Feature Selection from Sensor Time Series for Learning from Demonstration

handed in
BACHELOR'S THESIS

cand. ing. Mohamed Ali Masmoudi

born on the 04.10.1998
living in:
Christoph-Probst-str.8
80805, Muenchen
Tel.: 017657675606

Human-centered Assistive Robotics
Technical University of Munich

Univ.-Prof. Dr.-Ing. Dongheui Lee

Supervisor:	M. Sc. Christoph Willibald
Start:	25.05.2021
Intermediate Report:	25.06.2021
Delivery:	07.09.2021

TECHNISCHE UNIVERSITÄT MÜNCHEN
Human-centered Assistive Robotics
UNIV.-PROF. DR.-ING. DONGHEUI LEE

25.05.2021

B A C H E L O R T H E S I S
for
Mohamed Ali Masmoudi
Student ID 03709390, Degree EI

Feature Selection from Sensor Time Series for Learning from Demonstration

Problem description:

Learning from Demonstration (LfD) is an approach that allows to intuitively transfer task knowledge from humans to robots by demonstrating the task at hand e.g. via teleoperation or kinesthetic teaching. The gathered sensor data during a task demonstration can be described by a large set of generic features, such as measured forces acting on the end effector or minimal distances of the robot towards objects or important landmarks. Feature selection (FS) is the process of determining a minimal subset of task relevant features by eliminating redundant or uninformative features from the initial set. Since the search space of many ML setups, like Reinforcement Learning, scales exponentially with the number of features, FS is highly important to achieve sufficient performance in real world robotic tasks that involve a high dimensional state-action and feature space. FS approaches can be divided into filter [1, 2] and wrapper methods [3], which are employed in a separate step prior to the actual learning algorithm or integrated into a learning algorithm, respectively. The focus of the thesis project is set on FS from unlabeled sensor time series, where the following tasks are to be conducted:

Tasks:

- Literature research on feature selection with focus on time series data
- Implementation of at least two different algorithms for FS from unlabeled sensor time series, employing e.g. the approaches of [1], [2] or [3]
- Evaluation of the performance of the implemented approaches regarding elimination of redundant and irrelevant features

Bibliography:

[1] Davide Bacciu. Unsupervised feature selection for sensor time-series in pervasive computing applications. *Neural Computing and Applications*, 27(5):1077–1091, 2016.
[2] Fabrizio Bonacina, Eric Stefan Miele, and Alessandro Corsini. Time series clustering: A complex network-based approach for feature selection in multi-sensor data. *Modelling*, 1(1):1–21, 2020.
[3] Ruohao Xu, Mengmeng Li, Zhongliang Yang, Lifang Yang, Kangjia Qiao, and Zhigang Shang. Dynamic feature selection algorithm based on q-learning mechanism. *Applied Intelligence*, pages 1–12, 2021.

Supervisor:	M. Sc. Christoph Willibald
Start:	25.05.2021
Intermediate Report:	XX.XX.2021
Delivery:	12.10.2021

(D. Lee)
Univ.-Professor

Abstract

Among machine learning researchers, it is common to ought to deal with data sets containing a tremendous amount of heterogeneous features. This infers in a great challenge since classical machine learning strategies are not able to effectively deal with such number of inputs. As a result, it is normal to apply a preprocessing step to decrease the dimensionality. Feature selection provides an effective way to solve the problem of high dimensional data analysis by removing redundant and task-irrelevant data. It is a very effective way to reduce computation time, space complexity, improve learning rate and achieving better results. This is quite useful for robotics tasks especially those involving selecting task-relevant features from sensor data collected from the environment. Clearly, excluding important features can limit the quality of the learning process. At the same time, including redundant or superfluous data can result to slower learning or weaker end performance. In this thesis, we will discuss and implement several frequently used approaches in the field of feature selection and apply them on multivariate time series data. Finally, every method's performance will be tested on a real world experiment.

Contents

Chapter 1

Introduction

In the field of machine learning, it is common to performe a preprocessing step to let off all redundant or irrelevant features and decrease the dimensionality of the issue at hand. This can be accomplished with the help of 2 general approaches: feature extraction and feature selection. Although we will not be dealing with feature extraction in this thesis, it is worth differentiating between the 2 techniques. Feature extraction reduces the feature set by creating new features through transformations and combinations of the original features. Whereas features selection preserves the original features and that are relevant to the learning task and removing all redundant or irrelevant ones. In [BCAB18] we can see the difference between the two.

(a) Feature extraction

Figure 1.1: Examples of feature extraction and feature selection systems [BCAB18]

Some good feature selection methods have already been implemented for some complex problems. However only few do consider the case of pervasive computing; it is a vision of an environment enriched by a distributed network of sensors and devices that collect data continuously. In other words, not many feature selection algorithms do consider time as a valuable parameter in the task-learning process. By providing an effective way to collect good quality data, a robot would then be capable of not only performing tasks in a computationally effective way, but can also learn in an almost autonomous way a specific task under several different situations and react correctly accordingly.

1.1 Objective and Contribution

Our aim is to analyze and implement different feature selection algorithms that enables the robot with no prior knowledge of the environment nor with human/expert intervention to select the most relevant and non redundant features for the task. When performing multiple demonstration from the user, the robot will be collecting a set of heterogeneous data coming from different sensors. The feature selection methods will then extract the main features of the task, which will be the inputs for the machine learning models allowing the robot to start the learning process. In contrast to state of the art feature selection methods, the approach of this thesis focuses on the data as a function of time and will consider some interesting properties like correlation, linear dependence and the idea of mapping time series in the network domain to speed-up the learning process, and enhance the machine learning model generalization properties.
Therefore, the feature selection methods we are interested in should:

- Reduce computational cost for sensor information processing
- Suppress redundant/irrelevant information depleting predictive Performance

Moreover, the applications specific requirements are:

- Timeseries data
- Heterogeneous sensor information

The primary goal of this thesis is to gain an understanding of the most recent and effective developments in the field of unsupervised and supervised feature selection and analyze these techniques in real world robotics applications. Various feature selection (FS) methods based on different metrics will be analyzed in terms of accuracy and efficiency and later be tested on data obtained from a user demonstration. The Results will then be evaluated and scored to find out what feature selection method is more suitable for this kind of experiments. Therefore, this thesis focuses on answering the following questions:

- What are the available feature selection techniques and how do they work?
- What are the advantages and disadvantages or each method in time-series analysis in general?
- Based on the different demonstrations of the same task with different setups, what is the best approach to select the most relevant subset of features in terms of redundancy/noise reduction ?

Chapter 2

State of the art in feature selection

As mentioned above, feature selection is the process of reducing the dimensionality of the present features by detecting the relevant features and discarding the irrelevant ones.
This plays an essential role such as improving learning performance, preventing overfitting, and reducing the computational costs. For example, the number of training examples needed for the basic nearest-k neighbor classification algorithm to reach a given accuracy grows exponentially with the number of irrelevant features. Usually, feature selection can be categorized according to different perspectives. The first one is the supervision perspective; according to the availability of supervision (such as class labels in classification problems), feature selection can be broadly classified as supervised, unsupervised and semi-supervised methods. We can also consider the different selection strategies as a second criterion of categorization. Feature selection can also be broadly classified as wrapper, filter and embedded methods. In subsection 2.1, we will discuss each method briefly and give the circumstances under which we can use each one.

2.1 Selection Strategy perspective

Feature selection methods can be divided regarding the relationship between a feature selection algorithm and the inductive learning method used to infer a model, into three major approaches:

2.1.1 Filter methods

The methods used in this approach are independent of the learning algorithm. In other words, they only rely on the characteristics of the data to evaluate the data's importance to the task (characteristics such as correlation). This makes them usually more computationally efficient than other methods like wrapper methods (section 2.1.2). However, due the lack of the learning algorithms guiding and filtering the

features, the results may not be optimal. These methods normally consist of two steps; in the first step, the features are ranked according to evaluation criteria which rank all of them individually based on their importance. In the second step, all least important ones are filtered out depending on a parameter k which specifies the number of the k most important features. For example, Entropy measure has been used as filter method for feature selection for clustering [DL97].

2.1.2 Wrapper methods

The methods used in this approach are dependent of the learning algorithm. They rely on the predictive performance of a predefined learning algorithm to evaluate the quality of selected features. In other words, it uses the method of classification itself to measure the importance of features set. Given a specific learning algorithm, a typical wrapper method performs two steps: (1) search for a subset of features; and (2) evaluate the selected features. It repeats (1) and (2) until some stopping criteria are satisfied. Feature set search component first generates a subset of features; then the learning algorithm acts as a black box to evaluate the quality of these features based on the learning performance. For example, the whole process works iteratively until such as the highest learning performance is achieved or the desired number of selected features is obtained. Then the feature subset that gives the highest learning performance is returned as the selected features. Unfortunately, a known issue of wrapper methods is that the search space for d features is 2^d, which can be impractical for problem with a large number of features. Therefore, different search strategies such as sequential search [HS99], hill-climbing search, best-first search [KJ97];[AMXS16], branch-and-bound search [NF77] and genetic algorithms [Gol89]are proposed to yield a local optimum learning performance. However, the search space is still extremely huge for high-dimensional datasets. As a result, wrapper methods are sel- dom used in practice.
The following figure represents the process of the wrapper approach explaining steps (1) and (2);

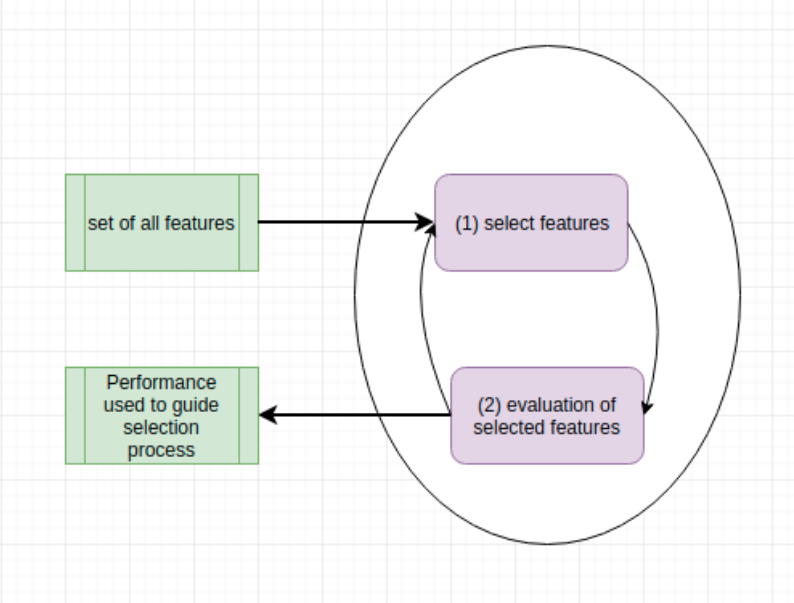


Figure 2.1: Wrapper approach for feature selection

2.1.3 Embedded methods

Embedded methods are a trade-off between filter and wrapper methods which embed the feature selection into model learning. Thus, they inherit the characteristics of wrapper and filter methods. (1) they include the interactions with the learning algorithm; and (2) they are far more efficient than the wrapper methods since they do not need to evaluate feature sets iteratively. The most widely used embedded methods are the regularization models which target to fit a learning model by minimizing the fitting errors and forcing feature coefficients to be small (or exact zero) simultaneously. Afterwards, both the regularization model and selected feature sets are returned as the results which are more robust and have a higher credibility in comparison to filters and wrapper methods. This technique searches for the most relevant and effective features for models. The most common embedded methods are regularization-based [EHJT04], including LASSO, elastic net, or ridge regression [YHQ+16].

