**fierClass**: a multi-signal, cepstrum-based, time series classifier

S. Gelmini\textsuperscript{a,}\textsuperscript{*}, S. Formentin\textsuperscript{a}, S. Strada\textsuperscript{a}, M. Tanelli\textsuperscript{b,}\textsuperscript{**}, S. Savaresi\textsuperscript{a}

\textsuperscript{a}Politecnico di Milano - Piazza Leonardo da Vinci 32, 20133 Milano (MI), Italy

Abstract

The task of learning behaviors of dynamical systems heavily involves time series analysis. Most often, to set up a classification problem, the analysis in time is seen as the main and most natural option. In general, working in the time domain entails a manual, time-consuming phase dealing with signal processing, features engineering and selection processes. Extracted features may also lead to a final result that is heavily dependent of subjective choices, making it hard to state whether the current solution is optimal under any perspective. In this work, leveraging a recent proposal to use the cepstrum as a frequency-based learning framework for time series analysis, we show how such an approach can handle classification with multiple input signals, combining them to yield very accurate results. Notably, the approach makes the whole design flow automatic, freeing it from the cumbersome and subjective step of handcrafting and selecting the most effective features. The method is validated on experimental data addressing the automatic classification of whether a car driver is using the smartphone while driving.

**Keywords**: Cepstrum; time series; classification; frequency-domain analysis.

1. Introduction

In the analysis of dynamical systems, time plays a key role. In fact, all the related signals evolve over time and any information that has to be inferred from data, ranging from mathematical models, to fault isolation, to systems use mode detection, can be considered as the result of a learning process with time series as predictors.

Modeling from time series has been the objective of system identification for more than 70 years so far, [1]. However, in fault or use-mode detection, the learning task can be more properly formulated as a classification problem and such an issue has been less deeply investigated within the control community, due to the categorical nature of the final result which is opposed to a general continuous output of control processes. Time series classification has been widely studied for problems related to fault detection in dynamic system [2, 3], but contributions can be found also in health-monitoring [4, 5] and predictive maintenance [6, 7]. These classification algorithms have been used not only in the most traditional engineering fields, but also in more broad sense for anomaly detection, prediction and forecasts problems such as, e.g., economics [8], finance [9], climate monitoring [10]. As the interested range of applications has spiked with the introduction of new technologies that benefit from a digital description, the problem of classifying structured time signals has increased and a large number of innovative time series classification algorithms have been proposed in the data-mining literature in the last years, see, e.g., [11].

As better illustrated in Section 2.1, the research community is constantly seeking for improvements of what currently represents the state of the art, though the performance of some these algorithms are difficult to beat. For instance, authors of [12] have recently proposed an innovative approach based on the widely used pair 1-Nearest Neighbor and Dynamic Time Warping algorithms, improving what is already known to outperform most of other time series classification algorithms. Furthermore, time series classification becomes even more challenging when dealing with multivariate time series. In fact, as discussed in [13] and its references, many attribute-value representation methods (i.e., methods that extracts a set of attributes from the data set, reducing the problem dimensionality and/or the length of the time series) are extensively discussed in the literature, though “these approaches may require in-depth domain knowledge for designing, may be labor intensive, and/or time consuming”, as remarked in [14]. An attempt to solve this problem is proposed by the authors of [15], in which they combine the benefits of recurrent neural networks (RNN) and adaptive differential evolution (ADE) algorithms, showing promising results, with non negligible shortcomings.

In general, the existing classification methods can be classified in two main classes: *model-based* and *data-driven* techniques. Model-based classification algorithms (see, e.g., [16]) aim to fit a dynamical model to each time series and then measure the similarity between the original series and the ones generated by their models. Such tech-
niques have shown a great potential against over-fitting and allow comparison among signals of different lengths; however, they usually provide worse overall performance with respect to other classification methods. [11].

Data-driven approaches include a large variety of techniques. In some cases, all the time samples of the series of interest can be taken as predictors (see, e.g., [10]). Such a naïve solution is intuitive, but usually also computationally intense (and sometimes unfeasible). More frequently, a set of features characterizing the time series (e.g., mean, variance, peak values, just to name a few) are used as predictors to which static classifiers are applied. Even if such approaches are apparently model-free, the choice of the most interesting features to train a classification model is all but straightforward and it usually relies upon some prior knowledge of the system behavior, and in any case introduces a subjective step which at best determines the shape of the final solution, and in the worst can prevents finding a satisfactory answer to the original problem.

