio de 2022

Index

1 Introduction and Objectives

The growing urbanization as ongoing societal trend requires novel mobility concepts for people and goods. Having more cars in the already crowed city will not help. The space to build more roads is limited. The only sustainable way is to optimize the usage of the limited infrastructure by high autonomy and automation of the mobility platforms (vehicles) in such urban environments. New opportunities for automated driving, driverless delivery of goods to stores and residences, robot taxis, and driverless public transportation will allow to maintain reliable services independent of the severity of the heath crisis while safeguarding the health risk for the driver and the passengers [1, 2, 3].

The European Union (EU) reports estimate that over 90% of road accidents occur due to human error [4], for instance, due to erroneous perception, incorrect decision-making, distraction, or otherwise. Further, the inability of human drivers to perceive and account for road and environmental factors beyond their line of sight results in inefficient vehicular operation (wasted energy due to frequent stop and go, inefficient path planning, traffic imbalances, and congestion). Autonomous vehicles have the potential of solving those issues and to improve traffic and fuel efficiency [5, 6].

In this scope, the interest in evaluating the comfort experienced by passengers in vehicles has raised. Not only with the objective of accommodating the passengers, but preventing health issues in form of headaches or fatigue. Perhaps the prime objective of an automated cabin is to be able to 'multitask' within a journey and increase the inherent value of that journey by enabling additional productivity, enjoyment and improved well-being whilst being driven [7]. This is more appropriate for longer trips (above 30 minutes), but even so to maximise productivity, many manufacturers concepts depict fully flexible seating within an office-like environment. This poses many challenges, like being able to function with complex tasks while subjected to motion.

Evaluating comfort is a complex topic since welfare is related to physical, psychological and physiological variables. But, general comfort can be categorized in two classes: environmental and riding. Environmental parameters include thermal comfort, air quality, noise, pressure gradients, etc, they relate to the judgments made by humans on the surrounding conditions. Specific values and ranges classify each of them as presented in the work of Silva [1]. Variables such as temperature and humidity are responsible for responses in the form of sweating and heavy breathing, while others, like noise, pressure and brightness are responsible for consequences along the lines of malaise and dizziness.

Riding comfort corresponds to dynamic factors, related to vibration, shocks, and acceleration. These parameters are the main responsible for producing motion sickness and general feeling of discomfort among other effects. Motion sickness is a result of the conflict between the human body's vestibular and visual sensory systems. It is mainly caused due to a contradiction between the sensed motion (the motion as sensed via our sensing systems) and the expected motion [8]. Predicting the future path of the vehicle trajectory from radially expanding optic flow reduces the build of motion sickness. Therefore, the driver having an anticipation of motion leads to a good match between the expected and sensed motion and reduced motion sickness. Consequently, passengers were more prone to motion sickness as they did not maintain visual references and focused on static scenes within the vehicle's interior [9]. In addition, an increase in sickness has been detected in passengers when performing secondary tasks and travelling rearward [7]. Average ride discomfort commonly originates from vibrations, accelerations and its time derivative, jerks, that affect ride quality and passenger posture. High values of accelerations or jerks can cause discomfort even during shorter periods of time [10]. Some kinematic parameters, such as braking and turns, can be considered as significantly affecting passenger comfort. As with motion sickness, the driver is less affected by the vehicle handling by taking preventive actions, while passengers have to respond to the maneuvers. Both general discomfort and motion sickness can be linked to specific aspects, such as route roughness, driving behaviour in terms of breaking, steering, and accelerating.

From the biomechanical point of view each part of the human body is represented by its equivalent mass, and elastic and damping elements. The natural frequencies of the different parts are also indicated in [11]. These vibrations are the main responsible for the discomfort felt by passengers and depend on direction and frequency. Therefore, in order to assess and evaluate the oscillations and their consequences the International Standardization Organization (ISO) developed a standardization and quantification system for this matter in the form of International Standard 2631 (ISO-2631-1) [12]. The quantification of the vibration level is done through the root mean square (RMS) of the signal corresponding to the instantaneous acceleration felt by a given body. For vehicle occupants, two frequency ranges are considered: 0.5–80 Hz for health, comfort, and perception; and 0.01–5 Hz for motion sickness. Furthermore, maximum motion sickness values have been regarded for frequencies around 0.2 Hz [13] and higher frequency vibrations are attenuated by the human body and reduce the discomfort. It is also interesting to note that monotone continuous low frequency vibrations increase fatigue, while transient vibrations produce stress.

The data used in this work originates from the UYANIK study [14] in Istanbul. It is part of an international research were 5 countries and the vehicle data from 450 drivers is collected. The data in this work belongs to a specific road type, specifically selected to minimize the effect of factors outside the driving behaviour. Moreover, the stretch is chosen to produce a better classification, since it provides more diverse data and it is better distributed.

With the intention of implementing machine learning techniques, the data requires being preprocessed through a chain of cleaning, normalizing, categorising and oversampling, to be suitable for the algorithms. The results of the techniques depend directly on the information provided during their learning phase. Thus, administering quality data is prime for a optimal performance.

The main objective is to implement a series of machine learning techniques to analyze the relationship between the handling of the vehicle and the general discomfortness produced by these actions. The results of the techniques will validate the use of each one and will extract the characteristic operations originating the discomfort. Analyzing the data with different algorithms, also intents to provide more knowledge for implementing other machine learning techniques.

From the available machine learning techniques, the implemented supervised Radial Distribution Function, focuses on predicting the effect different parameters have in the comfort, by making use of previously labeled data; while the unsupervised Self-Organized Map (SOM), which has been proven useful to classify driving styles [15], is centred in recognizing discomfort generating patterns, by learning with unlabeled data. They both act using Euclidean distances between vectors, but apply the calculations in distinct manner. They have been implemented using open source Python libraries.

Using the results of these techniques, this work also aims to obtain reliable distinction between driving styles, from a comfort perspective, based on the riding parameters. Contradistinguishing the parameters that have a comfortable ride as an outcome will allow to create recommendations for drivers based on their driving characteristics. This recommendations will be in the form of gas, break and steering wheel operation, which could be implemented in an advanced driving assistance system.

2 Comfort evaluation methods

The overall sensations of vibration discomfort depend upon the sensing as well as the frequency content of the stimulus producing them. One of the main standards for comfort is ISO 2631-1, where the RMS of the accelerations along all axes of the vehicle are defined to reflect more closely the health hazard to which the human body is exposed on the basis of the frequency and the direction of vibrations. To do so, ISO 2631-1 defines several filters to better match the different discomfort categories to specific frequencies.

As shown in Figure 1, three filters, w_f, w_d and w_k, divide the spectrum through the aforementioned significant range of 0.01-80 Hz. w_f is related to motion sickness, notice its peak is situated around 0.2 Hz, with the frequency range between $w_{f1} = 0.1$ Hz and $w_{f2} = 0.3$ Hz is defined as the most significant by ISO 2631-1. On the other hand, w_d and w_k mould the horizontal and vertical acceleration respectively, being responsible for weighting the general discomfort connected to the higher frequencies.

Following the work in [12], the weighted RMS acceleration values for each axis is expressed as,

$$a_{w_j,i} = \sqrt{\frac{1}{N}\sum_{k=1}^{N} a_i^2, w_j(k)}, \tag{1}$$

where i determines the direction, w_j is the corresponding filter and N is the number of samples of the acceleration data.

As for general comfort, only one parameter is presented, a_v. It accounts for the total weighted acceleration considering vertical, lateral and horizontal axes. As it is expressed in (2), the measured accelerations are first weighted with the corresponding filters and then multiplied by some factors which account for the perception of the acceleration and the direction of the measurement.

$$a_v = \left(k_x^2 a_{w_{d,x}} + k_y^2 a_{w_{d,y}} + k_z^2 a_{w_{k,z}}\right)^{1/2} \tag{2}$$

In [1, 16], for evaluation of the health effects and with the intend of adjusting the calculated values of a_v to human perception, the weighting parameters have been defined

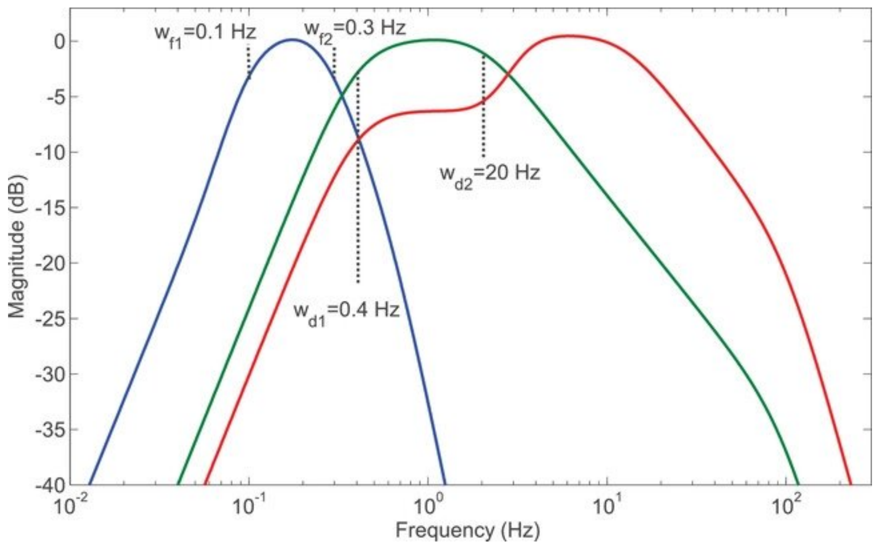


Figure 1: Amplitude responses of different weighting filters in ISO 2631, w_f : motion sickness (blue), w_d: global comfort horizontal-component (green), and w_k: global comfort vertical-component (red). Figure taken from [15]

Table 1: Comfort index (a_v) and its relation with general discomfort.

Range of a_v (m/s^2)	**Comfort level**
greater than 2.0	extremely uncomfortable
1.25-2.5	very uncomfortable
0.8-1.6	uncomfortable
0.5-1.0	fairly uncomfortable
0.315-0.63	a little uncomfortable
less than 0.315	not uncomfortable

as $k_x = 1.4$, $k_y = 1.4$ and $k_z = 1$. If the comfort is evaluated for passengers as this case, all the coefficients k_i will be one [17]. Depending on the value of a_v, the ride comfort can be classified with different comfort grades as the Table 1 indicates. This standard can be used individually or in combination with acceleration and jerk peaks.

There is a concrete form of (1), where $w_i = w_f$, so that only the filter for motion sickness is applied. This allows to define the motion sickness dose value (MSDV), which is expressed as,

$$MSDV_i = \sum_{k=1}^{N} a_i^2, w_f(k), \tag{3}$$

where i is the specific axis for which the MSDV is evaluated. The MSDV is one of the methods used to objectify the motion sickness ratings and was defined in accordance with ISO 2631-1. In some works an illness rating method, derived from the MSDV has been utilized [12, 16].

These standards are focused on the vertical vibration, which are a prime cause of seasickness and the MSDV model has been used to predict sickness on ships [18, 19]. However, transverse oscillations have shown to posses a greater relation to motion sickness in road transport [20]. Moreover, other works reported that the w_f frequency weighting, as stated in ISO 2631-1, did not give good predictions of the incidence of motion sickness in road coaches and suggested that they were not optimum for predicting sickness in cars [7, 21, 22].

Besides ISO standard, acceleration threshold theory has been utilized in [17, 23, 24, 25, 26], as done in this work, to evaluate the general discomfort. In this model, acceleration and jerk peaks are studied to assess the discomfort. In [24, 25], the threshold has been set around $1.5\,\mathrm{m/s^2}$ for the acceleration and $0.6\,\mathrm{m/s^3}$ for the jerk. Since they stand for studies conducted with buses and an automobile only carries seated passengers, it is expected that the thresholds should be set on the higher.

Therefore, in this work the evaluation has been done following the study in [15], where while using cars, the threshold has been set to $1.70\,\mathrm{m/s^2}$. Thus, the evaluation is done by counting the acceleration peaks with values above a certain threshold, such that:

$$n_i = n_i + 1,\ when\ a_i > Threshold, \tag{4}$$

where i determines the axis for which the counting is done. In addition, as researched in other works, differentiation between positive and negative acceleration peaks has been done, due to the relationship between positive acceleration and the gas pedal, and negative accelerations and the break pedal [27]. This distinction aims to better connect the discomfort felt with the drivers actions.

Furthermore, with the intent of relating the comfortness with the driving style, (4) only was implemented in the x and y directions. The z axis was discarded since it is independent to the drivers handling of the vehicle, and therefore of no use to create driving recommendations besides "avoid bumps".

3 Machine learning techniques

Machine Learning has become one of the most compelling areas of computer science research. Artificial intelligence being its core, is propelling the field further than ever before. The realm of machine learning techniques (MLTs) is divided into two main classes. The supervised machine learning techniques and the unsupervised machine learning techniques.

Supervised learning reflects the ability of an algorithm to generalize knowledge from available data with target or labeled cases so that the algorithm can be used to predict new (unlabeled) cases [28]. Supervised machine learning requires predetermining an output attribute besides the use of input attributes. The supervised learning step is where each training input pattern is accompanied by its correct classification of the predetermined attribute and "fed" to the algorithm. The difference between the algorithms current response and the desired one is used to change the internal properties of the algorithm. Of course, such an approach must do more than just learn the class exemplars since all that the fully trained technique would be able to do would be to template match on these exemplars. It must be able to generalise, that is to correctly classify the exemplars by recognizing similar patterns [29]. Pattern recognition might not be able to fully correlate unlabeled inputs to their corresponding output class. Therefore, some MLTs are designed to provide as a result the accuracy of the algorithm when classifying the input into the different output classes. In those cases, a percentage value is given for every output class, which represents the algorithms confidence level that the input belongs to each corresponding output class.