2.2 Supervision perspective

According to the supervision criterion, feature selection can be classified into supervised, unsupervised and semi-supervised methods.

2.2.1 Supervised feature selection

First is the **Supervised feature selection**; Supervised feature selection is the process of selecting a feature subset based on some criteria for measuring the importance and relevance of the features by utilizing the labeled data to train the feature selection model. It aims feature selection aims to maximize classification accuracy. The first step is to split the data into training and testing sets, classifiers are then trained based on a subset of features selected by supervised feature selection. In the second step, the trained classifier tries to predict class labels or regression targets of unseen samples in the test set with the selected features.
This approach is explained in the the help of the following figure:

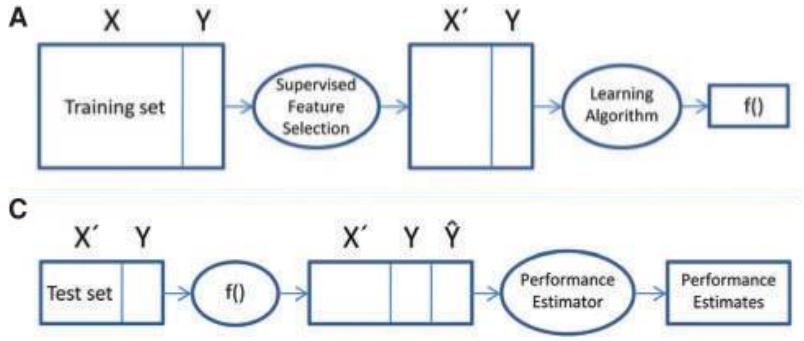


Figure 2.2: supervised feature selection process [SFK10]

The two steps together (A+C) consist the supervised feature selection approach; Preprocessing + Testing (evaluation of the built model).
In a preprocessing step (A), supervised feature selection reduces the set of features X to a subset X' (Y being the target attribute). Subsequently, the reduced training set is used to train a classifier f (). During testing (C), the trained classifier f () is evaluated using an independent test set with the feature space reduced to X according to the feature selection derived in the previous step. The classifier predicts $\hat{Y}$ for each instance. Various performance measures can then be calculated by comparing the predictions $\hat{Y}$ with the true values for Y [SFK10].

2.2.2 Unsupervised feature selection

Second is the **unsupervised feature selection**; this approach has the ability to identify and remove irrelevant and/or redundant features without needing a supervised dataset. In fact,the goal here is to identify for each feature a group (also known as cluster). Acquiring labeled data is particularly expensive in time, unsupervised feature selection may represent the better alternative for many complex tasks. When working with unlabeled data, unsupervised feature selection methods

seek alternative criteria to define feature relevance. In contrast to supervised feature selection, this approach is unbiased and performs well when no prior knowledge is available. They can also reduce the risk of data over-fitting when dealing with a new class of data [SFCOET19]. Unsupervised feature selection relies either on intrinsic properties of the data, such as the variance of features, similarity among features, consistency, entropy, etc. or the ability to o find good cluster structures in the data. A more formal definition of unsupervised learning can be found in [Kan05]- [Bro09]. It goes as follows:

Given a collection of m objects $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\}$, described by a set of n features $T = \{F_1, F_2, , \ldots, F_n\}$ possibly of different type (mixed data). Unsupervised feature selection consists in identifying a subset of features $T' \subseteq T$, without using class label information, such that T' does not contain irrelevant and/or redundant features, and good cluster structures in the data can be obtained or discovered.

To clarify this definition, we provide the following figure as of relevant and irrelevant features is shown.

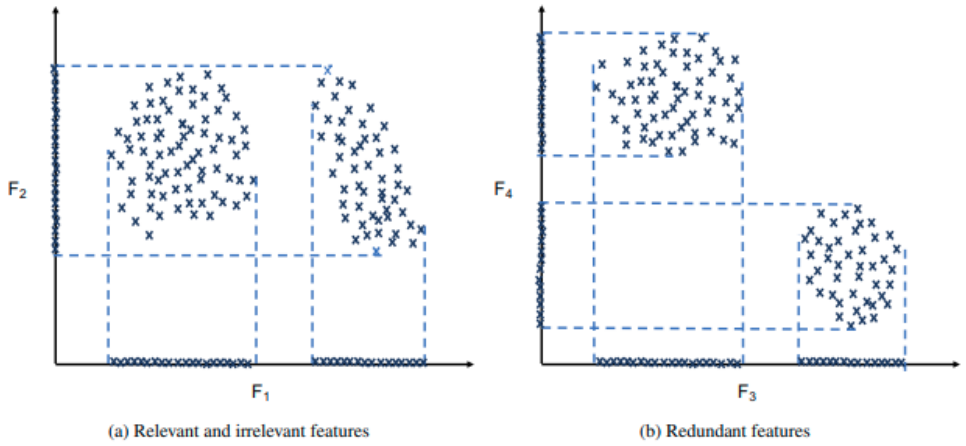


Figure 2.3: Relevant, irrelevant and redundant features[SFCOET19]

In this figure, we can see that F1 is a relevant feature because it can separate the data; as it can be seen when the data are projected to its respective axis. On the contrary, F2 is an example of an irrelevant feature because it by itself is unable to separate the data, as we can see in its projection.

In the case of unsupervised (clustering) tasks, this concept is closely linked to those features that reveal interesting and natural structures underlying the data.

On the other hand, a redundant feature refers to a feature that is relevant for discovering cluster structures in the data. But if it is removed from the data, it has not negative effect due to the existence of another feature (or set of features) that provides the same information. Redundant features unnecessarily increase the dimensionality, and therefore they can be removed.[SFCOET19]

2.2.3 Semi-supervised feature selection

At last, comes the **semi-supervised method**. This method is a trade-off between the two methods mentioned above. Thus, they inherit the merits of supervised and unsupervised feature selection methods. For many real-world applications, we have limited number of labeled data. In order to benefit from it, some algorithms have been developed to exploit both labeled and unlabeled data samples and take advantage of both supervised und unsupervised characteristics. Some new ways have been introduced to the semi-unsupervised feature selection approach such as the Laplacian score (LS). The basic idea of LS is to evaluate the features according to their locality preserving power. If two data points are close to each other, they belong to the same class with high probability. It is fundamentally based on Laplacian Eigenmaps [BN01] and Locality Preserving Projection [HN03] [YHQ+16].

2.3 Methods of feature selection in Multivariate time series

Most of the widely used approaches explained in the section 2.2 might require an expert intervention (cases in supervised feature selection approach), are only suitable for time independent labeled data and not for a multivariate time series data. Since the time dependency plays a major role in the robotics task, our feature selection methods need to take into consideration, that our features are not some constant values but rather be considered as time dependent. In this subsection, different methods used on multivariate time series data will be discussed and categorized.

2.3.1 Filter Methods

Filter methods rank features according to a relevancy score. The latter represents the main idea behind using filter methods in feature selection. Interactions and relationships between the data itself are extremely important for this Filter-based approach. Some of the methods that the filter approaches are based on to perform feature selection are: correlation, t-test, information gain, mutual information entropy, etc. Filter-based feature selection techniques are mainly applied because the following reasons:

- They proved to be efficient with high-dimensional data
- They are computationally efficient when compared to other techniques; Such computational aspects are a key factor to be taken into consideration if the application runs on low-power devices. [Bac16]

- they do not require any prior knowledge of the task; Filtering redundant features can be performed independently of the task we aim to learn.

Filter-based feature selection techniques, as mentioned earlier, use statistical metrics to determine the subset of features with potentially high accuracy. However, different filters may yield different filtered subsets, which may also delete some task-relevant data. Generally, the feature selection algorithms often rely on information theory concepts to choose what filter metrics to consider. We will talk briefly about feature selection metrics used in feature selection research;

Mutual information (MI):
Mutual Information Criterion (MIC) is a popular approach to analyze the correlation between features. It measures the dependency between the variables; The mutual information is a measure between two random variables X and Y, that quantifies the amount of information obtained about one random variable, through the other random variable. For two features, X and Y, the MI is estimated using:

$$I(X;Y) = H(X) + H(Y) - H(X,Y) \tag{2.1}$$

with

$$H(X,Y) = -\sum_{x\in X}\sum_{y\in Y} p(x,y)\log(p(x,y)) \tag{2.2}$$

$$H(X) = -\sum_{x} p(x)\log(p(x))(Shannon \quad entropy) \tag{2.3}$$

where p(x, y) is the joint probability of X and Y random variables and p(x), p(y) are the probability density functions of variable X and Y respectively. A large value of MI signifies high correlation of two variables. Zero value indicates that two variables are not correlated; if the mutual information between two variables is 0 ($p_{X,Y} = p_X p_Y$ and $MI(X;Y) = 0$) means that the two variables are statistically independent. For example, suppose X represents the roll of a fair 6-sided dice, and Y represents whether the roll is even (0 if even, 1 if odd). Clearly, the value of Y tells us something about the value of X and vice versa. That is, these variables share mutual information [VR17]. A multilabel feature selection algorithm using mutual information and ML-ReliefF for multilabel classification is proposed in [SSXZ20]; the mutual correlation is used here to improve the correlation degree between features and label sets. The novel mutual information-based correlation degree of features and label sets is developed to preprocess multilabel datasets for reducing the run time of ML-ReliefF (an algorithm constructed to evaluate the importance of features) [SSXZ20].

Pearson correlation (PC):
Pearson correlation (R-score) describes strength of the correlation between the features. The value of the correlation is calculated by dividing the covariance of two

features and dividing by the product of their standard deviations. Since the correlation value is divided by the product of the standards deviation of both signals, it is not affected by their scales For two features, X and Y, the MI is estimated using:

$$R = \frac{\text{cov}(X, Y)}{\sigma_X \sigma_Y} \tag{2.4}$$

where cov is the covariance of X and Y , σ_X represents the standard deviation of X and σ_Y represents the standard deviation of Y [VR17].
An example is shown in the following figure where the Pearson Correlation matrix of the used features in this thesis is calculated. In this case, filtering out features with that have high correlation (Usually greater than 0.5) might be an option: an example of 2 highly correlated feature can be feature 29 and 25.