In general when dealing with time series data, a frequency-domain analysis is also possible, which allows inferring information on the underlying physical phenomenon by inspecting the frequency ranges on which it acts. In control systems design, analysis in the frequency domain is a must. Inspired by this custom habit, in this work we propose a novel frequency-based approach for classification using time series data, leveraging the computation of the cepstrum coefficients, [17]. We refer to this classifier as fierClass, using the same lexical anagram that links spectrum to cepstrum to connect the terms classifier and fierClass. The proposed approach combines the flexibility of data-driven classifiers, meaning that the model identification phase is not necessary, with features that capture the underlying dynamics in the frequency-domain.

Although the cepstrum was first defined in the signal processing community as early as in 1963 [18], its use for time series clustering and modeling of dynamical systems has been only recently prompted by B. De Moor and coauthors, see, e.g., the recent [19]. More specifically, the proposed cepstrum-based classification has several advantages: first of all it allows removing the subjective (and time-consuming) step of feature engineering and selection, offering as native features the cepstrum coefficients themselves, and their distance, properly defined, as similarity measure to guide the classification process. The analysis of time series in this setting becomes easier and the classification policy straightforward. Additionally, thanks to this mathematical framework, the choice of the most informative frequency range is simply obtained tuning the hyperparameters. Further, thanks to the novel extension presented in this work, a cepstrum-based classifier may be shown to handle a vector-valued input, to deal with applications in which many different sensors provide the time series that can be used as inputs in the classification process, one of the most important problems in multivariate time series classification problems. In fact, we show how the combination of the cepstrum coefficients in the classification problem amounts to the convolution of the original signals in the time domain.

To test the capabilities of the proposed approach within a challenging setting, in this work the multi-signal cepstrum-based classification approach is applied to the problem of automatic detection of phone-usage while driving. This is a complex classification problem, the relevance of which is motivated by the need of enlarging the current driving-style estimation algorithms adding the important information of the amount of time spent using the phone during a journey. Of course, this is a major source of danger, and must thus be considered to form a reliable risk-index of a driver. This is of interest, for example, in the field of insurance telematics, [20, 21].

To perform such a classification, the time series measured by the smartphone sensors are used, that is three accelerations and three angular velocities. Based on the resulting time series, the novel approach is tested, and compared to a more classical SVM-based classifier. To our best knowledge, this is the first work where cepstrum-based signal processing is used for use mode detection via classification.

The paper shows that the cepstrum-based approach yields results which are directly comparable to the best SVM classifier, with the significant advantage of avoiding the qualitative step of feature selection, which is in general highly problem-dependent, and requires a deep knowledge of the problem domain. Furthermore, such a selection step inevitably adds arbitrariness to the final results, while the proposed cepstrum-based approach is fully automatic and quantitative. Moreover, the multi-signal approach allows one to automatically select which is the best combination of the available signals to be used for the classification task.

The rest of the paper is organized as follows. Section 2 discusses the existing panorama in the context of time series classification, to offer a comprehensive contextualization of the present work. Further, Section 3 presents fierClass, the proposed multi-signal, cepstrum-based classification approach, discussing both training and classification steps. Then, Section 4 is devoted to a detailed description of the experimental case study considered in this work, that is the automatic classification of phone usage while driving, discussing all the steps leading to the final solution.