Conversely, *unsupervised learning* involves pattern recognition without the involvement of a target attribute. It refers to the process of grouping data into clusters using automated methods or algorithms on data that has not been classified or categorized. In this category learning is also called unsupervised, competitive, or self-organizing [30]. Algorithms must "learn" the relationships or features from the available data and group cases with similar characteristics. To do so, unsupervised learning algorithms attempt to identify several prototypes in the data set that can serve as cluster centres. Unsupervised learning schemes can identify the natural structure in patterns but they can not, in general, discover the hidden structure [28]. This is specially useful when the natural structure is not obvious, but on the other hand, this means that unsupervised learning can not solve the exclusive-OR (XOR) problems, unless the architecture of the network forces such an association. Accordingly, when small amounts of data are available, the learning is specified as "semi-supervised" [28].

Unsupervised learning may appear, therefore, to be a poor relation to supervised learning. This is not the case, especially, where we are concerned with perceptual data (such as speech or vision) where supervised techniques assume that we already possess an accurate model of the underlying processes for assigning class membership, or at least, a knowledge of the number of classes and examples of each [29]. Since that might be wrong, unsupervised learning algorithms are suitable for creating the labels in the data that are subsequently used to implement supervised learning tasks.

There exist numerous MLTs, each one with its own properties and objectives. Explaining the specifics for each of them is out of the scope of this work. Therefore, only a few examples for each of the learning classes are now briefly mentioned.

Among supervised techniques, the neural networks are one of the most common types of neural network architectures currently used [31]. Neural network architectures are inspired by the structure of the brain. They are constituted by nodes, which resemble neurons, and their weights, resembling the connections between neurons. The simplest kind of neural network is a single-layer perceptron network [29], which consists of a single layer of output nodes, and it can be used as a binary classification model. More complex architectures form Multi-layer perceptrons, Feedforward Neural Networks, Radial Basis Networks, Deep Convolutional Networks, etc. Other supervised techniques like the Decision Tree algorithm emulate a tree, while the Naïve Bayes algorithm is cemented on Bayesian probability theorem [28].

Examples of unsupervised techniques are the Boltzmann Machine, the Autoencoder and the Self Organized Map. The Boltzman Machine describes a physical system using a network of units with constant "energy", and it produces binary results [32]. The Autoencoder has two main parts: an encoder that maps the input into the code, and a decoder that maps the code to a reconstruction of the input. It is commonly used for writ-

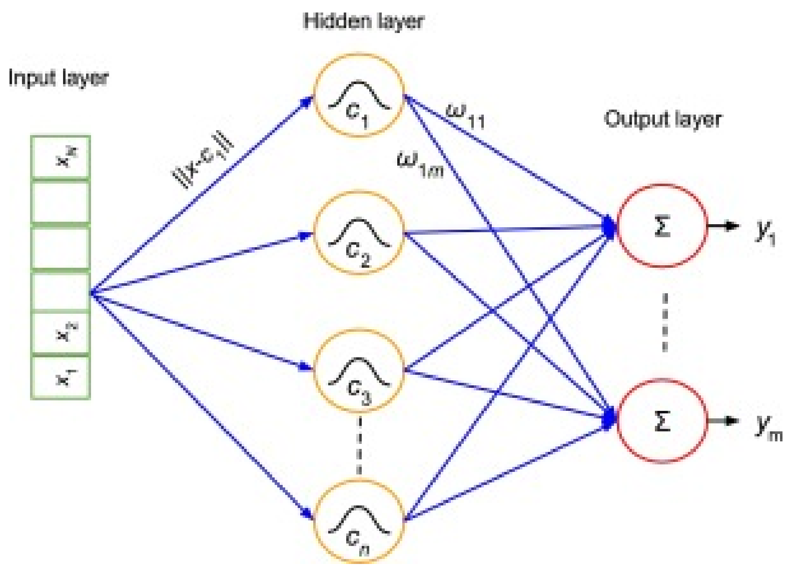


Figure 2: General scheme of a classifier RBFNetwork. Figure from [36]

ing prediction [33]. Finally, the Self Organized Map consists of sheet-like artificial neural network, where neighboring cells compete in their activities by means of mutual lateral interactions [30]. It is a great tool for visualization since it produces a map connecting the input and output spaces.

In this work, a supervised and an unsupervised MLT have been implemented, the Radial Basis Function Network and the Self Organized Map respectively. They have been selected since they both make use of Euclidean distance calculations. This allows to make a comparison between supervised and unsupervised algorithms with similar working basis. The theory and working principle of both techniques are explained in the following sections.

4 Radial Basis Function Network (RBFNetwork)

The Radial Basis Function (RBF) neural network, also named RBFNetwork or RBFNN, is a feedforward supervised network first introduced to the literature by Broomhead and Lowe (1988) [34]. At the time, it was used to interpolate due to the polynomial form the outputs present. The interpolation implementation has been found to be specially effective in task where interpolating long vectors is needed, when enlarging digital images for example. But more recently, it has become a strong classification tool, and thus, the focus in this chapter will be set on this property, since it is the one implemented.

The origin of the RBFNetwork relies in locally tuned nervous systems. For example, cells in the auditory system, selective to small bands of frequencies, or cells in the visual cortex sensitive visual features within a small region of the visual field [35]. Provided by a multi variable input space, these locally tuned neurons characteristically respond to a small range of the input space.

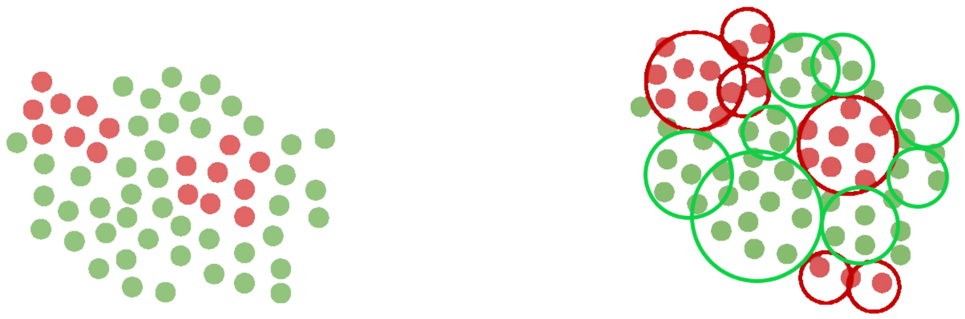


Figure 3: Different classes of point (red and green) in 2D space. On the right, the circles represent the radial basis functions mapped in the space according to the two point classes.

4.1 Working Principle

The RBFNN, in its simplest form only consists of three layers. A hidden non-linear layer is located between the classic input and output layers of the network. The general scheme is shown in Figure 2. The inputs are n-dimensional vectors, but to understand how the RBFNN works it is easier to start with a 2 dimensional case. Points in 2D space can be represented with (x, y) coordinates, which will form their position vector $\vec{v}$. Considering a set of points, where the points belong to different classes, the neighbours from the same class can be encircled and the circles differentiate the groups as shown in Figure 3. In reality, these circles are smooth drop off radial distribution functions, and the circles can be representative of the distance at which the distribution function obtain a certain percentage of the maximum value. The most common radial distribution function implemented is the Gaussian Distribution,

$$f(\vec{v}) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{||\vec{v}-\vec{c}||}{\sigma}\right)^2}, \tag{5}$$

where c, also called centroid, is the position vector of the center of the distribution and σ denotes the radius of the distribution, the bigger the σ, the slower the radius drops off, making the circle bigger.

The entirety of the points can not be accurately grouped using a limited number of circles with a fixed radius, but it is argued that with an unlimited set of circles and with different size circles the entirety of the space can be accurately mapped as shown in Figure 3. The limit of this argument consists of every point of the 2D space having its own small circle. For a n-dimensional space the working principle is the same, but instead of having circles the space will be distributed in n-dimensional hyperspheres.

So, a crucial part in a RBFNN is to select the number of centroids and their radius. Normally, the number of centroids k is fixed in the beginning, since the centroids will form the hidden layer of the network. Arguably, the accuracy of the network will increase as the number of centroids increases, at the cost of computational time and power. Therefore, having some knowledge of how the data is distributed helps with the selection of centroids, for instance, in [37] a Feature Selection technique is proposed for this task. A common but less effective method, is to evenly lay out the centroids. This only works for spread data sets. The most usual approach, is to use a series of vectors outside of the input layer. For the radius of the hyperspheres supervised learning is habitual. First, the radii are set randomly in some range and then modified through the learning process. However, finer initial radii can provide superior and faster results.

After the hidden layer is set, a random weighting w_{ij} between -1 and 1 is given to each connection from the centroid i to the output j. This weighting represents the importance the i centroid has in the j output. If the centroid c_i encapsulates the point of class j, then the value of w_{ij} will be high, and the value will be low in the opposite case. This weights are also updated in a supervised manner.

Since the RBFNetwork is a supervised MLT, a previous classification of the data is required. This classification is also the expected output of the network. The number of categories m, will define the number of outputs in the output layer. When a new input vector is fed to the algorithm, the distribution function f_i in (5) is computed for every centroid and the output $\vec{y}$ is algebraically calculated as follows:

$$\begin{bmatrix} w_{11} & w_{21} & \cdots & w_{n1} \\ w_{12} & w_{22} & \cdots & w_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ w_{1m} & w_{2m} & \cdots & w_{nm} \end{bmatrix} \cdot \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_k \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} \tag{6}$$

The output vector is representative of how confidently the algorithm classifies the input vector in the defined categories. For example, the y_1 output represents the probability of the input vector being part of the class number 1. If the σ of the distribution functions are forced to be low, meaning the drop off is more sudden, then, the network can be used to predict the class of the input vector by looking at the output with the highest value, which should be significantly bigger than the rest of the outputs.

During the training process, the output $\vec{y}$ is compared with the expected output $\hat{\vec{y}}$ of the input vector. When different, adjustments are made on the radius σ_i and w_{ij} weights using a gradient descent algorithm. Thus, for a iteration n, their values are adjusted as follows:

$$\sigma_i(n+1) = \sigma_i(n) + \beta \cdot f_i \cdot (\hat{y}_i - y_i)^2, \tag{7}$$

$$w_{ij}(n+1) = w_{ij}(n) + 0.01 \cdot \beta \cdot f_i \cdot \left(\frac{||\vec{v}_n - \vec{c}_i||}{\sigma_i} \right)^2 \cdot \left(\sum_{l=1}^{k} w_{il} \cdot (\hat{y}_l - y_l)^2 \right), \tag{8}$$

where β is the learning rate, $\vec{v}_n$ is the input vector in the iteration n and f_i is the results of (5) for the centroid c_i.

4.2 Interpretation

After the training process, a series of metrics are considered to determine the effectiveness of a supervised machine learning algorithms. In this work the accuracy and the mean square error (MSE) are studied.

The accuracy is the simplest metric, is the percentage inputs categorized in the right class. To do so, the element of the output vector with the highest value determines the predicted class of the input. Then the prediction is compared with the actual class of the input. For a number of inputs, the ratio of the correct predictions determines the accuracy.

The MSE is the average deviation of the categorization. It is the average mean square error between the predicted outputs and the actual outputs. It is mathematically represented in (9), where N is the total number of inputs. The smallest possible MSE is desired, which would indicate high precision of the algorithm and high reliability of the accuracy.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(\hat{\vec{y}_i} - \vec{y}_i)^2 \tag{9}$$

The disadvantage of these metrics is the lack of directional prediction. They do not inform about which classes are being over predicted and which are being under predicted. However, they are easy to implement and give direct information about the performance of the algorithm. In order to maximize the accuracy and minimize the MSE, the number of centroids, their location and the initial radius of the distribution function can be modified. The results obtained for the particular objective of this work, by changing these parameters in the RBFNN, are presented in Section 8.

5 Self Organized Maps (SOM)

Self-Organizing Map (SOM) networks, also known as Self-Organizing Feature Maps or Kohonen Networks, are unsupervised neural networks based on competitive learning. The Self-organizing Map presented in this work belongs to the self-organized category. It is a 2 dimensional sheet-like artificial neural network, where the neurons, after an unsupervised learning process, recognize patterns or classes of patterns across the inputs.

The structure is based in those parts of the mammalian brain, associated with sensory processing (eg, the visual and auditory cortex), where the existence of low, usually two, dimensional neural maps that reflect the topological ordering of the external stimuli [29].

The ability of the SOM to perform, in an unsupervised manner, dimensionality reduction and topological organisation means that they have been employed in a number of application areas, such as statistical encoding, data compressing and classification [29, 30]. The utility of SOMs for advanced driving assistance systems has already been certified [15].

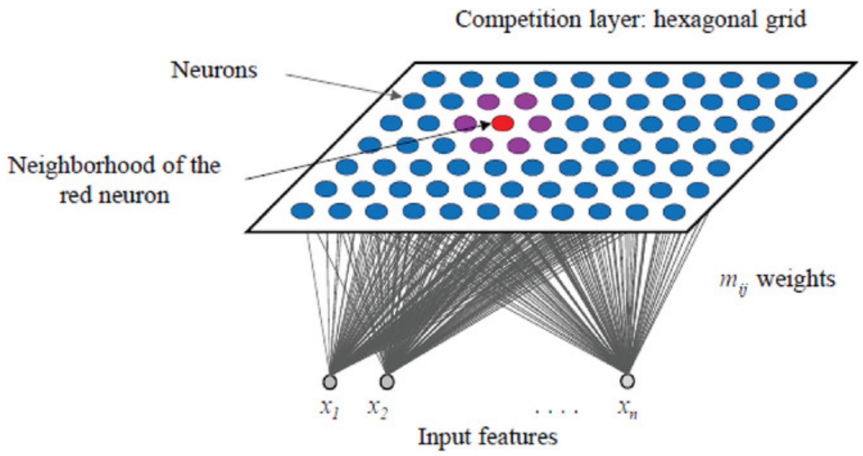

Competition layer: hexagonal grid
Neurons
Neighborhood of the red neuron
m_{ij} weights
x_1
x_2
....
x_n
Input features

Figure 4: Typical hexagonal SOM topology: structure of an N-input SOM, $\vec{x} = (\vec{x}_1, \vec{x}_2, ..., \vec{x}_N)$, and $K = 77$ output neurons distributed into a 7x11 two-dimensional hexagonal grid. Figure from [15].