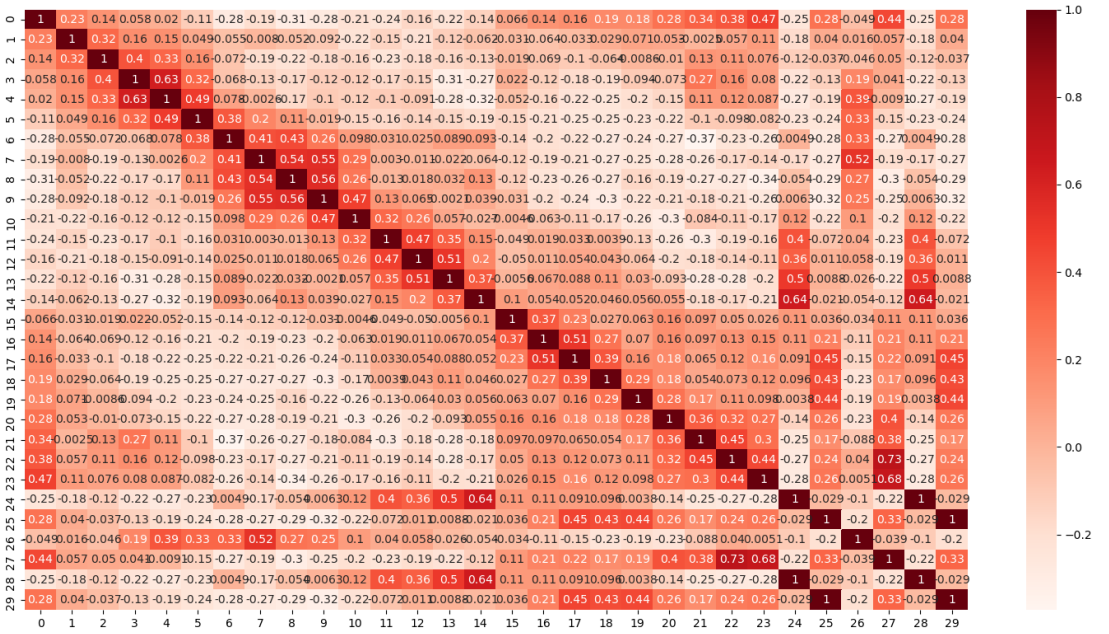


Figure 2.4: Pearson correlation matrix of the used dataset in this thesis (30 Features) [FBVV09]

Fisher criterion score (FS):
Fisher Criterion Score (F-Score) is widely used for classification tasks. This method computes the importance of each feature independently of the other features by comparing that feature's correlation to the output labels (supervised FS). The score is

calculated by measuring the variance between the expected value of the information and the observed value. When variance is minimized, information is maximized.

Features with higher F-Score have better separation ability in classification problems. Hence, features with low scores will be disregarded. Similar to the example of pearson correlation, a similarity matrix can be calculated and features with high F-score can be eliminated.
F-score is defined in the following:

$$F(i) = \frac{\left(\bar{X}_i^{(+)} - \bar{X}_i\right)^2 + \left(\bar{X}_i^{(-)} - \bar{X}_i\right)^2}{\frac{1}{n^+-1}\sum_{k-1}^{n_1}\left(\bar{X}_{k,i}^{(+)} - \bar{X}_i^{(+)}\right)^2 + \frac{1}{n_i-1}\sum_{k-1}^{n}\left(\bar{X}_{k,i}^{(-)} - \bar{X}_i^{(-)}\right)^2} \tag{2.5}$$

where $\bar{X}_i^{(+)}, \bar{X}_i^{(-)}$ and $\bar{X}_i$ are the averages of the i^{th} feature of the positive, negative and whole datasets; n+ and n - are the number of positive and negative instances, respectively; and $\overline{\mathrm{X}}_{\mathrm{k,i}}^{(+)}$ and $\overline{\mathrm{X}}_{\mathrm{k,i}}^{(-)}$ are the ith feature of the k^{th} positive instance and the ith feature of the k^{th} negative instance [VR17].

2.3.2 Clustering Methods

Filter methods which rely on dependence measures and predefined metrics are widely used in feature selection. However, recent studies showed that techniques that rely on feature clustering outperformed state of the art feature grouping algorithms[BMC20]. The main idea behind clustering different features together is to group the one who share similar properties in one set. in this section, some popular clustering methods will be discussed briefly here:

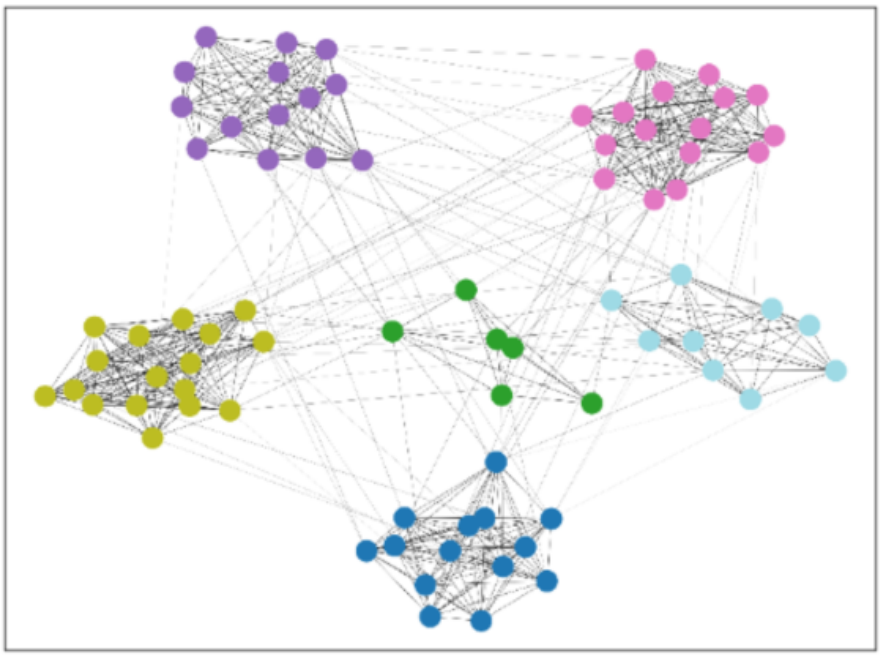


Figure 2.5: Example of clusters representation

K-means clustering (K-mean):
K-mean clustering algorithm is probably the most well-known clustering algorithm in feature selection. Given a set of points (samples) in a Euclidean space and a positive integer k (the number of clusters), split the points into k clusters so that the total sum of the (squared Euclidean) distances of each point to its nearest cluster center is minimized. This optimization objective is often called the k-means clustering objective [BDM09]. Obviously, the more number of features considered, the slower becomes the k-means clustering algorithm. Moreover, the existence of irrelevant features may affect the algorithm's capability on identifying the relevant features.
Given a matrix $A \in \mathbb{R}^{n \times d}$(representing n points $-$ rows $-$ described with respect to d features $-$ columns) and a positive integer k denoting the number of clusters, find the $n \times k$ indicator matrix X_{opt} such that

$$X_{\text{opt}} = \arg\min_{X \in \mathcal{X}} \left\| A - XX^T A \right\|_F^2 \tag{2.6}$$

The optimal value of the k -means clustering objective is

$$F_{\text{opt}} = \min_{X \in \mathcal{X}} \left\| A - XX^T A \right\|_F^2 = \left\| A - X_{\text{opt}} X_{\text{opt}}^T A \right\|_F^2 \tag{2.7}$$

In the above $\mathcal{X}$ denotes the set of all $n \times k$ indicator matrices X. From each cluster,

one representative will be selected and all other features from the same clusters will be ignored.

Agglomerative Hierarchical clustering:
In order to decide which clusters should be combined with the Agglomerative Hierarchical clustering method, a measure of dissimilarity between sets of samples is required. In most methods of hierarchical clustering, this is achieved by use of an appropriate metric (e.g. euclidean distances), and a linkage criterion which specifies the dissimilarity of sets as a function of the pairwise distances of observations in the sets. The choice of an appropriate metric will influence the size and shape of the clusters, as some elements may be close to one another according to one distance and farther away according to another. Some commonly used metrics for hierarchical clustering are: Euclidean distance,squared Euclidean distance, Manhattan distance, etc. The linkage criterion determines the distance between sets of observations as a function of the pairwise distances between observations[SB13]. Some commonly used linkage criteria between two sets of observations A and B are:
Maximum or complete linkage clustering:

$$\max\{d(a,b) : a \in A, b \in B\}. \tag{2.8}$$

Minimum or single linkage clustering:

$$\min\{d(a,b) : a \in A, b \in B\}. \tag{2.9}$$

In Agglomerative Hierarchical Clustering we will treat every data point as its own cluster, initially. Then subsequently we will keep merging nearest clusters together to form a new cluster. We will repeat this until all the data points are merged into a single cluster. For this reason, this method is not suitable for very large problems.Similarly to K-means clustering, one representative from each cluster will be selected to represent its group and the number of the selected features will decrease significantly.

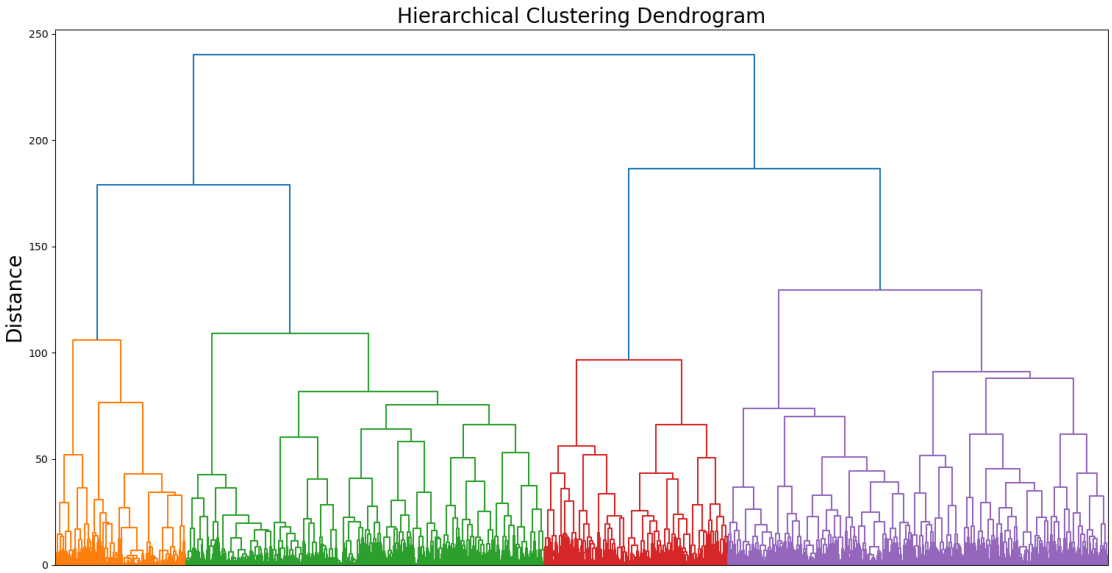


Figure 2.6: Example of clusters representation

Chapter 3

Feature selection for sensor time-series

3.1 Time series

Multivariate time series are usually used in multimedia, finance, medicine, speech recognition and robotics navigation and manipulation. A time series is a set of data points over time; Understanding how specific data points change over time is the basis for many statistical and business analyses ; a combination of the changes in relative humidity and temperature over time help dictate what we think of as "weather", for instance. Each time an action is performed within a network, logs are created that allow users to trace unique activities after they occurred. Even things such as economic data are technically time series data points, often compared to past results and projected into the future [dat20]

3.1.1 Characteristics

Time series have several characteristics that make their analysis different from other types of data.