2. Preliminaries

In this section, a partial overview on the state-of-the-art algorithms for time series classification is assessed, remarking the results already achieved and the limits of the approaches presented in the literature. Then, before introducing the proposed method, cepstrum, the mathematical framework used by fierClass, is briefly introduced and some of its properties discussed. Finally, the main uses of cepstrum are illustrated, familiarizing with one tool used in the proposed classification algorithm.
2.1. Related work

The problem of classifying time series has been significantly tackled in the past decade, as it was still considered one of the most challenging open tasks in 2006, [22]. It is out of the scope of this paper to provide an exhaustive survey on all the algorithms presented in the literature (for further details, please refer to [11, 23]). However, it is worth to mention that few main algorithms have proved to be the state of the art for the considered problem:

- **Distance-based methods**: distance based algorithms like Dynamic Time Warping (DTW), jointly combined with Nearest Neighbors (NN), have proved to be extremely difficult to beat [24], though the computational complexity is significant for long time series;

- **Features learning approaches**: alternatively, some works propose to construct an abstraction of the training dataset through a direct learning of the features, as done with the shapelet – Shapelet Transforms (ST) [25] – and bag-of-words – Bag of SFA Symbols (BOSS) [26] and WEASEL [27]. Unfortunately, these algorithms are not always more computationally effective, especially in the training phase, limiting the algorithm scalability to large datasets;

- **Ensemble approaches**: by combining multiple classifiers together, these algorithms have proved to achieve the highest classification accuracy. Indeed, the state-of-the-art time series classification algorithm is the Collection of Transformation Ensembles (COTE) [28] (or its hierarchical evolution, HIVET) – and bag-of-words – Bag of SFA Symbols (BOSS) – and WEASEL [27]. Unfortunately, these algorithms are not always more computationally effective, especially in the training phase, limiting the algorithm scalability to large datasets;

- **Decision tree approaches**: more computationally efficient algorithms are based on decision trees [32], though to a faster learning process corresponds a lower accuracy for certain algorithms, which is improved in the ensemble version proposed in [33];

- **Artificial neural networks approaches**: lately, the deep-learning community has started investigating the time series classification problem by means of several deep neural-networks, as widely illustrated in [29]. Although the numerous works showing remarkable performance, deep-learning is still prone to overfitting, as pointed out in [33], and research is attempting to limit this issue.

At this point, the reader might wonder whether there is actually the need of an additional method for classifying time series and the advantages of the proposed approach with respect to the state of the art:

- first of all, in many Internet of Things (IoT) applications, devices record multiple signals at the same time. Multivariate classification is a more challenging problem than univariate one, and many of the aforementioned algorithms cannot handle this issue. In fact, analyzing the different streams of data separately might limit the classification performance as the correlation between signals is not considered. In the proposed algorithm, thanks to the homomorphic properties of cepstrum (extensively covered later in this section), the proposed algorithm manages the classification with a multivariate stream of data;

- furthermore, on the contrary of ensemble methods and deep-learning, the predicted output of the proposed approach can be easily understood thanks to the intuitive formulation in the frequency domain. Thus, given comparable classification performance, it is worth investigating algorithms with a more intuitive decision process that could help explaining a given output also to non-data scientists;

- lastly, the computational burden of the state of the art algorithms is known to be an open research problem, whose solution would help spreading the use of time series classification also on devices with limited computational power (e.g., smartphones, wearable devices) or energy storage (e.g., off-grid embedded devices). As stated by the authors of COTE in [11], “an algorithm that is faster than COTE but not significantly less accurate would be a genuine advance in the field”. Instead, **fierClass** proposes a computational effective formulation that reduces the requested effort in the classification problem.

In this context, our method attempts to empty this gap. By exploiting the properties of cepstrum, **fierClass** classifies multivariate time series, requiring a lower computational effort than most of the aforementioned methods. Besides, its direct connection to the spectrum makes the output easily interpretable and provides insights about the involved dynamics.

2.2. What is cepstrum?

Cepstrum was first introduced in [18] as “the power spectrum of the logarithm of the power spectrum”, coining this new word by reversing the first syllable of spectrum [34]. It was initially proposed as a better alternative to the autocorrelation function, especially in presence of echo delays (e.g., echoes in seismological data). Later, the original definition of cepstrum was reformulated as power cepstrum, making a distinction with the complex one, differently computed:

\[ 	ext{Cepstrum}(x) = \text{LogPowerSpectrum}(\text{LogPowerSpectrum}(x)) \]

In this paper, only the power cepstrum is analyzed and the term “cepstrum” is sometimes used in place of “power cepstrum”. 