5.1 Working principle

5.1.1 SOM description

To describe the SOM map first assume a sequence of statistical samples of data vectors $\vec{x} = \vec{x}(n) \in \Re^N$, where n is the iteration number and N is the number of features of each neuron, and a subset of variable reference vectors $\vec{m}_i(n)$, $\{\vec{m}_i \subset \vec{x}(n) : i = 1, 2, ..., M\}$ [30]. The structure of a SOM is compressed as a linear n-dimensional grid. For simplicity a 2D single-layer linear grid will be used, which is also called the competitive layer.

The neurons in the 2D grid can be referred as nodes and are instantiated with reference vectors $m_k(n)$, thus, the number of neurons in the competitive layer, which is M, must be proportional to the size of the map. For instance, a map containing L neurons in the primary axis and H in the secondary axes (square maps are most common for symmetry reasons, specially when periodic boundary conditions are not applied) would require $M = L \times H$. Each node can be presented with its own (i, j) coordinate, thus, $\vec{m}_i$ can be substituted by $\vec{m}_{ij}$, which are the weight vectors of the map, also called output vectors. The remaining data vectors are denoted as the input vectors which are responsible for training the map.

All the nodes on the SOM map grid are connected directly to every input vector, but not to one another, meaning the nodes do not know the values of their neighbours, and only update the weight of their connections as a function of the given inputs.

5.1.2 SOM initialisation

The first thing to be set in a SOM is the size of the map itself. In general, the optimum size of the output map depends on the expected number of features present in the data. As a baseline, Vesanto´s rule in [38] defines the optimal number of neurons as $M = 5\sqrt{K}$, where K is the total number of samples. Distinctively, $N \times N$ maps are argued as ideal in

[31, 39], where N is the number of features of the input vectors. Hence, a certain amount of experimentation may be necessary in order to achieve the correct size of the array.

The topologies of the map lattice that can be found in literature are nearly always square or hexagonal. The square topology was the first one presented by Kohonen and has the advantage that it fits the topology of the common output devices (screens, printers). The hexagonal grid, as shown in Figure 4, maximizes the number of neighbours per neuron. This topology enhances the adaptation of the map and updation of weights during the learning process, since more neurons are considered as neighbours. It also enhances the visualization tool kit and cluster formation respect to a square grid.

To initialise the weight vectors in some proper way, random selection will often suffice according to Kohonen [30]. However, initial arrangement of the weights in the SOM influences directly the final state of the map, since it conditions the formation of clusters. A random distribution of initial weights may not be optimal, resulting in sparsely populated trainable nodes and poor classification performance [31]. A bad initial state of the grid may result in a bland map, where no collection of neurons can be distinguished from one to another.

To avoid this issue, some solutions appear in the literature. In [15], an auxiliary SOM is used to create the cluster centers for the final SOM map. In [**?**], few solutions depending on the linearity of the data are proposed. But for quasi-linear data, like the one in this work, the suggested solutions do not show any improvement over a random initialisation. Other solutions described in [31] are: initialising the weights to the same value and lumping of input vectors with similar values, which ideally increases the likelihood of all nodes being closer to the pattern vector; or adding random noise to inputs to distribute vectors over a larger pattern space.

An independent idea is to order the neurons depending on their modules. In this method, first, the neuron closest to the origin is placed at the (0, 0) position on the grid. With a linear alignment, the second closest neuron is placed in the (0, 1) position and so on with the rest.

The final feature to consider when designing a SOM, is the condition of the periodic boundary conditions (PBCs). When PBCs are taken into account, the map is considered to be repeated in every direction. This means that neurons in the edges have the neurons in the opposite edges as neighbours. Not considering PBCs highly fastens the training process.

5.1.3 SOM training

Once the weights have been initialised is time to train the SOM. The training requires to update the weights of the map K times, where K is the number of input vectors in the input vector set. In each iteration a random input vector x_k is chosen from the input vector set. Then the distance between every node and the input vector is calculated, normally the Euclidean distance by the means of the Pythagorean theorem is used as the distance metric.

$$d_{ij}(\vec{x}_k) = \sqrt{||\vec{x}_k - \vec{m}_{ij}||^2} \tag{10}$$

When all the distances are computed, the best matching unit (BMU) is selected. Meaning, the output vector with the smallest distance to the input vector is chosen. The BMU denoted as $\vec{m}_c$ by Kohonen, is the closest vector in the map respect to the input vector. This process is why the SOM is considered a competitive learning technique, since the output vectors "compete" in order to be the BMU.

Immediately after choosing the BMU the weights in the SOM are updated. For each iteration of the SOM, the weight vectors are updated as follows,

$$\vec{m}_{ij}(n+1) = \vec{m}_{ij}(n) + \alpha(n)\beta_{cij}(n)d_{ij}(\vec{x}_k(n)) \tag{11}$$

where n denotes the aforementioned iteration step, $x_k(n)$ is an input sample randomly selected from the training set at iteration n, β_{cij} is a neighborhood function or kernel around the best matching unit and $\alpha(n)$ is the learning rate.

Both $\alpha(n)$ and β_{cij} are decreasing functions approaching zero with each iteration. Specifically,

$$\alpha(n) = \alpha_0 \cdot exp\left(\frac{-n}{\tau}\right), \tag{12}$$

$$\beta_{cij} = \beta_0 \cdot exp\left(\frac{-d_c^2}{2\sigma^2(n)}\right), \tag{13}$$

where α_0 and β_0 are the initial learning rate and neighbourhood function values; τ is the time constant responsible of decreasing the learning rate and $\sigma(n)$, which is the radius of the neighbourhood function β_{cij}, concretely, $\sigma(n) = \sigma_0 \cdot exp(-n/\tau)$; and d_c is the distance to the BMU. It can be observed that when n, and thus, the number of input samples K tends to infinity only the BMU will be updated.

It may be appropriate to observe here that if the maps are used for pattern recognition, their classification accuracy can be multiplied if the cells are fine-tuned using supervised learning principles [30].

5.2 Interpretation: U-matrix

The SOM is an excellent MLT for result visualization. Once the training phase has ended, the distances between every neuron and its first and second neighbours are computed and added. This summation can be named as the "weight difference" of the neuron respect to its neighbours, and the mapping of the weight difference of all the neurons produces the weight map, also called U-matrix. With the U-matrix, the map can be colored by assigning colors depending on the value of the weight difference. The left image in Figure 5 displays a U-matrix, with colder colors for the neurons with lower weight difference. Clearly, cold and hot spots are pictured, showing the formation of clumps. The blue areas indicate the neurons which have been selected more times as the BMU during the training process and are therefore more similar to their neighbours.

The samples can be mapped on the weight map according to their corresponding BMU. Once all the samples are mapped, clusters can be identified with mathematical implementations like the k-Nearest Neighbors (kNN), see the right image in Figure 5. In

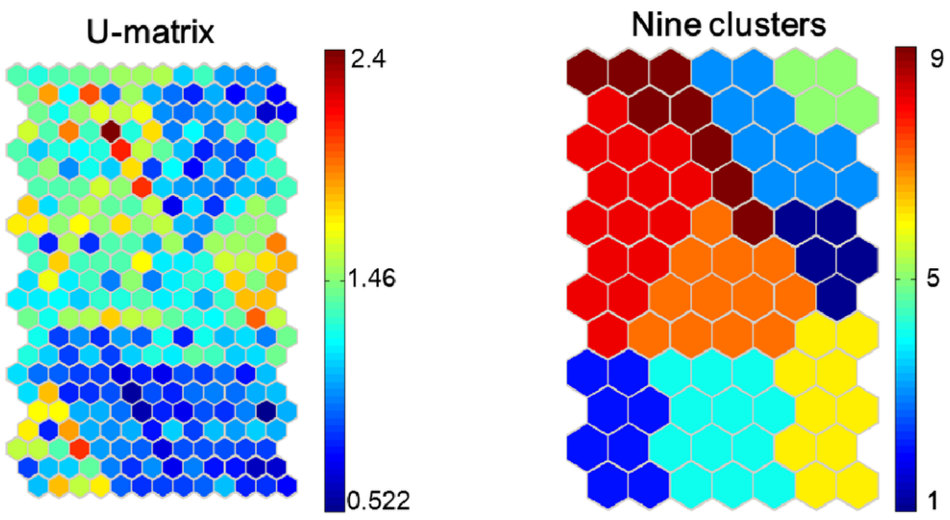


Figure 5: On the left, a U-matrix of 23 x 13, colored from blue to red according to the weight difference bar on its right. On the right, cluster arrangements of nine clusters formed with k-NN algorithm. Figures from [40].

this work the Quality Threshold (QT) computational technique has been used to form the clusters. Other clustering methods like the k-Nearest Neighbours and Density Based Scanning where implemented with little success, since due to the particular characteristics of the SOMs they were not able to properly distinguish clusters.

The QT algorithm ensures that the distance between any two neurons within a cluster should be below a given threshold. First, a random neuron is selected. Then, the distance, calculated according to (10), between the selected neuron and a neighbour is computed. If the distance is smaller than the threshold value, then the neighbour becomes part of the cluster. This calculation is done for every first neighbour, and then with the second neighbours and so on. As the cluster grows, the condition of the distance between every current neuron in the cluster and a candidate neuron being smaller than the threshold value has to be ensured. Finally, in the case that no neighbour neuron fulfills that condition, a new random neuron is selected to form a new cluster.

Due to its mode of action, the QT algorithm is ideal to implement, since the Euclidean distance follows the working principle of the SOM as described in Section 5.1.3. The QT algorithm generates mostly non-overlapping clusters. However, its weakness is the formation of clusters with too little elements. These clusters remain outside of the main clusters as outliers. Therefore, a minimum number of neurons are needed to recognize a cluster as meaningful, the outliers are considered particular cases with no value for pattern recognition.

With the clusters formed, to interpret the results, the means and the variances for all the variables are calculated for each cluster. This representation has shown to be the most useful since it extracts the core information of the clusters. Moreover, this way, the clusters can be categorized according the value of a certain parameter called criteria, which will define the cluster. Other representations, such as picturing the values of the variables in two and three dimensional grids was of no use, since some variables are overlapping and the images become blurry and uninterpretable.

The desired characteristics for a SOM are the formation of distinguished clusters in the U-matrix and distinctive classification for the criteria with little variance and overlapping. The most important one being the small variance of the criteria for the clusters. This implies that the comfort of the samples in the cluster is well defined. Additionally, this also results in the variance of vehicle variables being small, allowing to relate more precisely the values of the other variables with the discomfort. When overlapping of the criteria happens, the rest of the variables need to be observed, since different parameters may produce the same discomfort level.

These aspects decide the validity of the SOM and to optimize those features, some parameters can be modified and specified. The most important parameters to be defined are the dimensions of the map, the initial arrangement of the weights, if periodic boundary conditions (PBCs) are considered, how many samples are going to be used for the training and how many samples to use to map and form cluster. The observations done for each one of these features are presented in the Section 9.

6 UYANIK database

The dataset used in this work emerges from the real-world database collected with "UYANIK" [14]. The research, done by the University of Sabançi at Istanbul, has the international support of several research groups in the United States, Japan, Turkey, Italy, and Singapore, which form an alliance (NEDO Alliance) to share the worldwide driving, road, and vehicle-specific data obtained from 450 drivers. Toward that end, three vehicles ("Uyanik" in Istanbul, "Nagoya Vehicle" in Japan [41], and "UT-Drive" in Dallas, Texas, USA [42]) have been equipped in cooperation with each other. The unification of these researches permits a head-on comparisons between different drivers more easily.

As mentioned, this study only makes use of the data collection obtained by the "Uyanik" vehicle. The methodology used in Istanbul for acquiring the data allows for more precise "real-world" data compared to simulated environment studies [1, 16], where the lack of surrounding influences, such as: traffic, result in worse implementations. But the drawbacks of live human data collection, such as, equipment failures, technical difficulties, climate and weather, together with traffic, driving local culture and effectiveness of law enforcement, give rise to major challenges. Specially, the driver's physical and mental conditions along with the driving behaviors, which are the focus in this work.

The data collection performed in Istanbul consisted of a 25 km route formed by different kinds of sections (city, highway, secondary roads, and a university campus) accounting for 40 minutes for each run. A total of 108 drivers consisting of 19 females, from the ages between 21-48, and 89 males, from the ages between 22-61, drove the vehicle through the entire route. But due to equipment malfunction only samples from 101 drivers are usable.

The data from the 101 drivers was received through several channels. First, three cameras were responsible for recording 640×480 30 fps uncompressed digital video tape from the left and right sides of the driver and the road ahead. Second, four microphone channels and a sync signal between the two acquisition systems, which were sampled at

48 kHz and 24 bits per sample. Third, CAN-Bus signals like the vehicle speed (VS), the engine RPM (ERPM), the steering wheel angle (SWA). This data from the CAN-Bus cannot be read at a programmable rate, it fluctuates around, either 10 or 32 Hz. Therefore, in order to have uniformity, they were re-converted to 32 Hz when needed. Additionally, brake and gas pedal pressure sensors were installed, their readings as well as the GPS recordings were digitized at the CAN-Bus sampling rate. Finally, a laser distance measuring device, attached to an IMU XYZ Accelerator measuring sensor set-up, was installed in the front bumper. The IMU Accelerator is responsible of recording the values of the accelerations which are used for the evaluation of comfort variables defined in the second chapter. All the recordings required getting synchronized at the highest frequency of 32 Hz. The re-synchronization generates a set of easily processable spreadsheet-like data chunks.