- The time series variable (for example, velocity) may have a trend over time. This refers to the increasing or decreasing values in a given time series.
- The variable may exhibit cyclicity or periodicity. This refers to the repeating cycle over a specific period. For example, a robot can do the same subtask over and over in order to achieve a goal.
- The data will have serial correlation between subsequent observations.
- The data will almost always have an irregular component, which is referred to as the White Noise. This is the random variation not explained by any other factor. White Noise is a stationary process with a constant mean and variance [Tra19].

3.1.2 Applications in robotics

Robots rely exclusively on sensors in order to navigate or manipulate its environment. Usually, Robots are deployed with a set of sensors collecting the data over time under the form of Time series. This way allows the robots to have a model of the physical world enabling sensors to communicate information and link the operations with the actuators in order to achieve a certain task.

3.2 Importance of feature selection in robotics

The new trend in the robotics field is to enable robots to interact in human environments and become efficient when performing everyday tasks. One of the most famous ways to achieve such goal is a new way of teaching a robot, which is Programming by Demonstration (PbD). A human can teach a robot how to perform a task by physically guiding the robotic arm throughout the task (kinesthetic teaching). The demonstrated tasks may involve a sequence of basic sub-tasks, with different characteristics. Learning such a complex motion requires determining the specific constraints in each part of the task. In this area, feature selection may be fundamental in order to learn the task efficiently. Feature selection would help not only in decreasing the task complexity and computational cost but also in focusing on learning just the variables that are important for each region of the task[PUNB13].

3.2.1 Applications in robotics: importance of feature selection

Learning from redundant and noisy data affects the performance of the learning modules and make it less and less reliable. To tackles this issue, the availability of feature selection techniques becomes fundamental. Feature selection (FS) aims to improve the quality of the collected sensor streams by filtering out redundant and noisy data and selecting a subset of all features set with little to no human/expert intervention. With the help of these techniques discussed in this thesis, the various unsupervised FS methods will prove how their applications will improve the models interpretability, accuracy and generalization all while dramatically speeding up the learning process.

3.2.2 Incremental Cross-correlation algorithm

This method consists of identifying non redundant sensors data in an unsupervised fashion even in the presence of large portions of noisy features [Bac16]. It is based on the cross correlation of the Multivariate Time Series and determines whether a feature is redundant or not. The number of sensors streams feeding the learning models is then optimized which can be quite effective for limited computational requirements when working with heterogeneous information sources. One of the

main advantages of the Incremental Cross-correlation Filter (ICF) is the fact that it is completely independent of the learning process. Thus, it can be applied of sensors data and deliver the main features without being affected by the complexity of a specific learning model.
The ICF algorithm focuses on reduction of feature redundancy based on the pairwise cross-correlation of the data. The latter is a set of 4 selection/elimination rules which determines whether a features is in the selected features set of the deleted features set. The first phase is to calculate the feature redundancy matrix R in (0,1). All the diagonal values of this matrix are set to 0 set account for the trivial correlation (the signal with itself). The other matrix values are then obtained by calculating the maximum cross-correlation between all univariate sequences the following formula [Bac16]:

$$\phi_{x^1x^2}(\tau) = \sum_{t=\max\{0,\tau\}}^{\min\left\{\left(T^1-1+\tau\right),\left(T^2-1\right)\right\}} x^1(t-\tau)\cdot x^2(t) \tag{3.1}$$

where $\tau \in [-(T^1-1),\ldots,0,\ldots,(T^2-1)]$ and $T^1, T^2$are the time-series lengths
The values of the latter are then normalized to account for the amplitudes of different scales and reading. The normalized cross correlation is introduced:

$$\bar{\phi}_{x^1x^2}(\tau) = \frac{\phi_{x^1x^2}(\tau)}{\phi_{x^1x^1}(0)\cdot\phi_{x^2x^2}(0)} \tag{3.2}$$

where $\phi_{xx}(0)$ denotes the zero-lag autocorrelation, i.e., the correlation of a time-series x with itself. The normalized function $\bar{\phi}_{x^1x^2}(\tau)$ takes values in $[-1,+1]$, where a value of $\bar{\phi}_{x^1x^2}(\tau) = 1$ denotes that the two time-series have the exact same shape if aligned at time τ. Similarly, a value of $\bar{\phi}_{x^1x^2}(\tau) = -1$ indicates that the time-series have the same shape but opposite signs, while $\bar{\phi}_{x^1x^2}(\tau) = 0$ denotes complete signal uncorrelation [Bac16]. However, it is recommended to add some tolerance to the pair and account 0.99 as correlated and 0.1 as uncorrelated as thresholds. Based on this mathematical approach, the percentage of samples in which each pair i,j is correlated is calculated and if the both signals are correlated more than a given percentage defined by the expert the value (20 % is usually sufficient to detect redundancies), the corresponding Rij in the redundancy matrix ($R \in \{0,1\}^{D\times D}$ with D number of features) is set to 1 and 0 otherwise. The second phase consists of deleting all features with much noise; This phase relies on the fact that time series with noise tend to have a mean of autocorrelation equal to 0 in all lags other than lag=0 The third phase exploits the information in the redundancy matrix R and follows iteratively 4 different rules with the undeleted or unselected features.

- Rule 0: Select completely uncorrelated features.
- Rule 1: Delete completely correlated features.
- Rule 2: If R contains only ones select less correlated feature.

- Rule 4: If stuck, pick one feature for selection and one for deletion.

After the completion of all these steps, all features would be divided in 2 different sets; selected features and deleted features.
The ICF algorithm performs a multivariant feature selection process on pairwise features at a time to gain their dependencies information. Therefore, it maintains a limited computational complexity and can handle large amounts of heterogeneous data.

3.2.3 Complex Network-based clustering algorithm

Traditional unsupervised Feature selection methods based on dependence measures, such as correlation coefficients, linear dependence, or statistical redundancy, are already widely used. Recently, feature clustering demonstrated its merit in terms of accuracy with respect to other unsupervised approaches. In addition, clustering algorithms outperform state-of-the-art methods in detecting groups of similar features, as well as in selecting metrics (one or more features) out of every cluster to reduce the dimensionality of the data with more or less granularity based on the application. The most common approaches involve modifying the conventional clustering algorithms to adapt them to deal with raw time series or to convert time series into static data (feature vectors or model parameters) so that conventional clustering algorithms can be directly applied. In addition, according to the way clustering is performed, the algorithms can be grouped into whole time-series clustering and subsequence clustering, a valid alternative to reduce the computational costs by working separately on time series segments. It has been recently demonstrated that network approaches can provide novel insights for the understanding of complex systems, outperforming classical methods in the ability to capture arbitrary clusters. In particular, the weakness of conventional techniques resides in the use of distance functions which allow finding clusters of a predefined shape. In addition, they identify only local relationships among neighbor data samples, being indifferent to long distance global relationships [BMC20].
In this section, we will talk about the main steps of the proposed clustering approach:

- Remove time series noise through a low-pass filter.
- Segment time series y_n into consecutive non-overlapping intervals $s_n^1, s_n^2, \ldots, s_n^T$ corresponding to a fixed time amplitude L, where T is the number of segments extracted for each time series.
- Transform every signal segment $s_n^t (t = 1, \ldots, T \text{ and } n = 1, \ldots, N)$ into a weighted natural visibility graph G_n^f.
- Construct a feature vector $k_n^t = ((k_n^t)_1, (k_n^t)_2, \ldots, (k_n^t)_L)$ for each visibility graph $G_{n'}^t$ where $\left(k_n^f\right)_i$ is the degree centrality of the i th node in the graph and k_n^t the degree sequence of the graph.

- Define a distance matrix D^t for every t th segment ($t = 1, \ldots, T$), where the entry d_{ij}^t is the Euclidean distance between the degree centrality vectors k_i^t and k_j^t. Every matrix gives a measure of how different every pair of time series is in the t th segment.

- Compute a global distance matrix D that covers the full time period T where the entry (i, j) is computed as $d_{ij} = \frac{1}{T} \sum_{t=1}^{T} d_{ij}^t$, averaging the contributions of the individual distance matrices associated to every segment.

- Normalize D between 0 and 1, making it possible to define a similarity matrix as $S = 1 - D$, which measures how similar every pair of time series is over the full time period.

- Build a weighted graph C considering S as an adjacency matrix.

- Cluster the original time series by applying a community detection algorithm on the graph C and visualize the results through a force-directed layout: The community detection is done by the Louvainâs algorithm; it is done by maximizing the following modularity function [BMC20]:

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{d_i d_j}{2m} \right] \delta\left(c_i, c_j\right)$$

  where m stands for the number of edges of G, A_{ij} represents the weight of the edge between i and j (set to 0 if such an edge does not exist), d_i is the degree of vertex i (i.e. the number of neighbors of i), c_i is the community to which vertex i belongs and the δ -function $\delta(u, v)$ is defined as 1 if $u = v$, and 0 otherwise [DP15].

The next figure illustrates the flowchart of the methodology.

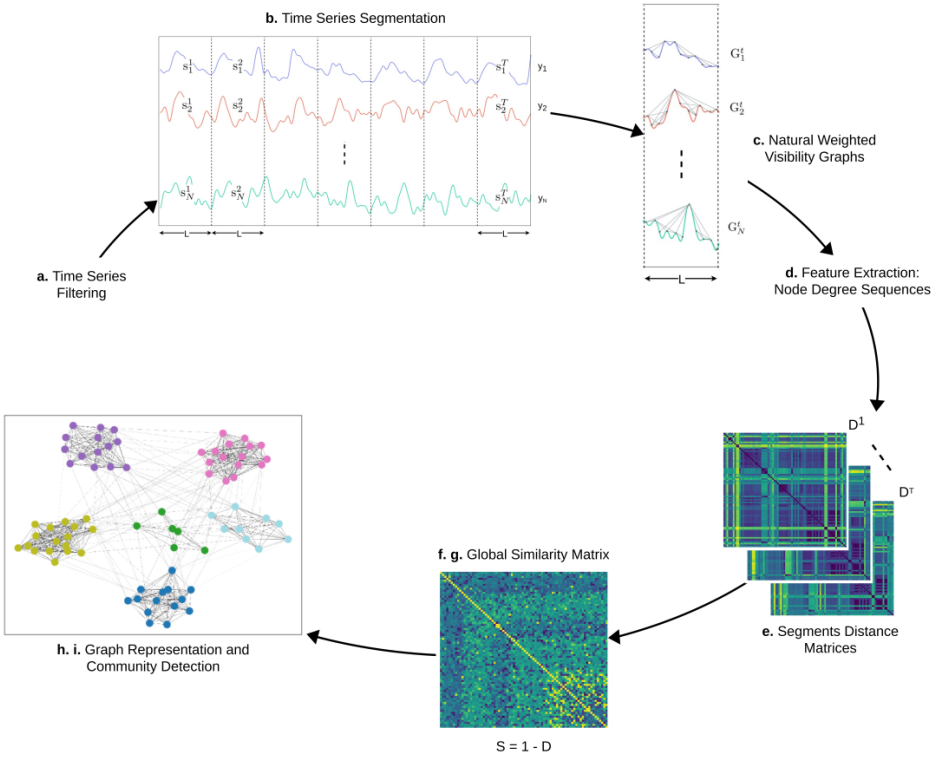


Figure 3.1: Flowchart of the proposed time series clustering methodology [BMC20]