All these variables constitute the *vehicle variables* and they can be further distinguished in two subgroups. On the one side, the vehicle *state variables* describe the disposition of the vehicle. On the other hand, the vehicle *handling variables* describe the actions taken by the drivers when operating the vehicle. The state variables are the acceleration for each axis, (Acc_x, Acc_x_pos, Acc_x_neg, Acc_y, Acc_z), the engine rpm (ERPM), the pitch rate (PR), the roll rate (RR), the yaw rate (YR), and the vehicle speed (VS). The handling variables are the the brake pedal (BP) and gas pedal (GP) pressure sensor signals, the percentage of the gas pedal pressure (PGP), the steering wheel angle (SWA) and the steering wheel relative speed (SWRS).

The route is divided in the aforementioned 4 groups depending on the type of road, the most adequate stretch for researching driver styles has been selected for implementing MLTs. In order to be suitable for this work, the route must be the same for every driver. Moreover, the data needs to be varied and properly distributed for the comfort variables to relate the comfort with the driving style and isolate the discomfort produced by the vehicle handling from other external factors.

Data from highway stretches, with fluent traffic and high mean speeds, shows low deviations and lack of variety, hardening the task of classifying comfort groups, since most of samples fall in the same comfort category. Thus, the idea of using the highway stretch is abandoned. City stretches are discarded since too many samples contain data where the vehicles are stopped in red lights or hard traffic and therefore represent “false” data.

Arguably, the best option and the one used in this work, is the data collected in the university stretch. Despite only accounting for 6 minutes on average of the entire 40 minute ride, and thus, provides a smaller amount of samples, it exhibits the most balanced variable distribution from all the stretches.

7 Data processing

The efficiency of machine learning techniques is highly dependent on the data used for training them. In this section the actions taken over the Uyanik dataset are described. The aim of this process is to make them suitable for implementation and to obtain maximal performance from the learning algorithms. Additionally, how the comfort levels are defined will be explicated.

The data used in this work belongs to 18 drivers, considered to perform the most diverse actions for the same university stretch. The ride has been divided in 15 second sample windows, with a total number of 933 samples. Each window has a 7.5 second overlap with the consecutive sample. For each sample window a set of 21 variables have been defined. 15 of the 21 variables are the vehicle variables described in Section 6. The root mean square (RMS) values, across the 15 seconds of the sample windows, have been calculated for each of these variables. The other 6 variables are the comfort variables in (4), calculated in each sample window for each axis. They are labeled as nax, nax_pos, nax_neg, nay, nay_pos, nay_neg; where _pos and _neg distinguish the use of only positive or negative accelerations respectively.

7.1 Data cleaning

As previously mentioned, only data from the university section has been taken due to its varied and properly distributed values for the variables. Nevertheless, measurement errors and special situations like bumps or dangerous driving might produce data non representative of a controlled environment.

The outlier cases are detected and the corresponding samples have been removed since they misrepresent the real data. To consider a sample as an outlier, the value of any measured vehicle parameter has to be 10% away form any other data from that parameter. In total, twelve samples have been removed reducing the number of usable samples from 933 to 921.

7.2 Normalization

The variables in the dataset represent unlike magnitudes and hold values of different ranges. Since the MLTs implemented in this work make use of Euclidean distances without caring of their nature, the disparities in range between the variables need to be taken into account. Not doing so would assume that some variables have more importance than others. The simplest and most efficient solution is to normalize each variable.

The normalization could be weighted as proposed in [43], giving slightly higher values to variables that are considered more important or significant. But without enough information to make a weighted normalization, the data for each variable has been evenly normalized between 0 and 1. The normalized distributions for the vehicle and comfort variables of the 921 samples is shown in Figure 6. It can be seen that the distributions are centred around a certain value and become more skewed towards the top.

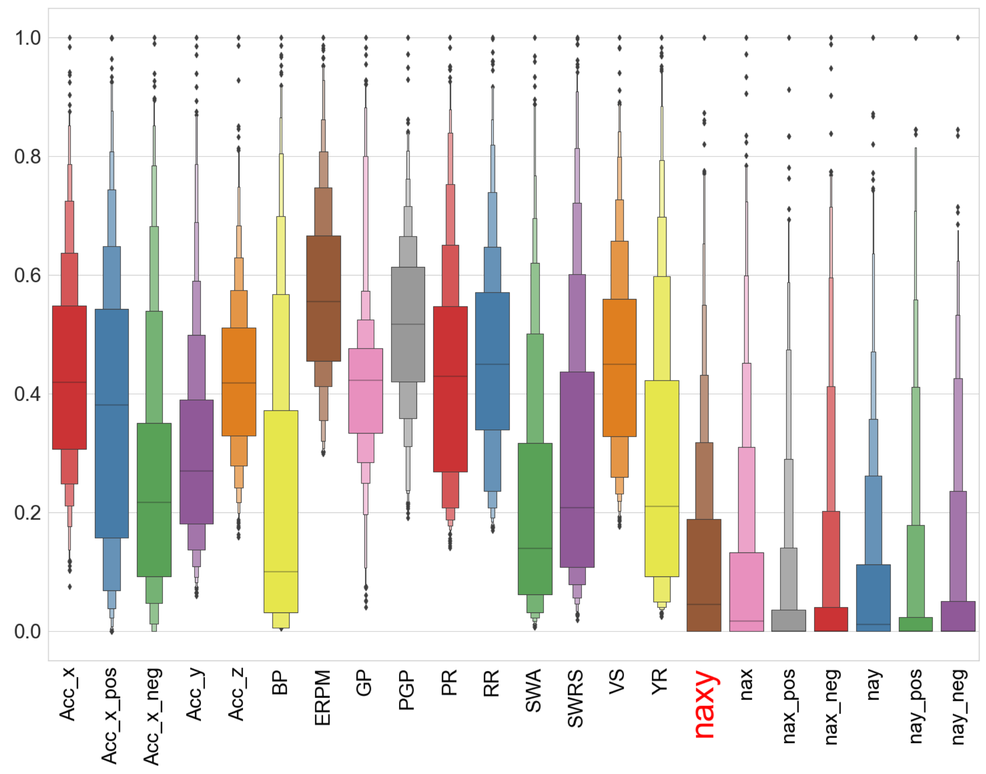


Figure 6: Normalized distributions of the vehicle variables and the comfort variables for the 921 samples.

7.3 Selection of comfort variable as evaluation Criteria

With the intent of researching the acceleration threshold variables from (4), a comfort variable has to be selected to classify the samples. As mentioned in Section 2, the accelerations in the z axis have been neglected. Thus, several ways to combine these variables in the x and y axes have been studied in order to get an unique comfort evaluation criteria.

The first idea was to use the nax and nay variables. They are the sum of the peaks counted on the positive and negative directions for each axis, that is, $nax = nax\_pos + nax\_neg$. A second idea consisted in using a new variable by subtracting the negative accelerations to the positive ones. Thus, we defined the variables for each sample window so $nax\_sub = nax\_pos - nax\_neg$ and $nay\_sub = nay\_pos - nay\_neg$. This variable was rapidly discarded since sustainable data was lost and input abundance is prime in high density data studies.

Nonetheless, studying the nax_sub variable revealed that 84.8% of the samples for nax and 86.32% of the nay are composed only by positive or negative values in each window. That is to say, in 84.8% of the samples $nax = nax\_pos$ or $nax = nax\_neg$, and 86.32% of the samples have $nay = nay\_pos$ or $nay = nay\_neg$. This observation hints the usage of the nax_pos and nax_neg variables separately, since each of them will be majorly represented in different windows and they are associated to unrelated vehicle variables. On the other hand, there is no need to split the nay variable since it is expected to be

Table 2: Discomfort categorization depending on the number of times the acceleration threshold was surpassed for the composed (*naxy*) variable.

Discomfort level		**naxy**
Not Uncomfortable	$n_i \in$	0 - 21
A little Uncomfortable	$n_i \in$	22 - 50
Fairly Uncomfortable	$n_i \in$	51 - 105
Uncomfortable	$n_i \in$	106 - 203
Very Uncomfortable	$n_i \geq$	204

linked to the vehicle variables corresponding to the steering wheel and those do not have directional differentiation.

Therefore, the *nax_pos*, *nax_neg* and *nay* variables are selected to study, comprehend the results and provide adequate driving advice to the drivers. Nevertheless, classifying the comfortableness according to variables with different nature is complex, so having an absolute acceleration threshold variable becomes convenient. For this task, a new variable is created following the results in [26], the RMS of the n_i variables in the xy plane has been calculated for each sample window following the next equation:

$$naxy = \sqrt{(nax\_pos + nax\_neg)^2 + nay^2}. \tag{14}$$

This variable is called "criteria" through the rest of this work, since it is the criteria for which the comfort level of the samples will be classified.

7.4 Comfort level categorization

In studies conducted with buses where samples of 5 minutes are examined [17, 23]. It has been determined that the number of peaks in any direction required for an uncomfortable ride correspond to values over 1 or 2 [23], and a value of 2.3 peaks in [17].

Since cars are arguably more comfortable than buses due to suspension, grip and general stability, the threshold needs to be slightly higher. Since the samples used in this work correspond to 15 minute stretches, the selected threshold to classify the n_i stands at a value of 8 in any direction. Thus, windows with $n_i < 8$ are regarded as comfortable and $n_i \geq 8$ as uncomfortable. In this work, additional categorization of the uncomfortable class has been done by dividing the uncomfortable spectrum in different ranges according to the distribution of the variables. For the criteria, the limits are calculated according to (14) and [17, 23], and rounded up to be integers as shown in Table 2.

7.5 Oversampling

When trying to split the acceleration threshold variables, the problem of imbalanced data is confronted as it can be seen in Figure 6. Most of the samples display low values of

n_i, for instance, for the criteria, 58.8% of the samples have lower values than 21, which corresponds to roughly 10.3% of its maximum value. When MLTs, mainly unsupervised ones, are trained with imbalanced data their reliability decreases since they are unable to properly classify the samples [44]. In this case, for the *naxy* variable, the learner can achieve 58.8% accuracy by predicting every new sample as comfortable, but such a learner is useless since it can hardly distinguish uncomfortable classes.

To correct imbalanceness across the samples three major rescaling approaches are presented. The first approach is to perform *undersampling* on the major class (class with more samples), that is, some comfortable samples are selectively dropped so that the classes are balanced. The second approach is to perform *oversampling* on the minority class (class with less samples than the majority class), that is, to increase the number of uncomfortable samples. The third approach is *threshold-moving* so the dividing threshold divides the classes with equal number of samples in each of them. Since the threshold is already defined and moving it would not allow a proper comfort classification, so this last method is neglected.

Undersampling techniques like Near Miss [45] or Tomek´s Links [46] could be used to clean the data while balancing the classes. However, the data is heavily comfortable and would reduce drastically the number of samples, around 700 would be eliminated. It makes undersampling unreasonable because valuable information is most probably lost. Therefore, oversampling results the most sensible option.

Oversampling is not simply duplicating existing samples and it was already discussed discarding duplicates; otherwise, serious overfitting happens. Oversampling methods generate synthetic samples by interpolating neighborhood samples of the minority class. In this work, three different oversampling methods are applied, the Synthetic Minority Oversampling Technique (SMOTE), the Adaptive Synthetic Sampling (ADASYN) and the Support Vector Machine SMOTE (SVMSMOTE).

SMOTE creates synthetic samples of the minority class by interpolating a point at a random distance between two sample vectors [47]. To do so, a random minority sample $\vec{s}_1$ is chosen, then a neighbour sample $\vec{s}_2$ from the same class is selected, and the new sample vector $\vec{v}$ is created following the Equation (15), where x is a random number between 0 and 1.

$$\vec{v} = \vec{s}_1 + x \cdot (\vec{s}_1 - \vec{s}_2) \tag{15}$$

ADASYN builds on the methodology of SMOTE, by shifting the importance of the classification boundary. Instead of using a random value of x in Equation (15), ADASYN uses a weighted distribution to compensate for the skewed distributions of minority examples [48]. ADASYN also uses a weighted distribution for different minority classes according to their level of difficulty in learning, so more synthetic data is generated for minority class examples that are harder to learn.

SVMSMOTE is trained to predict future instances. It focuses only on the minority class instances residing along the decision boundary, due to the fact that this region is the most crucial for establishing the decision boundary [49]. Furthermore, the new

Table 3: Discomfort categorization depending on the number of times the acceleration threshold was surpassed for the longitudinal (*nax_pos and nax_neg*), transversal (*nay*) and composed (*naxy*) variable.

Discomfort level		**nax_pos**	**nax_neg**	**nay**	**naxy**
Not Uncomfortable	$n_i \in$	0 - 7	0 - 7	0 - 15	0 - 21
A little Uncomfortable	$n_i \in$	8 - 17	8 - 17	16 - 35	22 - 50
Fairly Uncomfortable	$n_i \in$	18 - 41	18 - 41	36 - 83	51 - 105
Uncomfortable	$n_i \in$	42 - 71	42 - 71	84 - 143	106 - 203
Very Uncomfortable	$n_i \geq$	72	72	144	204

minority samples are generated outside the current minority region by extrapolation and the boundary is consolidated by interpolation. This way data is expanded outside the limits, which could be beneficial.

In a first study, the *naxy* variable was selected to execute the oversampling on, since it is the absolute categorization variable, as it has been explained in Section 7.1. Since, the oversampling techniques do not take into account (14), oversampling over *naxy* resulted in incoherent samples, i.e. samples containing high values of *naxy* but low values of *nax_pos*, *nax_neg* and *nay* which did not fulfill (14). This made classifying the samples impossible since the criteria could classify one sample as uncomfortable but all the rest of n_i parameters classified it as comfortable. In addition it creates struggles when trying to identify driving patterns correlated to the specific handling variables. Therefore, the idea of oversampling the *naxy* variable was rejected.

In order to avoid the previous issue, and after trying several solutions, three separate oversamplings have been executed on the original data. Each one of them over the *nax_pos*, *nax_neg* and *nay* variables. On this wise, these three variables contain the same amount of samples for each of the discomfort intervals. However, the equality of distribution on the *naxy* variable is compromised. After the oversampling, *naxy* is computed and therefore, an uniform distribution along *naxy* can not be assured. The oversampling intervals are chosen to fulfill the categorization in Table 3.

After oversampling the selected three parameters and recalculating *naxy*, the three methods result in similar distributions, see Table 4. Therefore, a better understanding of the quality of each method is required. To do so, the “variance” respect to the ideal distribution (equal in every comfort class) is calculated such as:

$$variance = \frac{\sum_{i=0}^{4}(n_i - N/5)^2}{N} \tag{16}$$

where N is the total amount of samples and n_i is the number of samples in the comfort class i ($i = 0$: not uncomfortable, and $i = 4$: very uncomfortable). The smaller the variance the better, since represents a better distribution of the samples along the discomfort classes.

Table 4 shows the different distributions, plus the total number of samples and the variance of each one of the methods. The data obtained by oversampling with ADASYN

Table 4: Number of samples in each discomfort category of *naxy*, after executing the SMOTE, ADASYN and SVMSMOTE oversampling techniques. The total number of sample for each method and the variance respect to an uniform distribution.

Discomfort level	naxy	**SMOTE**	**ADASYN**	**SVMSMOTE**
Not Uncomfortable	0 - 21	1609	1020	889
A little Uncomfortable	22 - 50	1816	1948	1339
Fairly Uncomfortable	51 - 105	1730	1840	1240
Uncomfortable	106 - 203	1739	2175	1758
Very Uncomfortable	≥204	953	890	1113
Total samples		7840	7873	6339
Variance		63.1	171	65.2

is discarded, since its variance is almost three times as big as the other two. SMOTE and SVMSMOTE present similar variances, but the samples are spread differently. SMOTE produces less samples in the very uncomfortable class, while SVMSMOTE, due to its working principle, creates samples with values over the previous maximum and thus producing more uncomfortable samples. Additionally, SVMSMOTE produces samples with negative values of *nax_pos*, *nax_neg* and *nay*, which makes no sense, since the times the acceleration threshold has been surpassed can not be negative, the minimum must be zero. Thus, for the SVMSMOTE the samples with negative values have been discarded, explaining the smaller amount of samples in the not uncomfortable class and in total for this method.

From the values obtained for the variance it is decided that the data obtained from the SMOTE and the SVMSMOTE will be used to train the MLT algorithms. Observing the different results emerging from the mirrored distributions for each method becomes interesting.

8 RBFNetwork implementation

The implementation of the RBFNN has been done following the explanation Section 4. As mentioned there, the number of centroids, their location and the initial radius of the distribution function have to be specified. As suggested in [35, 36], the location of the centroids were determined by using samples from the dataset and the radius of each centroid was initially set to be a third of the distance to the closest neighbour. This provides the algorithm with useful centroids from the beginning, since it occupies space where samples are likely to be located.

With these parameters set, a number of networks were trained with different amount of centroids, but showed overall low accuracy. To improve the results, several solutions were tried: the number of input variables was systematically reduced from 19 (all the vehicle variables) down to just 2 handling variables, the data from the three over-

Table 5: Accuracy and MSE of the RBFNetworks containing different amount of centroids and different number of input variables (19: all vehicle variables; 8: all handling variables; 2: BP and SWA).

No. input var.	No. centroids	Accuracy %	MSE %
19	500	15.4	86.5
	1000	17.8	82.2
	3000	16.7	78.1
8	500	32.9	75.3
	1000	34.4	74.2
	3000	33.8	72.6
2	500	51.9	72.3
	1000	56.3	68.4
	3000	55.6	70.1

sampling methods were implemented, the number of input samples was changed, different centroid arrangements were tried and the number of epochs of the training phase was increased.

For networks with different number of input variables, the accuracy and MSE of the best arrangements are tabulated in Table 5. It can be seen that the accuracy of the network is very low despite increasing the number of centroids. The best results, having only two input variables, only showed 56.3% accuracy, so almost half of the samples were not properly classified; and the value of the MSE is 68.4%, marking high deviation in prediction. Thus, the algorithm does not properly classify the samples and has very low confidence in its predictions.

By observing the final radii of the centroids and weights of the output layer, it was clear that the data from different classes is mixed in space. A specific value of the variables does not result in a direct discomfort type, since more than one class have similar values for different vehicle variables. Thus, the algorithm produced distribution functions with big radius that overlap, or with very small radius, which contain a minimal number of samples inside.

Apart from the data categorized in Section 7.4, RBFNetworks were trained with data oversampled in just two groups. One being the comfortable group with $naxy < 22$ and the other being the uncomfortable group when $naxy > 22$. Doing so brought up the accuracy to 93% and lowered down the MSE to 11.2%, even with all the 19 input variables. This means the data in the comfortable class is well defined and the data is mixed and overlaps across the uncomfortable categories. Moreover, the centroids of the comfortable class were located closest to the origin of the n dimensional space. Logically, making small maneuvering actions when driving produces small discomfort. These results are of no use to classify the samples, since the discomfort is not connected to specific handling variables.

All together, the results show the RBFNetwork is not appropriate to classify the data according to the categorization in Section 7.4, or that the categorization may not be ideal for a supervised learning technique. It could also be caused by the design of the criteria. Accelerations in different directions correlate to different handling parameters and some samples belonging to separate classes, but with the same axial origin, have overlapping data in some variables. For instance, samples with the similar break pedal action may have different comfort levels, caused by the accelerations in the y axis where the break has no influence.

With the RBFNN having a poor performance with the data oversampled in Section 7.5, a unsupervised machine learning techniques seems more appropriate to analyze the data, since it does not care about the categories the samples have been given. It would be able to create new classifications.

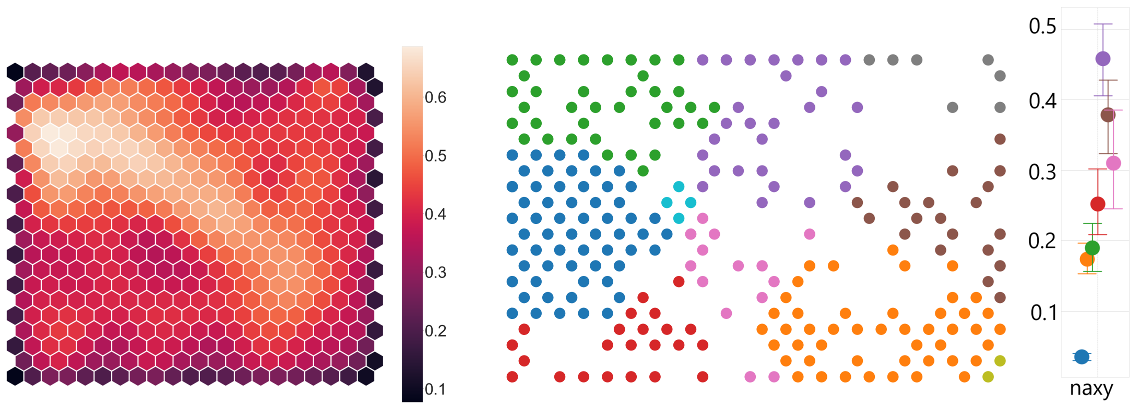
9 SOM implementation

Firstly, the use of hexagonal lattice map in this work must be addressed. Easy to implement, its maximization of neighbours produces a more effective learning process since more nodes are updated every cycle. Moreover, despite making the training 4 times slower, periodic boundary conditions are regarded as a must. Not considering PBCs resulted in poor maps, where the neurons on the edges have the lowest weight difference values, since they have less neighbours. In Figure 7a this issue is displayed. Apart from having the cluster centers in the edges, the middle ground of the grid is very uniform which is bad for cluster formation. Most of these clusters then, are just extensions of the edges, with high variance for the criteria, as seen on the right of the image. None of the initialisation methods proposed were able to significantly improve the quality of the SOM with PBCs off.

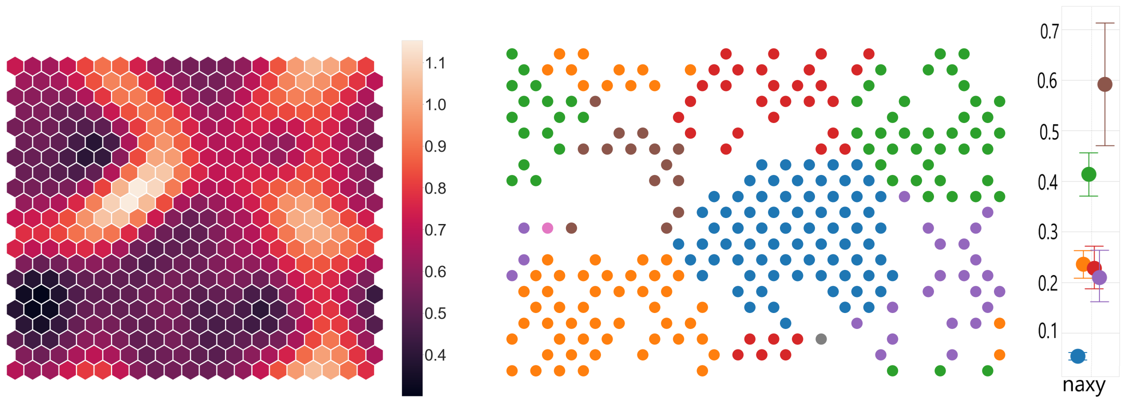
So with the PBCs on, different initialisation methods presented in the theory were implemented. Ordering according to the modules with linear and radial alignments produced close to none positive results. An example of a weight map obtained with vertical alignment is displayed in Figure 7b. It shows limited dark spots and a bland middle area, which produced overlapping for the *naxy* and vehicle variables. Using an auxiliary SOM was not sufficient either, to distinguish the cluster centers (see Figure 7c). Some cluster centers were too similar to each other and the final clusters join, resulting in high variance of the criteria. With several other methods showing even worse results, the solution, taken from [50], is to randomly initialise the map, but setting a large initial neighbourhood radius which reduces slowly. This is done by selecting a big initial σ in (13).

Using Vesanto´s rule, the initial topology of the map was set to 21 $\times$ 21 for the data obtained oversampling with SMOTE ($K = 7840$, $M \approx 443$), and 20 $\times$ 20 for the SVMSMOTE ($K = 6339$, $M \approx 398$). Apart from these dimension, a set of experiments were carried out with maps of 15 $\times$ 15 and 25 $\times$ 25. However, it was seen in the results from these last two failed to create cluster which properly classify the criteria.

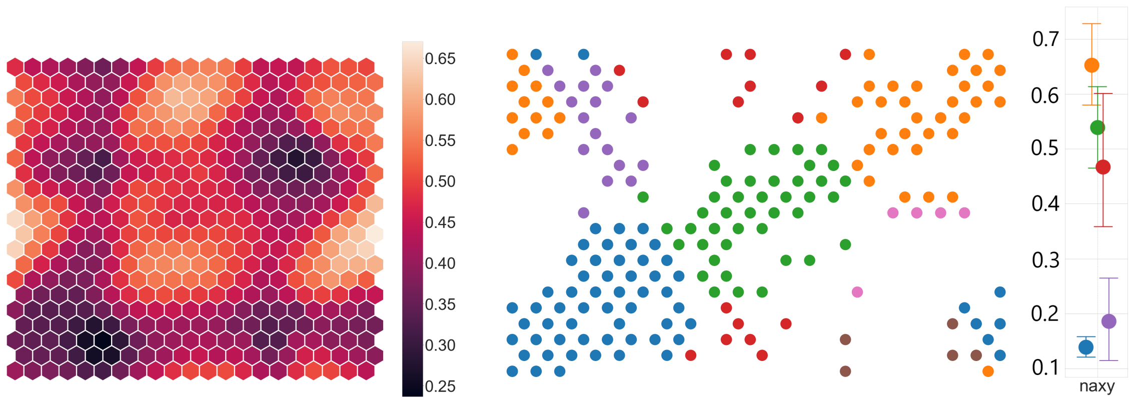
In the smaller 15 $\times$ 15 map, the lack of space produced just a few clusters with high variances. Figure 8a shows the formation of only 4 clusters, which resulted in big variance of the *naxy* variable across them.



(a) 21 x 21 SOM, without PBC.



(b) 21 x 21 SOM, with PBC, initialised by ordering the neurons depending on their modulus, in a vertical alignment.

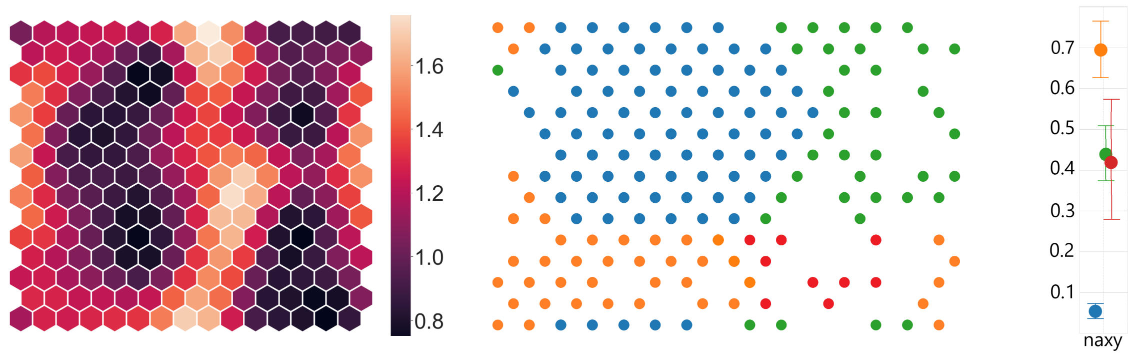


(c) 21 x 21 SOM, with PBC, initialised by fixing the centers formed by a smaller auxiliary SOM.