3.2.4 Genetic algorithm: Wrapper method

Genetic algorithm (GA) is based on Darwinâs theory "Survival of the Fittest". That is the best solutions in the current population of features are selected for mating in order to produce better solutions. By keeping the good solutions and killing the bad solutions, we can reach an optimal or semi-optimal solution. In contrast to the previous two feature selection approaches, this new method is not time dependent. The selection process is exclusively influenced by the machine learning model Support Vector Machine (SVC). The idea behind this method is to select the 'best' feature based on their fitness level (classification accuracy). Since our experiment is a classification problem, we can have a feature selection algorithm that relies on how good a set of feature performs to predict the right class. The higher the accuracy the better the solution. Our used dataset is divided into train and test samples. Based on the train data, the SVC will be trained using the selected feature elements by each solution in the population. After being trained, it will be tested according to the test data.
Based on the fitness value of each solution, we can select the best of them as parents. These parents are placed together in the mating pool for generating offspring which

will be the members of the new population of the next generation. Such offspring are created by applying the crossover and mutation operations over the selected parents [Gad]. The following steps explains the main steps of the wrapper method.

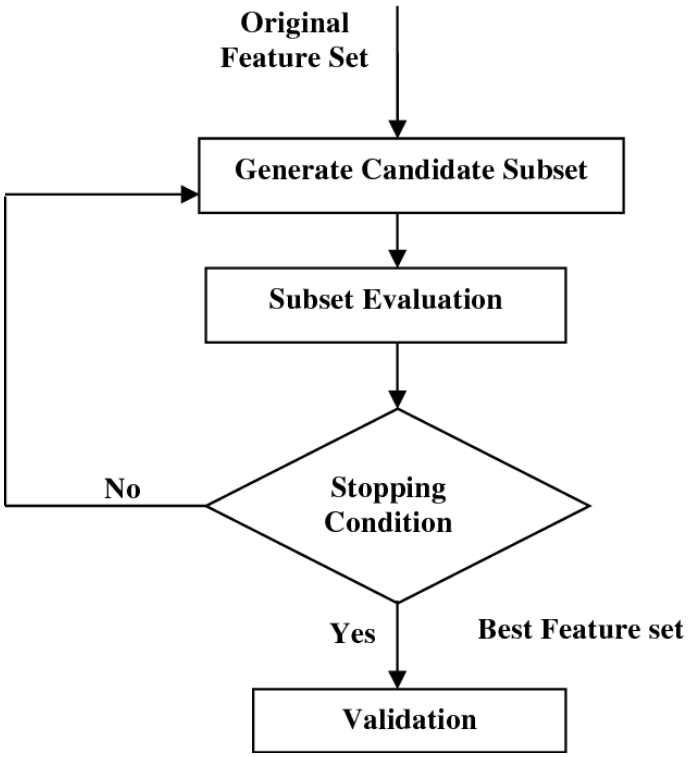


Figure 3.2: Genetic algorithm steps [VR17]

Fitness level:

Each feature in a classification problem has a degree of contribution in predicting the right class. GA assumes that combining 2 features with a high contribution in predicting the right class will produce an even better solution if combined in the same set. After each iteration, the best features will be saved for the next set and weakest features will be eliminated. The fitness level represents the accuracy level for each set of features in predicting the right class. This will help in creating better and better features sets after each iteration/generation and will try to find what combination of features is more likely to get the highest accuracy [Gad].

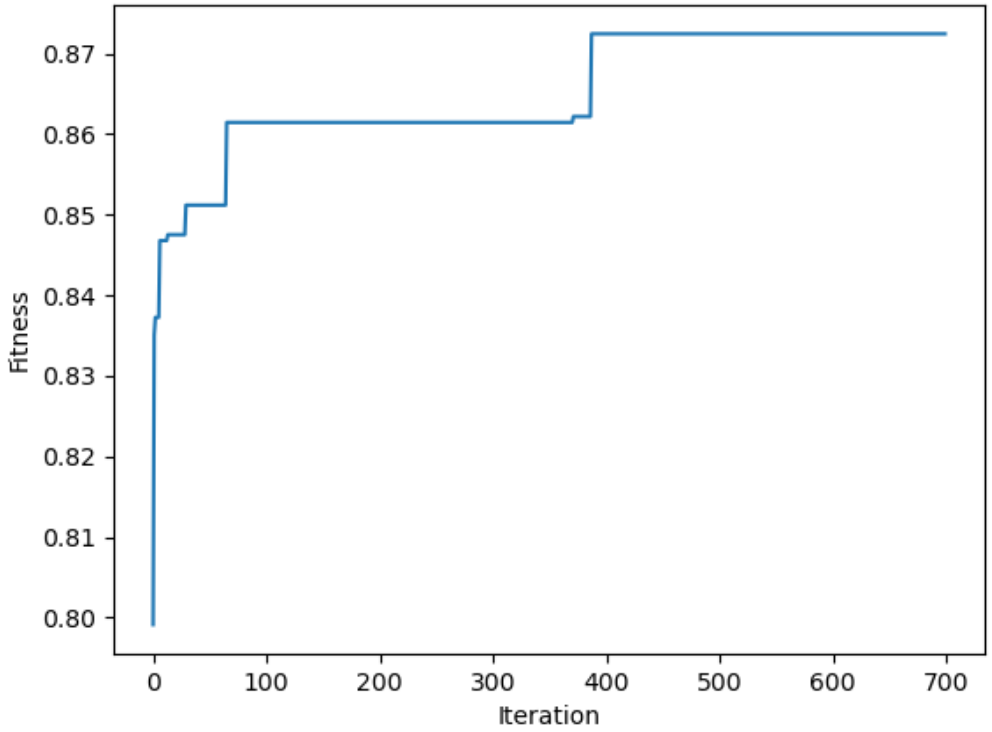


Figure 3.3: Progress of the fitness value for the selected features

Chapter 4

Evaluation of the different feature selection methods

In order to evaluate the effectiveness of the feature selection methods mentioned in the previous chapter, we used the dataset from a recent research paper [FBVV09], in which a robot should be able to infer the best policy (a set of decisions) to navigate his own environment with the help of a user demonstration. The features selected from each method will be fed to machine learning models (like support vector machine, K-Nearest neighbours, etc.) and depending on the quality of the selected features, the accuracy and performance will be compared. The policy can be easily obtained, since it is the same for the whole task and only 4 different actions are possible. In the next section, the experiment will be further discussed and explained in details.
This experiment has been conducted at the Federal University of Ceara, Brazil and was used in the work on the paper: Short-term memory mechanisms in neural network learning of robot navigation tasks: A case study [FBVV09]

Figure 4.1: Robot SCITOS G5.

4.1 Used data set

Before considering what features we might consider in this dataset, an experiment overview is given here to describe the most important steps of this experiment. The SCITOS G5 robot has a 360 degrees rotation capability and is equipped with 24 ultrasound sensors (sampling rate is 9 samples per second) arranged circularly around its 'waist' with a range within 20cm – 300cm (an ultra sound sensor every 15 degrees) . The robot task is to navigate through the room following the wall in a clock wise manner and avoid any obstacles that come in the way.
The robot will have to take one of 4 decisions at each time steps:

- move forward (Class 1)
- slight right-turn (class 2)
- sharp right-turn (class 3)
- slight left-turn (class 4)

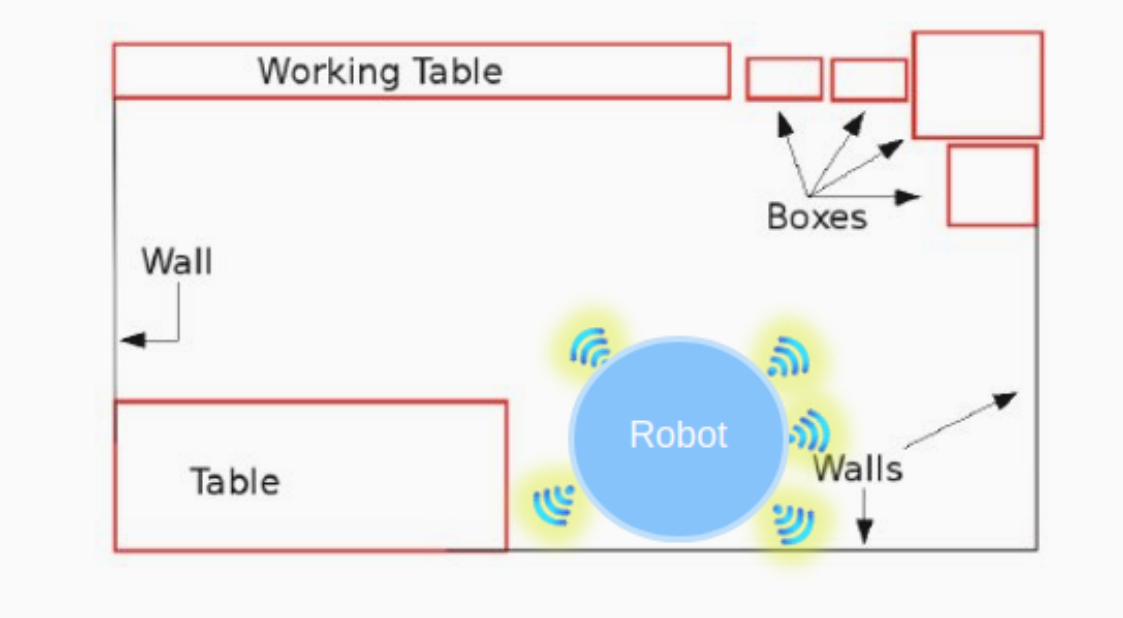


Figure 4.2: Experiment Overview

The data set consists of 30 different features and a class label which are constructed in the following way:

- The first 24 features represent the raw measurements of the ultra-sound sensors; feature 1 is the ultrasound sensor at the front of the robot (reference angle: 180°) feature 2 is at the reference angle: 165° and so on

- Feature 25 to feature 28 are the simplified distances (front, left,right,back) They consist, of the minimum sensor readings among those within 60 degree arcs located at the front, left, right and back parts of the robot, respectively.