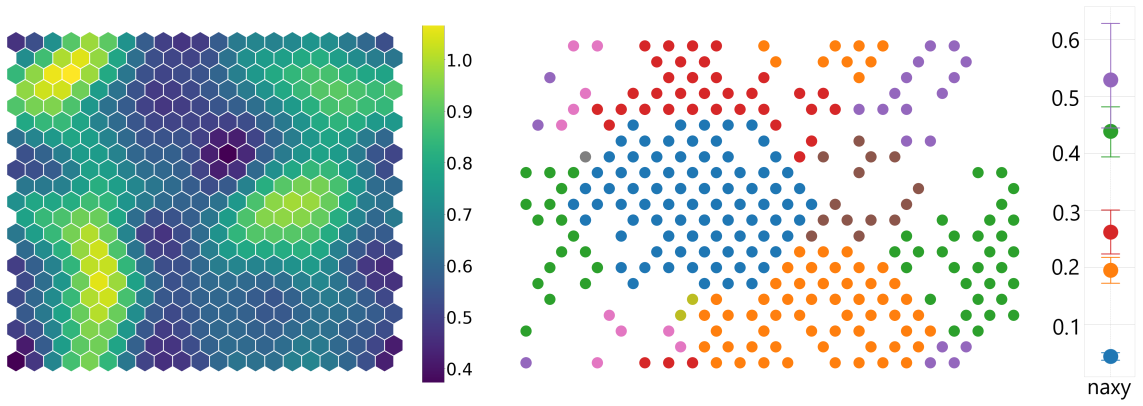
Figure 7: A series of visual representations of the SOMs created with initialisation methods. On the left, the U-matrix; on the middle the clusters formed when mapping the original data; and on the right, the mean and variance of the criteria for the clusters.

For the bigger 25 $\times$ 25 map, the abundance of space resulted in formation of separated clusters with identical values for the criteria. Neurons with similar weights are located in separated regions. In Figure 8c this can be observed, where a total of 11 clusters are formed. Some outliers are big enough to be meaningful, but are copies of other clusters, meaning that secondary clusters were formed with no additional information.

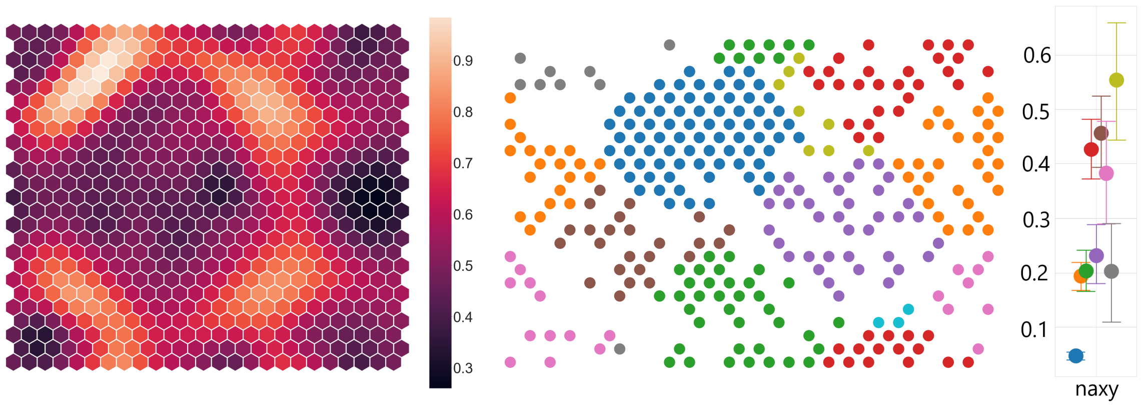
The 21 $\times$ 21 map size presented the best distinction for the criteria. It exhibits the smallest variance overall and minimal overlapping, as seen in Figure 8b. So, Vesanto´s rule was ensured and 21 × 21 maps were built and further trained to improve the classification capabilities.



(a) 15 x 15 SOM, with PBC, randomly initialised with big initial neighbourhood radius.



(b) 21 x 21 SOM, with PBC, randomly initialised with big initial neighbourhood radius.

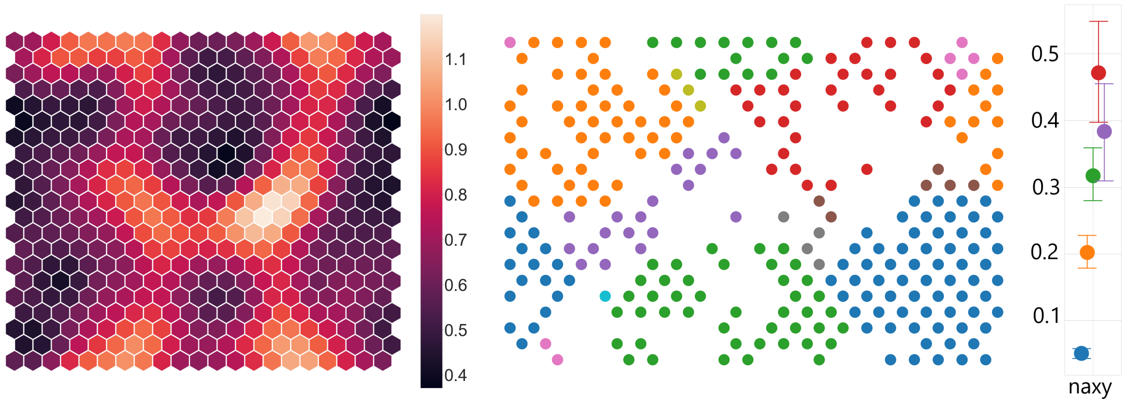


(c) 25 x 25 SOM, with PBC, randomly initialised with big initial neighbourhood radius.

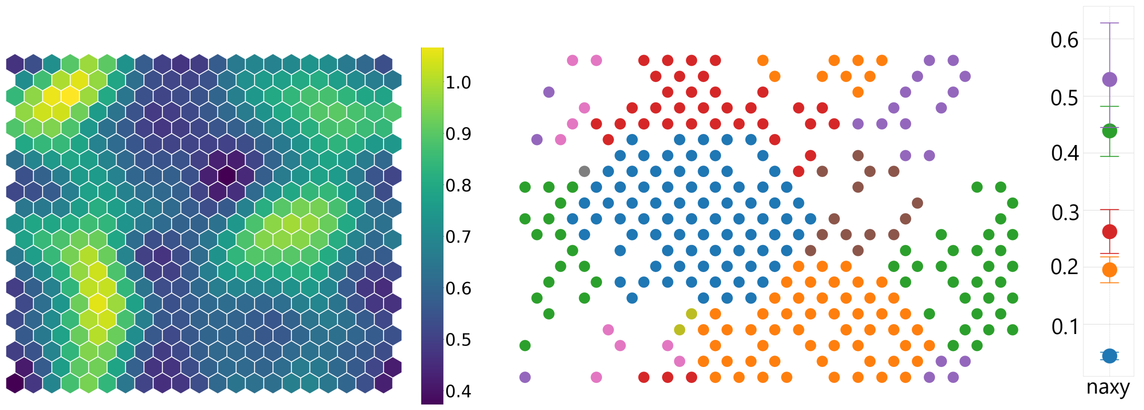
Figure 8: A series of visual representations of different sized SOMs. On the left, the U-matrix; on the middle the clusters formed when mapping the original data; and on the right, the mean and variance of the criteria for the clusters.

For training the map, initially, the entirety of the samples obtained in Section 7.5 were used. But the high amount of input samples resulted in an overload of data that produced bland U-matrices with little distinction between clusters and with many outliers. Therefore, a number of random samples from the entire oversampled dataset were used to train the SOMs. Setting the number of input samples at the same amount of windows in the original data (921), did not fix the issue. This amount is just a bit more than double the neurons in the map, and thus, not every neuron ended up being updated. By trial and error, the optimal number of input samples was set between 4 and 5 times the number of neurons in the map.

Finally, the results obtained by training using the data oversampled with SVMSMOTE, Figure 9a, and SMOTE, Figure 9b, were studied. To compare the data obtained by oversampling with SMOTE and SVMSMOTE the criteria, at the right of the images, needs to be compared, since both produce properly clustered maps. It can bee seen that SMOTE provides better differentiation for the comfort variable with smaller variance, and thus, it better classifies the samples. Therefore, the data oversampled with the SMOTE technique is considered superior in this work and the results and recommendations are analysed using the SOM trained with this data. The reason for the SVMSMOTE to be worse could be that it produced a more discomfort dataset and the original samples are majorly comfortable. The SMOTE on the other hand, with a lower distribution seems to better portray the original data.



(a) 21 x 21 SOM, with PBC, randomly initialised with big initial neighbourhood radius, and trained with the data oversampled using SVMSMOTE.



(b) 21 x 21 SOM, with PBC, randomly initialised with big initial neighbourhood radius, and trained with the data oversampled using SMOTE.

Figure 9: A series of visual representations of the SOMs created with different data when training. On the left, the U-matrix; on the middle the clusters formed when mapping the original data; and on the right, the mean and variance of the criteria for the clusters.

10 Cluster Analysis, Driving Recommendations and Driver Style Analysis

10.1 Cluster Analysis and Driving Recommendations

With the best conditions for the SOM, the results analysed in this section belong to a map of 21 x 21 dimensions, with PBCs and trained with 1764 samples of the SMOTE data set.

For every cluster obtained from mapping the original 921 samples into the SOM, the means and variances of every variable are computed. These values are then graphed in Figure 10. Displaying the *nax* and *nay* variables also allows to understand the origin of the discomfort and to better correlate it to the handling variables.

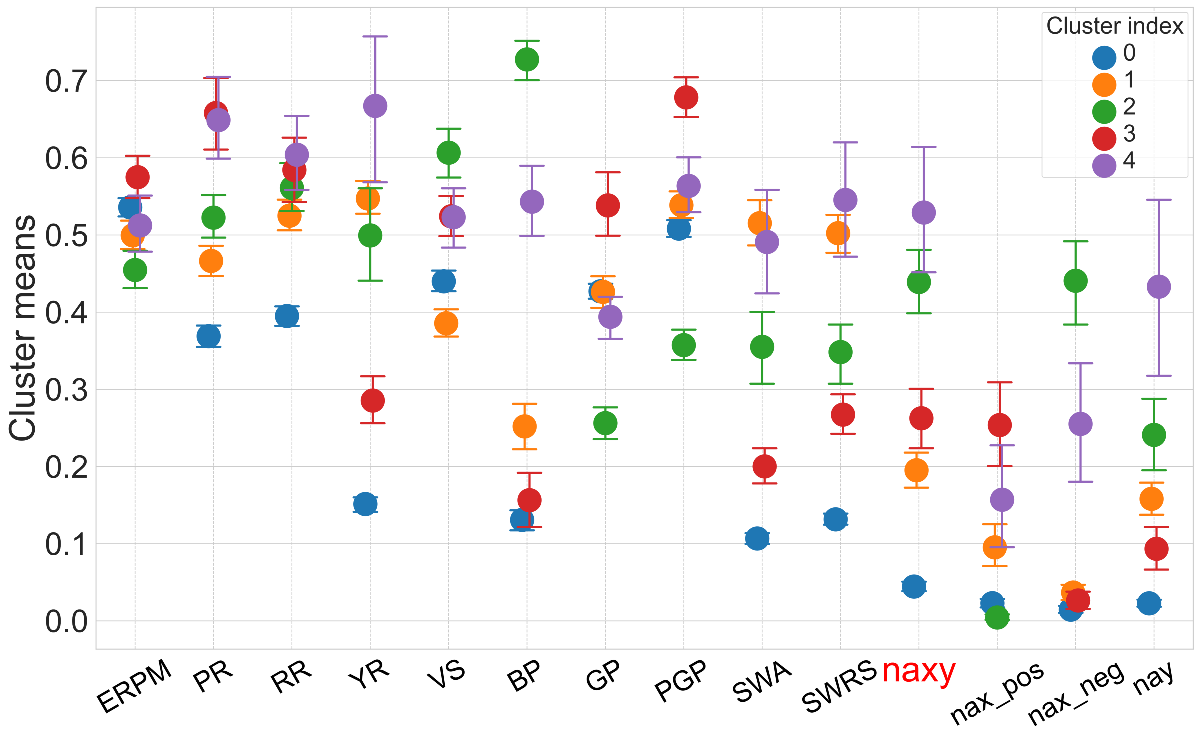


Figure 10: Mean values and variances of the clusters formed using a 21 x 21 SOM map with periodic boundary conditions and randomly initialised with big initial neighbourhood radius (SOM in Figure 9b).

The first 5 variables are the vehicle state variables. They characterize the conditions in which the vehicle handling variables are relevant. For instance, the cluster 1 has higher value for the BP than the 0 and 3 clusters while having the same *nax_neg* value, which could be due to the lower VS. The state variables are mostly grouped except of the YR, making harder to interpret. The ERPM variable will be discarded for the discussion since all of the clusters have arguably the same values. To the contrary, the vehicle handling variables are more spread out, helping produce more distinct driving recommendations.

Taken into account the absolute comfort variable *naxy* the clusters are categorized and then described according to the most meaningful vehicle variables for each one. Additionally suggestions to improve the ride experience for each one are the following:

0 - Blue. *Comfortable*: it scores the lowest value for almost all the variables. The medium values for GP and PGP do not translate in discomfort, due to the low YR and VS. The variables indicate the vehicle just keeps the speed and drives in a strait line. It is comfortable in every and therefore, no suggestion are needed for samples in this cluster.

1 - Orange. *A little Uncomfortable*: it is a bit discomforting in the y axis. With the smallest VS and high YR it is understood that it corresponds to slow tight curves having the highest SWA. It is also in the limit of being fairly uncomfortable in the positive direction of the x axis. This is reflected in a high GP and PGP values, similar to cluster 1, while having lower VS. Not much can be done to solve this, apart from moderating the use of the gas pedal.

2 - Green. *Fairly Uncomfortable*: it is in the limit of extremely discomforting in the x_neg direction and fairly uncomfortable in the y axis, the second most discomforting cluster. It scores highest in VS and lowest ERPM, which probably means high gears. Being in the limit of being fairly uncomfortable in the y direction, a small preventive suggestion would be to slightly lower the speed and keep the steering wheal as strait as possible. It scores significantly higher for the BP, and the lowest in gas pedal. This describes intense breaking which is suggested to be avoided by lowering the speed and being gentle when breaking.

3 - Red. *Fairly Uncomfortable*: an uncomfortable experience in the x_pos direction is felt in this cluster. However, it is not too high in the overall discomfort since it only scores in one axis. It has the highest ERPM, PR, GP and PGP. Together, with low values of YR, BP and steering wheel, denotes a straight acceleration window. It is strongly suggested to lower the excessive use of the throttle.

4 - Purple. *Uncomfortable*: it is the most discomforting cluster overall and discomforting in every direction. It has the highest values for almost every variable except VS and GP. The high value for nax_pos is not clear since the gas pedal usage is similar to 1 and 2 clusters, but it could be caused by the increased PR. However the elevated values for the nax_neg and nay are clearly depicted by the excessive maneuvering of the BP, SWA and SWRS, which are specially recommended to be decreased in order to minimize their discomfort effects.

10.1.1 General observations

Note from the picture of the results, that the variance of the criteria correlates with the variance of the rest of the variables, the bigger the variance of *naxy*, the bigger the variance of the other variables. Furthermore, the variances shrink as the comfort increases. This is most probably due to the data being more comfortably distributed, which indicates that having a better distributed experimental data set would enrich the map. Moreover, non of the variables reach the maximum value, in the very uncomfortable category, because those samples are left as outliers in the SOM, so getting rid of them at the start could also have been useful.

Despite having discarded the ERPM variable for the recommendations, it would be interesting to posses information about other variables, such as the gear box of the car, to better describe the state of the vehicle and have a clearer image of the road conditions.

The correlation between nax_neg and the BP is strong, marginally affected by the other variables. At first glance, the nay variable seems to correlate with the SWA and SWRS, but under some circumstances, depicted by the orange cluster, it does not fully match them. For the nax_pos, is logically related to the gas peal. However, nax_pos has a more direct connection, but not fully, with the PGP rather than with the GP. Also depends on the circumstances like the PR and VS. For this direction, knowledge of the gear box would also be beneficial, since the car has more torque at lower gears. This could also help explain the high values of the gas pedal variables, while the discomfort in the positive x direction is not that elevated. The need to have a minimum pressure to maintain speed could also raise the general GP and PGP values. They only drop when the break is heavily used, for obvious reasons.

Another interesting observation is the general lower value of the nax_pos variable compared to the nax_neg. This could be caused by the relative small horse power of the vehicle and its inability to produce sudden acceleration from the gas pedal signal. On the contrary, the breaks produce a more instantaneous and effective velocity decrease increasing the discomfort on the negative direction.

Finally, the strong accelerations in the x positive direction do not amount to much acceleration in other direction, indicating that drivers do not tend to strongly speed up during curves. The other two directions, x negative and y, display simultaneous high discomfort values, and drivers with those samples break while changing direction.

10.2 Driver Style analysis

The clusters obtained from the SOM are used in this section to analyse the driving style of each driver. This allows to compare the maneuvering of the vehicle and the discomfort produced respectively by each driver. In Figure 12, the percentage of samples belonging to each cluster are pictured for every driver. The distribution has been ordered from most comfortable on the left, to least comfortable on the right.

It can directly be seen that drivers are mostly comforting, having a heavy presence of comfortable samples. This is expected, since most of the data fit in the comfortable category, and for the same road drivers are required to take similar actions. However, the significant information relays in distribution of the uncomfortable samples. Using the knowledge gained in the previous section some distinctions are made between drivers.

IM1059 has the smallest percentage of the most discomforting cluster (green and

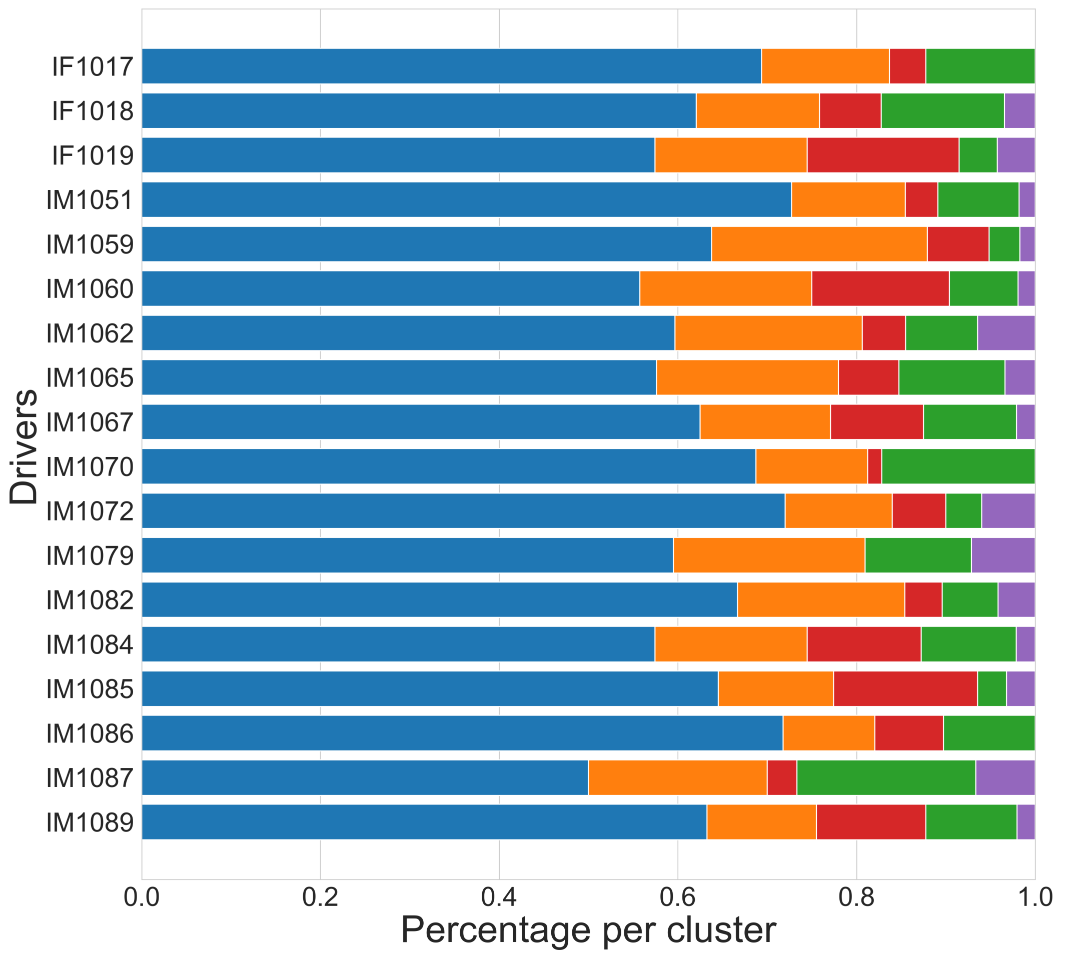


Figure 11: Distribution of clusters created with SOM for every driver. Cluster colors follow the color pattern in Figure 10.

purple together). With a reasonable red and the biggest percentage of orange, it can be considered one of the most comfortable drivers. He generally accelerates slowly and is careful when breaking.

IF1017 and IM1086 can also be the most comfortable drivers. They have some of the highest percentage of comfortable samples and no purple samples, which are the most discomforting. Not much can be recommended for these drivers, IM1017 could press softer the break pedal and IM1086 could reduce the gas pedal pressure.

IM1070 also does not have any purple sample, but has one of the highest percentages of green samples. Shows a heavier breaking pattern than the previous two.

IF1019, IM1060 and IM1085 have the highest percentage of red samples. They also produce little samples of the green and purple categories. Thus, the only suggestion for them is to control the pressure of the gas pedal.

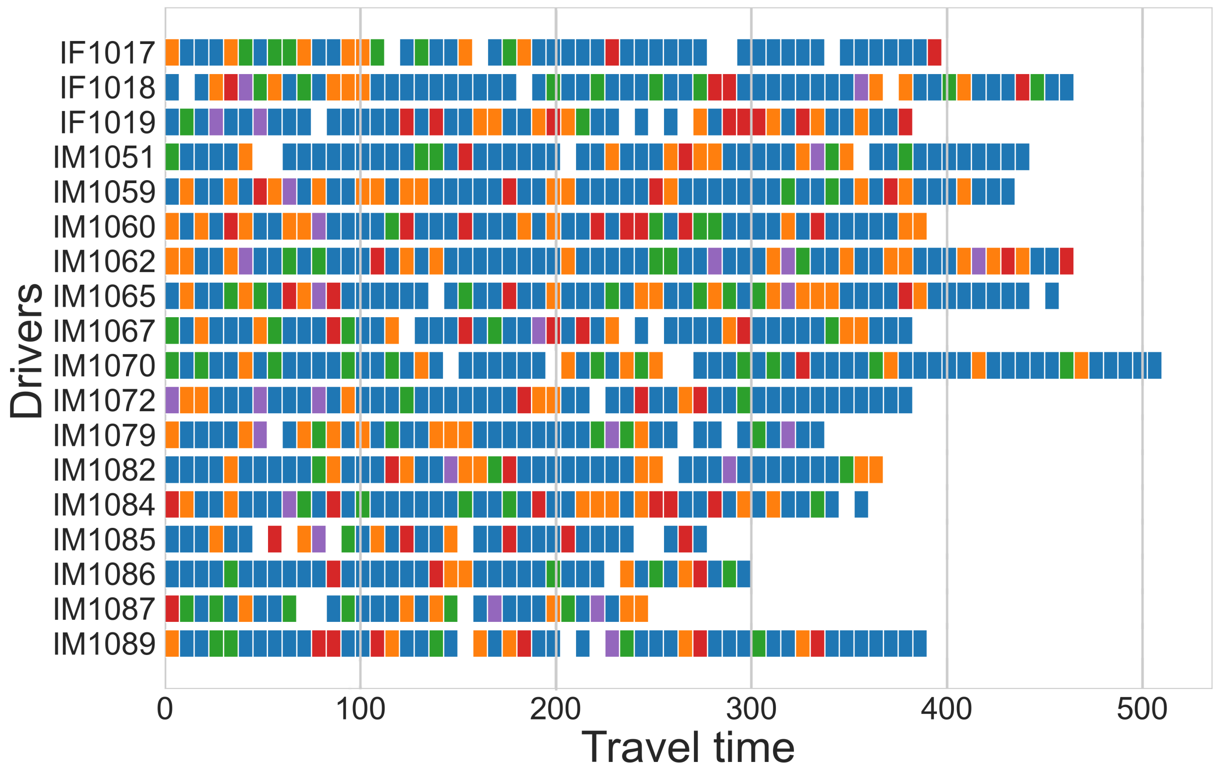


Figure 12: Cluster assignment for every sample of the drivers and ordered in time. The color of the samples follows the color pattern of the clusters in Figure 10.

Finally, it is obvious that IM1087 is the most discomforting driver. Despite half of the samples belong to the comfortable cluster, he occasions the smallest percentage of blue samples. Moreover, the green and purple bar combination is significantly bigger for him. Therefore, he does not accelerate strongly, but abuses the breaks and steering well. This shows a tendency for driving fast into corners to then correct the velocity with the break.

The rest of the drivers display some combination of these behaviours. Apart of the driver styles, it is also interesting to observe when and how the discomforts are produced. To do so, in Figure 12 every sample of each driver has been pictured, with the correspondent cluster color, in respect to the time of measurement. In the image the time of each sample is considered to be 7.5 seconds to account for the overlapping. The white spots symbolize the outlier samples, either from the data processing, or from the SOM.

No specific pattern seems to appear for the discomfort depending on the time of the ride. This brings to light the small influence external factor have on comfort for this data. However, some interesting points can be extracted from this picture by observing the arrangement of the differently clustered samples. The blue samples are not discussed since they are comfortable steady drives through straits, and the rest of the colors are studied on top of what can be considered as a blue background.

The orange samples generally appear in groups or paired with red ones. When they are group it indicates the slow acceleration that elongates in time, and when paired with red, it means the steady acceleration proceeds, or is proceeded by, a harder use of the gas pedal. The red samples mostly appear alone when not paired with a orange one. Therefore, the high accelerations from strong throttling are momentaneous and do not last much. The same happens with the green and purple samples, specially with the last one, meaning the breaking and trajectory changing occur shortly. The individual purple samples hint the lack of big curves on the road. The reason for two red or green samples being together, could be that the discomforting accelerations mostly happen during the overlapped time of consecutive samples.

For some of the drivers a few things can also be pointed out. IF1019 breaks at the beginning, probably slowing much the vehicle, and then speeds up the most through the rest of the stretch. IM1070 also breaks heavily and his ride lasts the longest, since he travels at lower speeds without excessive use of the throttle. IM1078, who was previously considered as the most discomforting, displays the shortest travel time, meaning he had the highest speed of any driver, and therefore needed to break and steer more.

Analyzing the driving styles and road behaviours in these ways can help to generate specific recommendations for each driver. Moreover, the travel analysis could improve the general comfort of the ride when encountered with the same road stretch.

11 Conclusions

The main objective of this work has been fulfilled. The general comfort of passengers in cars has been studied using machine learning techniques. Specifically, the discomfort generated by acceleration peaks using acceleration threshold theory. Moreover, the secondary goals of distinguishing the origin generating the discomfort and creating recommendations according to the driving style, have been successfully achieved. Despite meaning to use several MLTs, only two were implemented. The Radial Distribution Function turned out ineffective, but the Self-Organized Map was able to fix the issues the previous one encountered, enabling a rich analysis of the data and the creation recommendations for the different driving styles. In what follows, an overview of the research, the problems faced and instructions for further works are presented.