- Feature 29 and 30 represent front and left distance (same as feature 25 and 26, respectively)

- Last column consists corresponding class label (Target) (move forward (Class 1),slight right-turn (class 2),sharp right-turn (class 3),slight left-turn (class 4))

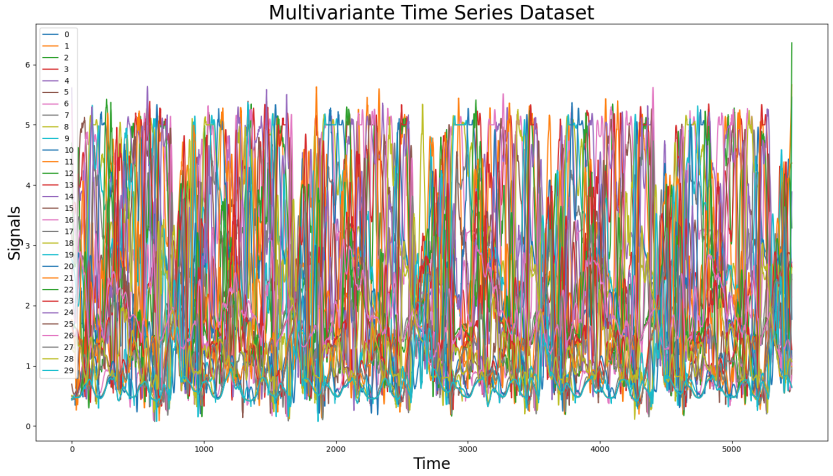


Figure 4.3: Visualization of all features on the robot

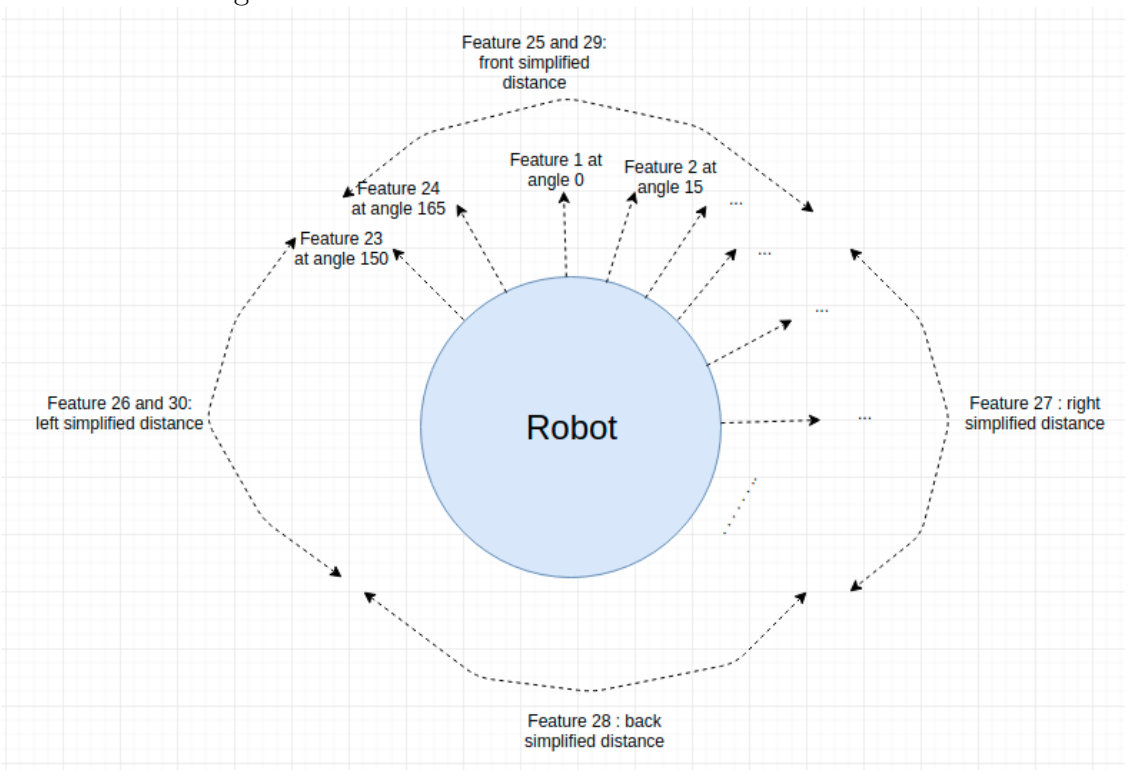


Figure 4.4: Graph of all 30 features/signals in time

The **goal** behind this experiment is to find which feature selection method delivers the 'best' features, which helps the robot determine the best decision or class to take.

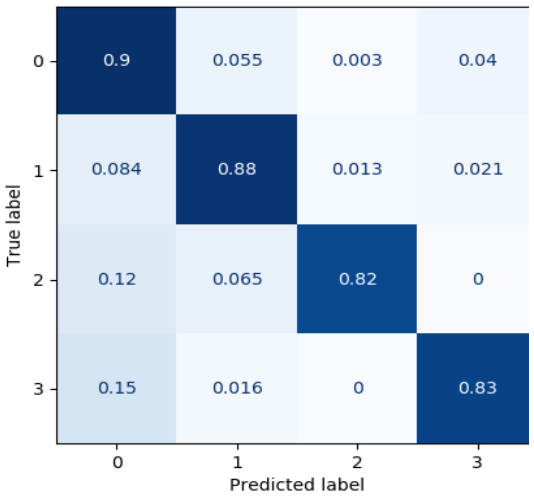
4.2 Results and evaluation

To evaluate the results of each feature selection algorithm, different machine learning models have been used to interpret the quality of the selection process. For this purpose, a support vector machine (SVM), a K-Nearest neighbours (KNN) , and neural network with 2 hidden layers (2 fully connected layers with 100 nodes each) have been used to check how good they can predict the corresponding class at each time step depending on the input features. The purpose of this step is to compare the performance of the classification models when given different input features and these input features will be determined with the help of the feature selection methods explained in the last chapter. All of the mentioned models are specifically designed for classification tasks. In our case, The model should decide what decision to make with the input features. In other words, the models will try to predict what move to robot should take depending on the input features; forward, slight right turn, sharp right turn or slight left turn . Since the performance and accuracy of each model highly depend on the quality of the input features, the results of the models vary significantly.

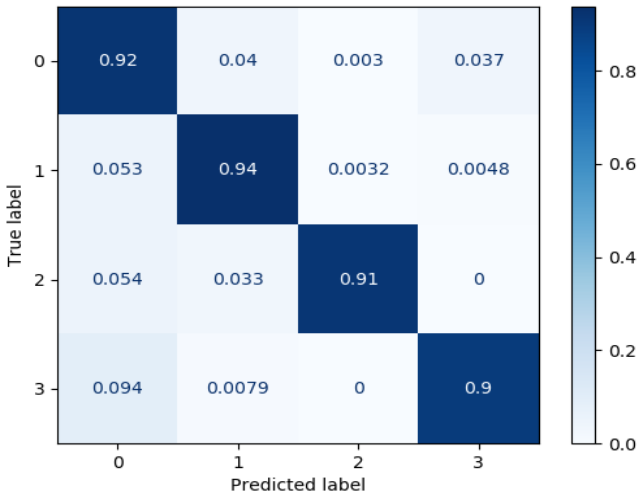
The following table summarizes the different results gathered from this experiment:

	Selected features: number of selected features	relies on predefined parameters	KNN accuracy in %	KNN run time	SVC accuracy in %	SVC run time	NN accuracy %	NN Run time
All features	ALL: 1 to 30: 30	NO	87.90	0.11	92.18	0.20	93.71	145
Clustering features	2,6,9,17,13, 25,26,23:8	NO	92.30	0.04	96.21	0.22	97.07	126
ICF features	3,4,5,6,7,8, 11,12,14,15 ,23:11	YES	87.90	0.08	85.77	19.04	91.20	142
Wrapper features	25,26,30:3	NO	98.90	0.03	99.88	0.16	98.78	111

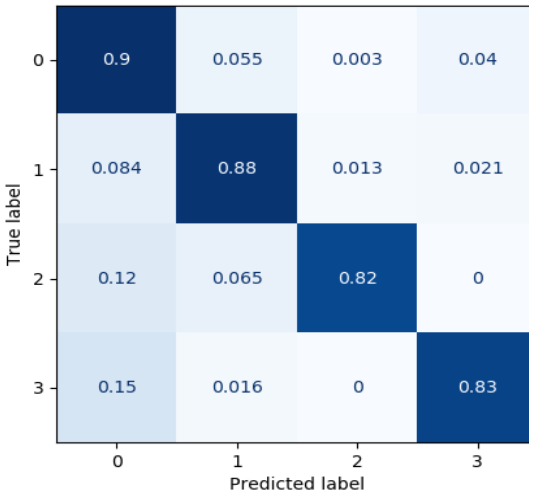
To visualize prediction accuracy of the learned models, we use confusion matrices. This way we can see how often did the machine learning models predicted the right class on the corresponding time step.



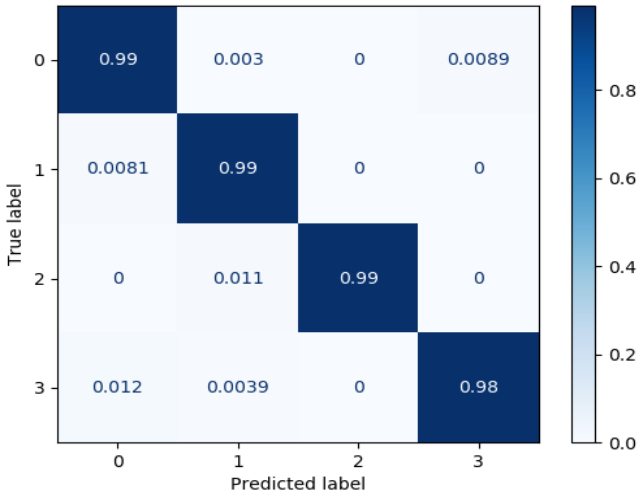
(a) KNN with all features



(b) KNN with Clustering features

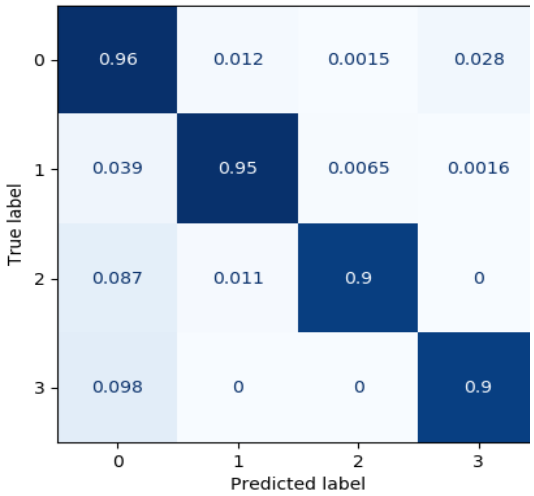


(c) KNN with ICF features

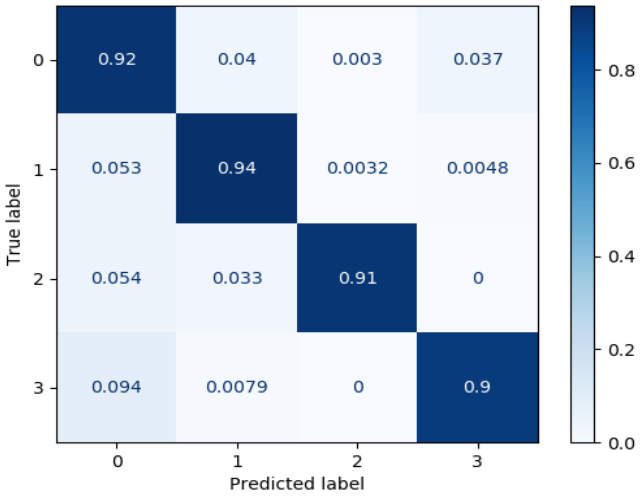


(d) KNN with wrapper features

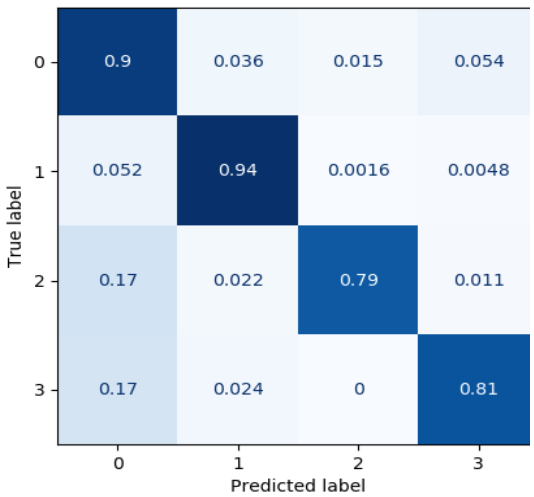
Figure 4.5: K-Nearest neighbours (KNN) confusion matrices with different input features from the feature selection algorithms



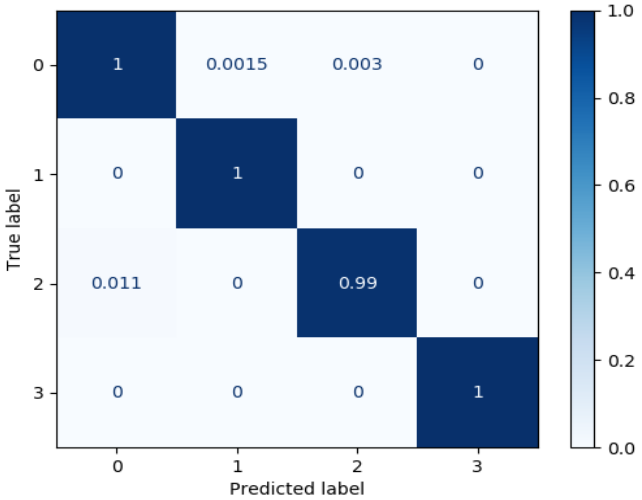
(a) SVC with all features



(b) SVC with Clustering features



(c) SVC with ICF features



(d) SVC with wrapper features

Figure 4.6: support vector machine model (SVM) confusion matrices with different input features from the feature selection algorithms

Note: for each of the machine learning models GridsearchCV has been used to determine the best hyper parameters:

- For K-Nearest neighbours (KNN): number of neighbors was 1
- For support vector machine (SVC) : kernel='poly',degree=4,coef0=3

- For the neural network : structure: /100/100/4 with 300 epochs

First of all, we can see from the table that selecting less features does not mean less run time and less accuracy. This statement can be justified by the results of working with all features (30) and working with the features selected from ICF. For example the SVC model accuracy dropped from a 92.18 % accuracy when working with all features to just 85.77 % accuracy when working the features selected with ICF. This can also seen with the confusion matrices in Figure 4.5; In comparison to working with other combinations of features, SVC with ICF features (c) has the lowest success rate in determining the right class. The run time of the SVC model working with ICF features was almost a 100 times slower than working with all features.

4.3 Interpretation of the results

To complete this thesis and understand the results of the machine learning models, it is necessary to analyse the causes that led to them. In this chapter, we will try to explain why the performance of the implemented feature selection methods varied. First we will take a naive approach for interpretation, secondly, we will rely on some metrics to test the importance of the features and lastly, we will focus on the neural networks learning curves.
Note: All of the algorithms that have been implemented for the unsupervised feature selection are based on the following 3 references:

- Cross-correlation Filter (ICF): Unsupervised feature selection for sensor time-series in pervasive computing applications' [Bac16]
- Complex Network-based clustering Approach : Time Series Clustering: A Complex Network-Based Approach for Feature Selection in Multi-Sensor Data [BMC20]
- Wrapper approach algorithm : Practical Computer Vision Applications Using Deep Learning with CNNs: With Detailed Examples in Python Using TensorFlow and Kivy [aR18]

Before starting to analyze the performance of each machine learning model and the feature selection methods based on mathematical models and metrics, we consider, firstly, a rather naive way to interpret the results; According to 4.2 and 4.4, in order for the robot to follow the wall in **a clock wise manner**, the main features that the robot should use in order to figure out the best policy are the sensors/features pointing to the wall. More specifically feature 25 or 29 (front), 26 or 30 (left) as simplified distances and features 24,0,1,2, 23,22,21,20 as ultra sound values which make out the simplified distances front and left). The latter can detect if the distance between the robot and wall/objects is too small and therefore change the direction

of its movement.
According to 4.2, the ICF feature selection method fails to select the most important features (25/29 or 26/30). In fact, the ICF ignores all simplified distances and focuses rather on the ultra sound sensors pointing from the back and right of the robot. In contrast, the wrapper method, all important simplified distances (25 and 29) which helps find the position of robot accurately to the wall. The Complex Network-based clustering approach is a mix between ultra sound data and simplified distances.
In this step, we will consider the a information gain to test how good did the FS method performed: Information Gain is calculated for a split by subtracting the weighted entropies of each branch from the original entropy, meaning in our case, the higher the Information Gain of a feature, the more Entropy removed. This mean, zero entropy means maximum information gain! The entropy is calculated in the following way:

$$E = -\sum_{i}^{C} p_i \log_2 p_i \tag{4.1}$$

Information gain=Entropy of distribution before the split - entropy of distribution after it

The formulas are taken from this [Zho19]

The following values represent the information gain of our 30 features in order:

0.312	0.286	0.304	0.341	0.372	0.3882	0.299
0.376	0.327	0.302	0.301	0.359	0.385	0.418
0.590	0.323	0.385	0.489	0.466	0.426	0.261
0.285	0.376	0.344	0.779	0.603	0.286	0.361
0.782	0.603					

from these values, we can tell that features 30/26 with 0.603 ,25/29 with 0.782 and 15 with 0.590 have the highest information gain values. This explains why the machine learning models working with features from the wrapper method got the highest accuracy 4.2. However, the model working with the ICF features had only worked with features with low information gain. The model working with the complex Network-based clustering method features also manage to find get a good accuracy since the FS method managed to select both simplified distances (feature 25 and 26) along other features.

In general, ICF has (in almost all models) the lowest accuracy and takes the longest time to converge. The latter can be justified by the fact that ICF takes some strong assumptions on the nature of the data and has some predefined values that directly

influence the selection process; According to the papers that ICF was based on, if the auto-correlation (correlation of a signal with a delayed copy of itself as a function of delay) of every time-series has a mean and variance of auto-correlation close to 0 off lag $\tau = 0$ (for all other lags),it should be considered as noise. Since the mean and variance of auto correlation in practice cannot be exactly 0, a threshold in the algorithm has been assigned to each one. If the mean is smaller than 0.1, the corresponding feature should be deleted and be considered as noise.
Mathematically we can express the conditions as the following:

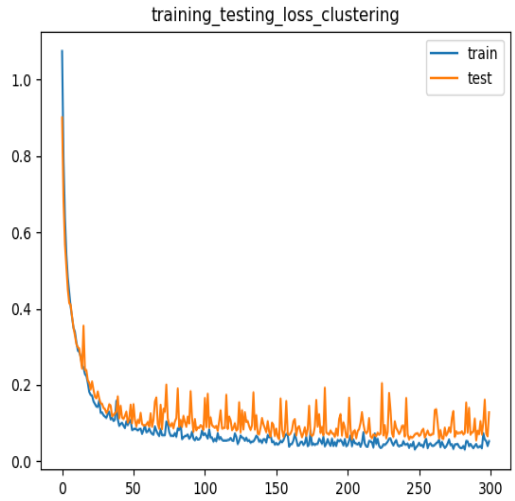
$$\psi_i^* = \frac{\sum_{n=1}^{N} \frac{1}{T^n - 1} \sum_{\tau \neq 0} \left| \phi_{x_i^n x_i^n}(\tau) \right|}{N} < 0.1 \tag{4.2}$$

where $\phi_{x_i^n x_i^n}(\tau)$ is defined in (3.1) and T^n is the length of the time series. Unfortunately, these strong assumptions do not consider the nature of the data of each experiment. The algorithm filters out 19 features as they fulfil the above conditions and saves only the 11 features seen in the table 4.2.
In contrast to the ICF algorithm, the Complex Network-based clustering [BMC20] does not use any predefined parameters as thresholds to detect redundancy between the features. It uses different tools from complex network theory and community detection to cluster similar features together. More specifically, visibility graphs are used to map the features from time domain to the network domain. The results of this method, showed a significant improvement comparing to working with all features; In all Machine learning models that take their input features from the Complex Network-based clustering method , we noticed a 4%- 5% accuracy improvement and shorter run time all while reducing the dimensionality of the dataset by 73.33%. This method proved to be quite successful for our feature selection task as it proved its merits on various machine learning models.
The last method is the wrapper method which does not take the time dependency into consideration. The only important factor that is taken into consideration in this approach is the fitness score which is given with an another SVC model. It relies on 'the survival of the fittest' which implies that only features who contribute the most to improving the accuracy of the machine learning model are selected. Since this approach is not only able to detected redundancy between features but also select task relevant ones, it displays the strongest performance between all models; this method showed great performance by improving the model's accuracy by 7%- 8% all while reducing the dimensionality of the dataset by 90%
To analyse the quality of the selected features even further, the learning curves of the designed neural network are graphed in the following figure. In the process of designing the neural network, the loss will be calculated with the help of Categorical CrossEntropy which is a common loss function for multiclass classification.

$$\mathrm{CCE}(p, t) = -\sum_{c=1}^{C} t_{o,c} \log \left(p_{o,c} \right) \tag{4.3}$$

C represents all the classes (in our case 4), prediction p and target t and observation o. If for example the robot should take a slight left turn the target will be [0,0,0,1], a confident and well-trained neural network would in that case output e.g. [0,0.1,0.04,0.95]. Formula was obtained from this [ver]

(a) all features



(b) features selected from Complex Network-based clustering approach

(c) features selected from ICF approach

(d) features selected from wrapper method

Figure 4.7: Training and validation curves of the neural network

By training the neural network, we've been plotting the loss on the training and the validation sets epoch by epoch (total of 300 epochs). the training loss will go down either when the models learn the actual task or when it learns from irrelevant redundant data [Zho19]. This is the case with the different features sets we used. However, The validation loss will go down only if the model learns the actual task. The size of the gap gives an idea how much noise/irrelevant data the model has learned and therefore, in our case, the quality of the selected features. The plots can be categorized into two sections:

- Cases of overfitting: Figures (a) and (c): in (a) we can see the gap between the learning curves when working with all features is quite significant which also explains the low accuracy results. Moreover, the learning curves of the ICF features (c) seem to diverge through the end which implies that the ICF algorithm gave only selected potentially redundant or irrelevant features to learn the task. This is also reflected by the very poor accuracy results of this neural network-.