The data taken from the Uyanik database was first preprocessed. 15 second sample windows were created to be studied. The RMS of the vehicle variables and the discomfort variables were calculated for each window. Then, the comfort variable $naxy$, which encapsulates the discomfort in the x and y axes, was selected to proceed with the research and 5 discomfort classes were defined for it, ranging from not uncomfortable to very uncomfortable. Since the discomfort variable presented very bad distribution, synthetic samples were produced applying three oversampling algorithms, for a greater performance of the MLTs.

The results of the RBFNetwork showed the algorithm was inadequate to classify the given data in the 5 comfort categories. A deeper analysis revealed the data across the uncomfortable classes is scattered and mixed, while the data from the most comfortable class is well defined. The reasons for this could be the following: the discomfort categories are not properly defined and samples are wrongly classified; the number of samples in the original data which belong to the comfortable class is comparably bigger than for the uncomfortable classes, and thus, after oversampling most of the samples in the uncomfortable classes are synthetically produced and do not represent experimental data; or selecting the *naxy* as the criteria, with the intend to take into account multiple directional accelerations, may not be adequate.

The SOM on the other hand, due to its unsupervised nature, is less affected by the problems the RBFNetwork faces. Despite both techniques making use of Euclidean distances, the ability of the SOM to map the samples in an abstract grid, made it possible to distinguish several discomfort groups. The vehicle variables were then differentiated according to the clusters, which made possible analyzing the variables causing the discomfort. Recommendations for each cluster were derived from these results and the driving style of each driver was analyzed.

The next comparison can be made between both techniques. The neurons in the competitive layer of the SOM correspond to centroids of the RBF. During the training phase, when a neuron of the SOM is updated, the location of the analog centroid is displaced along with its neighbours towards an optimum position. This way, all the SOM centroids in one area belong to the same cluster. When new samples are mapped, the best matching unit assigns the corresponding group, regardless of their previous categorization.

The results from both the RBF and the SOM share one characteristics. The most comfortable samples have generally the smallest value for the vehicle variables and samples on the discomfort range overlap. This creates problems for supervised techniques, as seen with the RBF.

In order to solve this issues, the next solutions are proposed. A better categorization of the discomfort levels, more related to the sensation felt. The variances of the criteria produced by the SOM can be taken as a starting point. Too many samples lay in the comfortable class, so gathering more experimental data, with uncomfortable characteristics, would be the best option. This would allow to mark a better distinction of the uncomfortable classes and generate more realistic synthetic samples by oversampling.

Another interesting alternative is selecting samples of shorter duration. This could make the RMS of the variables more precise. Additionally, a driver analysis could be done using the vehicle speed and calculating the distance travelled in each sample. This way, road architecture like curves could be identified by comparing the driving patterns at different points of the trip.

Suggestions for future MLT implementations are Decision Trees or Feature Selection - Extraction algorithms. The Decision Tree is a interesting supervised technique that could classify samples by dividing the data through a chain of selections. It could emphasize more in some variables, in contrast to the normalized Euclidean distance calculations use done this work. Among semi-supervised methods, Feature Selection or Feature Extraction could be useful to extract valuable relationships between certain vehicle variables and specific discomfort parameters. Studying the discomfort variables separately, instead of using a composed variables like *naxy*, also seems compelling for these techniques.

It is clear in any case, that machine learning techniques are powerful tools for analyzing complex data. Obtaining reliable algorithms to predict or detect discomforting driving pattern could be integrated in the future.

References

[1] M. C. G. da Silva, "Measurements of comfort in vehicles," *Meas. Sci. Technol.*, vol. 13, no. 6, pp. R41–R60, apr 2002.

[2] M. J. Griffin, "Minimum health and safety requirements for workers exposed to hand-transmitted vibration and whole-body vibration in the european unio," *Occup. Environ. Med.*, vol. 61(5), p. 387–397, 2004.

[3] J. Garcia Solano, "Generation of ride comfort index," Ph.D. dissertation, UPC, Escola Tècnica Superior d'Enginyeria Industrial de Barcelona, Departament de Projectes d'Enginyeria, Jun 2014.

[4] WHO Regional Office for Europe, "European status report on road safety: towards safer roads and healthier transport choices." *Transp. Res. Part C Emerg. Technol.*, 2009.

[5] S. Li, K. Li, R. Rajamani, and J. Wang, "Model predictive multi-objective vehicular adaptive cruise control," *IEEE Trans. Control Syst. Technol.*, vol. 19(3), p. 556–566, 2011.

[6] D. J. Fagnant and K. M. Kockelman, "The travel and environmental implications of shared autonomous vehicles, using agent-based model scenarios," *Transp. Res. Part C Emerg. Technol.*, vol. 40, pp. 1–13, 2014.

[7] S. Saltera, C. Dielsa, P. Herriottsb, S. Kanarachosa, and D. Thakea, "Motion sickness in automated vehicles with forward and rearward facing seating orientations," *Appl. Ergon.*, vol. 78, pp. 54–61, 2019.

[8] J. Reason and J. Brand, "Motion sickness." England: Academic Press: Oxford, 1975.

[9] M. Elbanhawi, M. Simic, and R. Jazar, "In the passenger seat: Investigating ride comfort measures in autonomous cars," *IEEE Intell. Transp. Syst. Mag.*, vol. 15, pp. 1939–1390, 2015.

[10] L. Svensson and J. C. Eriksson, "Tuning for ride quality in autonomous vehicle : Application to linear quadratic path planning algorithm," 2015.

[11] Bruel & Kjaer, "Frequency response of the human body", in Human vibration. Naerum, Denmark: Bruel & Kjaer, 1999.

[12] ISO 2631, "Evaluation of the human exposure to whole-body vibration." Geneva: International Standard Organization, 1997.

[13] J. F. Golding, A. Mueller, and M. A. Gresty, "A motion sickness maximum around the 0.2 hz frequency range of horizontal translational oscillation." *Aviat. Space Environ. Med.*, vol. 72(3), pp. 188–192, 2001.

[14] H. Abut, H. Erdoğan, A. Erçil, B. Çürüklü, H. C. Koman, F. Taş, A. Ö. Argunşah, S. Coşar, B. Akan, H. Karabalkan, E. Çökelek, R. Fıçıcı, V. Sezer, S. Danış, M. Karaca, M. Abbak, M. G. Uzunbaş, K. Eritmen, M. Imamoğlu, and Ç. Karabat, "Real-world data collection with "uyanik"," in *In-Vehicle Corpus and Signal Processing for Driver Behavior.* Boston, MA: Springer US, 2009, pp. 23–43.

[15] O. Mata-Carballeira, I. del Campo, and E. Asua, "An eco-driving approach for ride comfort improvement," *IET Intell. Transp. Syst.*, vol. 16(2), p. 186–205, 2022.

[16] B. Aykent, F. Merienne, C. Guillet, D. Paillot, and A. Kemeny., "Motion sickness evaluation and comparison for a static driving simulator and a dynamic driving simulator." *J. Automot. Eng. Int.*, vol. Proc. IMechE, Part D, pp. 1–12, 2014.

[17] J. C. Castellanos and F. Fruett, "Embedded system to evaluate the passenger comfort in public transportation based on dynamical vehicle behavior with user's feedback," *Measurement*, vol. 47, pp. 442–451, 2014.

[18] L. A and G. MJ., "Motion sickness and motion characteristics of vessels at sea," *Ergonomics*, vol. 31, pp. 1373–94, 1988.

[19] A. Lawther and M. J. Griffin, "Prediction of the incidence of motion sickness from the magnitude, frequency, and duration of vertical oscillation." *J. Acoust. Soc. Am.*, vol. 82, pp. 957–966, 1987.

[20] B. E. Donohew and M. J. Griffin, "Motion sickness: Effect of the frequency of lateral oscillation," *Aviat. Space Environ. Med.*, vol. 75(8), pp. 649–656, 2004.

[21] M. J. Griffin and M. M. Newman, "Visual field effects on motion sickness in cars." *Aviat. Space Environ. Med.*, vol. 475(9), pp. 739–748, 2004.

[22] T. M and G. MJ., "Motion sickness in public road transport: The effect of driver, route and vehicle." *Ergonomics*, vol. 42, pp. 1646–64, 1999.

[23] L. Eboli, G. Mazzulla, and G. Pungillo, "Measuring bus comfort levels by using acceleration instantaneous values," *Transp. Res. Proc.*, vol. 18, pp. 27–34, 2016.

[24] C. Sohn, J. Andert, and D. Jolovic, "An analysis of the tradeoff between fuel consumption and ride comfort for the pulse and glide driving strategy," *IEEE Trans. Veh. Technol.*, vol. 69, no. 7, pp. 7223–7233, 2020.

[25] T. Nguyen, N. NguyenDinh, B. Lechner, and Y. D. Wong, "Insight into the lateral ride discomfort thresholds of young-adult bus passengers at multiple postures: Case of singapore," *Case Stud. Transp. Policy*, vol. 7, no. 3, pp. 617–627, 2019.

[26] B. Barabino, L. Eboli, G. Mazzulla, S. Mozzoni, R. Murru, and G. Pungillo, "An innovative methodology to define the bus comfort level," *Transp. Res. Proc.*, vol. 41, pp. 461–470, 2019.

[27] Z. Cui, X. Xia, and X. Pei, "A modified vehicle following control system on the curved road based on model predictive control." *IEEE*, p. 1098–1103, 2021.

[28] M. W. Berry, A. H. Mohamed, and Y. B. Wah, "A systematic review on supervised and unsupervised machine learning algorithms for data science," in *Supervised and Unsupervised Learning for Data Science*. Cham: Springer International Publishing, 2020.

[29] N. M. Allinson, "Self-organising neural maps and their applications," in *Theory and Applications of Neural Networks*, J. G. Taylor and C. L. T. Mannion, Eds. London: Springer London, 1992, pp. 101–118.

[30] T. Kohonen, "The self-organizing map," *Proc. IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.

[31] A. Hajian and P. Styles, "Artificial neural networks, in application of soft computing and intelligent methods in geophysics," in *Application of Soft Computing and Intelligent Methods in Geophysics*. Cham: Springer International Publishing, 2018, pp. 3–69.

[32] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A learning algorithm for boltzmann machines," *Cogn. Sci.*, vol. 9, no. 1, pp. 147–169, 1985.

[33] C.-Y. Liou, W.-C. Cheng, J.-W. Liou, and D.-R. Liou, "Autoencoder for words," *Neurocomputing*, vol. 139, pp. 84–96, 2014.

[34] D. Broomhead and D. Lowe, "Radial basis functions, multi-variable functional interpolation and adaptive networks," *Royal Signals And Radar Establishment Malvern*, vol. RSRE-MEMO-4148, 03 1988.

[35] "Three learning phases for radial-basis-function networks," *Neural Networks*, vol. 14, no. 4, pp. 439–458, 2001.

[36] H. Faris, I. Aljarah, and S. Mirjalili, "Chapter 28 - evolving radial basis function networks using moth–flame optimizer," in *Handbook of Neural Computation*, P. Samui, S. Sekhar, and V. E. Balas, Eds. Academic Press, 2017, pp. 537–550.

[37] I. Czarnowski and P. Jedrzejowicz, "An approach to rbf initialization with feature selection," *Adv. Intell. Syst. Comput*, vol. 322, pp. 671–682, 01 2015.

[38] J. Vesanto, J. Himberg, E. Alhoniemi, and J. Parhankangas, "Self-organizing map in matlab: the som toolbox," in *In Proceedings of the Matlab DSP Conference*, 1999, pp. 35–40.

[39] M. Johnsson, "Applications of selforganized maps." Rijeka, Croatia: InTechg, 2020.

[40] S. Kaski and T. Kohonen, "Exploratory data analysis by the self-organizing map: Structures of welfare and poverty in the world," 09 1996.

[41] C. Miyajima, T. Kusakawa, T. Nishino, N. Kitaoka, K. Itou, and K. Takeda, "On-going data collection of driving behavior signals," 2009, pp. 45–54.

[42] P. Angkititrakul, J. H. L. Hansen, S. Choi, T. Creek, J. Hayes, J. Kim, D. Kwak, L. T. Noecker, and A. Phan, "Utdrive: The smart vehicle project." Boston, MA: Springer US, pp. 55–67.

[43] Q. Song and N. Kasabov, “Wdn-rbf: weighted data normalization for radial basic function type neural networks,” in *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No.04CH37541)*, vol. 3, 2004, pp. 2095–2098 vol.3.

[44] Z.-H. Zhou, “Linear models,” in *Machine Learning.* Singapore: Springer Singapore, 2008, pp. 55–57.

[45] J. Zhang and I. Mani, “KNN Approach to Unbalanced Data Distributions: A Case Study Involving Information Extraction,” in *Proceedings of the ICML’2003 Workshop on Learning from Imbalanced Datasets*, 2003.

[46] I. Tomek, “Two modifications of cnn,” *IEEE Trans. Syst. Man Cybern.: Syst.*, vol. SMC-6, no. 11, pp. 769–772, 1976.

[47] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer., “Smote: synthetic minority over-sampling technique,” *J. Artif. Intell. Res.*, p. 321–357, 2002.

[48] H. He, Y. Bai, E. A. Garcia, and S. Li, “Adasyn: Adaptive synthetic sampling approach for imbalanced learning,” in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, 2008, pp. 1322–1328.

[49] H. M. Nguyen, E. W. Cooper, and K. Kamei, “Borderline over-sampling for imbalanced data classification,” *Int. J. Knowl. Eng. Soft Data Paradigms*, vol. 3, pp. 4–21, 2011.

[50] R. Beale and T. Jackson, “Neural computing: An introduction.” GBR: IOP Publishing Ltd., 1990.