- Cases of a good fitting model: Figures (b) and (d): in (b), the gap of the learning curves is quite small which explains a good model that was able to generalize the task. In (d) the gap between the learning curves is quite insignificant which implies that the wrapper method did select the right task relevant features. The latter is also reflected in the accuracy results of the wrapper method.

4.4 Advantages and disadvantages of each method

In this final section, we will discuss the advantages and disadvantages of each feature selection method. Obviously, one should choose the correct approach depending on the characteristics of the experiment. In our wall following task, where the robot should learn the right policy to follow the wall, we needed to consider the time as an important factor in some feature selection methods (ICF and Complex Network-based clustering approach). However this might not be the case for other tasks where the data is not affected by the time. Therefore, this will be as general as possible and not task-specific.

Correlation approach: Incremental Cross-correlation Filter (ICF):

- Advantages: Relying on predefined dependence measures like correlation is a good way to consider how features change over time. The time dependencies may play a significant role especially with tasks involving transferring knowledge to a robot. If the robotics task can be be segmented into different sub tasks, analyzing the scores like cross correlation of mutual information 2.1 between the features can be quite helpful [PUNB13]

- Disadvantages: This approach will be highly dependent on the users choice in certain thresholds. In our case, Many features were filtered out just because they were perceived as noise even though they were not. Moreover, the user must define himself the threshold at which 2 features are considered correlated and therefore redundant. For this method, doing the experiment under different conditions is necessary in order to determine what experimentally the best values to be used as thresholds.

Complex Network-based clustering approach:

- Advantages: clustering-based feature selection methods have demonstrated to outperform traditional approaches in terms of accuracy [BMC20]. They proved to quite effective when dealing with complex nonlinear dynamic systems associated with gathering both homogeneous and heterogeneous data streams e.g. Timeseries [BMC20]. In our case, we have been able to reduce the dimensionality of the dataset by 73.33%. Another advantage is discovering hidden relationships between the single features consisting the data set enriching the information content about the features roles within the network.

- Disadvantages: Unfortunately, clustering approaches rely on other algorithms like louvain algorithm to maximize a predefined modularity function. Only with this way, the community detection can be performed and the redundant features can be grouped together.

Wrapper approach: Genetic Algorithms (GA):

- Advantages: One of the biggest advantages of the wrapper method is that it is able to not only eliminate redundant data but also select task-relevant ones. Unlike the previous methods, it aims to choose a subset of input variables by eliminating irrelevant features only based on the fitness level. In our case, this method improved the accuracy by 9 % when compared with using all features all while filtering out 90 % of the features. It is quite effective for problem involving a relatively small number of features. Since the algorithm relies on Support Vector Machine (SVM) to test the fitness level (classfication accuracy).

- Disadvantages: GA's disadvantage is that it cannot find exact global optimum as there is no guarantee for a best solution. It is dependent on the number of iterations, population size, number of mutations and a number of other parameters to control their evolutionary search for the solution to their given problems [Ele17] . Moreover, this wrapper method is a greedy algorithm meaning it does not consider any time dependencies or other relationships between the features and focuses only one one single criteria, which is finding out what combination of features has the bigger fitness score. Finally, this method tries different feature combinations which can be computationally expensive if the feature set is large.

Chapter 5

Conclusion

In this chapter, we will draw a conclusion from the results and interpretation from the previous chapter.
The primary objective of improving the accuracy and performance of learning the wall following task, by applying unsupervised feature selection approaches, is successfully achieved. Dealing with problems involving a large number of input features compromising of sensor time-series data can contain noisy, highly redundant information which can hamper the deployment of effective predictive machine learning models. As noted in chapter 1, most of state-of-the-art feature selection method algorithms are poorly suited to deal with multivariate time series data with characteristics like noise or the heterogeneous nature. To address these limitations, we discussed in this thesis multiple feature methods capable to cope with the mentioned characteristics. Even though not all methods were not equally successful in achieving the task, we can see that feature selection can be sometimes essential in tasks involving a large number of features by identifying non-redundant sensor sources in an unsupervised fashion even in presence of a large proportion of noisy features.
Through this work, robotics engineers can rely on some of the proposed feature selection methods in order to find what sensors are actually needed for the robot to learn a certain task. These methods can reduce the overall time, processing power and the budget, which are key factors in the designing of any project.

Limitations: Even though, the proposed methods helped to recognize the redundant and task relevant features. They have a multiple limitations; The filter and the Complex Network-based clustering Approach,for example, cannot recognize task relevant features. As they only rely of dependence measures, such as correlation coefficients or linear dependence without any prior knowledge of the task, some important features who might contribute to generalize the learning task might be considered as noise or irrelevant and therefore deleted. Concerning the wrapper method, multiple feature combinations must be tried. This can be very computationally expensive when dealing with thousands of features.

Chapter 6

Future Work

Following are the tasks which can extend the scope of the current work:

- Unlike this work, which focuses on filtering the features after the experiment has been performed.Algorithms method can be developed to perform feature selection while doing the experiment.
- Unlike the implemented wrapper method, a better, more intelligent wrapper method can recognize important features without trying multiple combinations.
- The Incremental Cross-correlation Filter (ICF) algorithm can be improved further to get better performance.
- Instead of using machine learning models (in a supervised manner) to test the performance of the select features. A reinforcement leaning approach could be used; for example, we can see which combination of features leads to a high reward function value.

Bibliography

[AMXS16] Hiromasa Arai, Crystal Maung, Ke Xu, and Haim Schweitzer. Unsupervised feature selection by heuristic search with provable bounds on suboptimality. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[aR18] Aá¸¥mad Jād al Rabb. *Practical Computer Vision Applications Using Deep Learning with CNNs: With Detailed Examples in Python Using TensorFlow and Kivy*. Apress, 2018.

[Bac16] Davide Bacciu. Unsupervised feature selection for sensor time-series in pervasive computing applications. *Neural Computing and Applications*, 27(5):1077–1091, 2016.

[BCAB18] Verónica Bolón-Canedo and Amparo Alonso-Betanzos. Recent advances in ensembles for feature selection. 2018.

[BDM09] Christos Boutsidis, Petros Drineas, and Michael W Mahoney. Unsupervised feature selection for the k-means clustering problem. In *Advances in Neural Information Processing Systems*, pages 153–161, 2009.

[BMC20] Fabrizio Bonacina, Eric Stefan Miele, and Alessandro Corsini. Time series clustering: A complex network-based approach for feature selection in multi-sensor data. *Modelling*, 1(1):1–21, 2020.

[BN01] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Nips*, volume 14, pages 585–591, 2001.

[Bro09] Gavin Brown. A new perspective for information theoretic feature selection. In *Artificial intelligence and statistics*, pages 49–56. PMLR, 2009.

[dat20] Influx data. The importance of time series database: Influxdata paper, Jul 2020. URL: `https://www.influxdata.com/resources/why-time-series-matters-for-metrics-real-time-and-sensor-data/`.

[DL97] Manoranjan Dash and Huan Liu. Feature selection for classification. *Intelligent data analysis*, 1(1-4):131–156, 1997.

[DP15] Nicolas Dugué and Anthony Perez. *Directed Louvain: maximizing modularity in directed networks*. PhD thesis, Université d'Orléans, 2015.

[EHJT04] Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.

[Ele17] Electricalvoice Electricalvoice. Genetic algorithm: Advantages amp; disadvantages, Aug 2017. URL: `https://electricalvoice.com/genetic-algorithm-advantages-disadvantages/`.

[FBVV09] Ananda L Freire, Guilherme A Barreto, Marcus Veloso, and Antonio T Varela. Short-term memory mechanisms in neural network learning of robot navigation tasks: A case study. In *2009 6th Latin American Robotics Symposium (LARS 2009)*, pages 1–6. IEEE, 2009.

[Gad] Ahmed Gad. Feature reduction using genetic algorithm with python. URL: `https://www.kdnuggets.com/2019/03/feature-reduction-genetic-algorithm-python.html`.

[Gol89] David E Goldberg. Genetic algorithms in search, optimization, and machine learning. addison. *Reading*, 1989.

[HN03] X He and P Niyogi. Locality preserving projections, advances in neural information processing systems16, vancouver. *British Columbia, Canada*, 2003.

[HS99] Mark A Hall and Lloyd A Smith. Feature selection for machine learning: comparing a correlation-based filter approach to the wrapper. In *FLAIRS conference*, volume 1999, pages 235–239, 1999.

[Kan05] Jaewoo Kang. Data models for exploratory analysis of heterogeneous microarray data. Technical report, North Carolina State University. Dept. of Computer Science, 2005.

[KJ97] Ron Kohavi and George H John. Wrappers for feature subset selection. *Artificial intelligence*, 97(1-2):273–324, 1997.

[NF77] Patrenahalli M. Narendra and Keinosuke Fukunaga. A branch and bound algorithm for feature subset selection. *IEEE Transactions on computers*, 26(09):917–922, 1977.

[PUNB13] Lucia Pais, Keisuke Umezawa, Yoshihiko Nakamura, and Aude Billard. Learning robot skills through motion segmentation and constraints extraction. In *HRI Workshop on Collaborative Manipulation*. Citeseer, 2013.

[SB13] K Sasirekha and P Baby. Agglomerative hierarchical clustering algorithm-a. *International Journal of Scientific and Research Publications*, 83:83, 2013.

[SFCOET19] Saúl Solorio-Fernández, Ariel Carrasco-Ochoa, Luis Enrique Erro, and Sta Ma Tonantzintla. Unsupervised feature selection method for mixed data. 2019.

[SFK10] Pawel Smialowski, Dmitrij Frishman, and Stefan Kramer. Pitfalls of supervised feature selection. *Bioinformatics*, 26(3):440–443, 2010.

[SSXZ20] Enhui Shi, Lin Sun, Jiucheng Xu, and Shiguang Zhang. Multilabel feature selection using mutual information and ml-relieff for multilabel classification. *IEEE Access*, 8:145381–145400, 2020.

[Tra19] Finance Train. Characteristics of time series, Oct 2019. URL: `https://financetrain.com/characteristics-of-time-series/`.

[ver]

[VR17] S Visalakshi and V Radha. A hybrid filter and wrapper feature-selection approach for detecting contamination in drinking water management system. *Journal of Engineering Science and Technology*, 12(7):1819–1832, 2017.

[YHQ+16] Xu-Kui Yang, Liang He, Dan Qu, Wei-Qiang Zhang, and Michael T Johnson. Semi-supervised feature selection for audio classification based on constraint compensated laplacian score. *EURASIP Journal on Audio, Speech, and Music Processing*, 2016(1):1–10, 2016.

[Zho19] Victor Zhou. A simple explanation of information gain and entropy, Jun 2019. URL: `https://victorzhou.com/blog/information-gain/`.