3
Given a stationary stochastic process \( s(k) \) and its spectrum \( \Phi_s \), the power cepstrum is defined as the inverse Fourier transform of the logarithm computed on the spectrum \( \Phi_s \) [35], as

\[
c_s(k) = \mathcal{F}^{-1} \left( \log \left( \Phi_s \right) \right) = \frac{1}{2\pi} \int_{0}^{2\pi} \log \left( \Phi_s \left( e^{i\theta} \right) \right) e^{ik\theta} d\theta.
\] (1)

Alternatively, as illustrated in [35], the cepstrum could also be obtained given the representation of the stochastic process through a stable and minimum phase autoregressive moving average (ARMA) model. Indeed, given an ARMA(p,q) with poles \( \{\alpha_1, \ldots, \alpha_p\} \) and zeros \( \{\beta_1, \ldots, \beta_q\} \), fed with a white noise with variance \( \sigma^2 \), its power cepstrum is

\[
c_s(k) = \begin{cases} 
\log \sigma^2 & k = 0 \\
\sum_{i=1}^{p} \frac{\gamma_1}{|\alpha_i|^2} - \sum_{i=1}^{q} \frac{\beta_i}{|\alpha_i|^2} & k \neq 0
\end{cases}.
\] (2)

A main property of the cepstrum is to be a homomorphic system (see, for instance, [35] [36]). The concept of homomorphic systems was introduced by Oppenheim [37], who demonstrated these are the ones in which nonlinear relationships could be converted into linear in their transform domains.

This result can be easily explained with an example. Suppose to compute the cepstrum of a stationary stochastic process \( o(k) \) obtained by the convolution of two given stationary stochastic processes \( s_1(k) \), \( s_2(k) \). This can be expressed as

\[
o(k) = s_1(k) \ast s_2(k) = \sum_{j=-\infty}^{+\infty} s_1(k-j)s_2(j).
\] (3)

Due to the convolution, the spectrum \( \Phi_o \) is formulated as the product of the two stochastic processes' spectra \( \Phi_{s_1} \) and \( \Phi_{s_2} \)

\[
\Phi_o = \Phi_{s_1} \cdot \Phi_{s_2}.
\] (4)

By applying the logarithm, [4] can be rewritten as

\[
\log \left( \Phi_o \right) = \log \left( \Phi_{s_1} \cdot \Phi_{s_2} \right) = \log \left( \Phi_{s_1} \right) + \log \left( \Phi_{s_2} \right).
\] (5)

The cepstrum coefficients of \( o(k) \) are obtained by taking the inverse Fourier transform of \( \log \left( \Phi_o \right) \), meaning

\[
c_o(k) = \mathcal{F}^{-1} \left( \log \left( \Phi_o \right) \right) = \mathcal{F}^{-1} \left( \log \left( \Phi_{s_1} \cdot \Phi_{s_2} \right) \right) = \mathcal{F}^{-1} \left( \log \left( \Phi_{s_1} \right) + \log \left( \Phi_{s_2} \right) \right)
\] (6)

\[= \mathcal{F}^{-1} \left( \log \left( \Phi_{s_1} \right) \right) + \mathcal{F}^{-1} \left( \log \left( \Phi_{s_2} \right) \right) = c_{s_1}(k) + c_{s_2}(k).
\]

This property proves that the cepstrum computed on the convoluted signal is exactly as the sum of the coefficients computed on each signal separately. Thanks to this homomorphic property, cepstrum has been used in many applications in which it is paramount to separate the source (i.e., an input signal) from the transmission path that filters it.

2.3. What is cepstrum used for?

In the literature, the cepstral analysis has already been employed to solve problems in mechanics [39], identification [38] [39], acoustics [40] [41], recognition and classification in bioengineering [42] [43] [44] and music [45] [46].

Cepstrum coefficients have been also used for time series clustering [19], the unsupervised process to group together time series based on their dynamics, by analyzing the coefficients of different instances. Numerous cepstrum-based metric have been proposed to solve this problem [19] [17]. However, the most used is the so-called Martin distance [47]. Given two time series generated by two different ARMA models \( M_1 \), \( M_2 \), the Martin distance is defined as

\[
d(M_1, M_2) = \sqrt{\sum_{k=0}^{\infty} k |c_{M_1}(k) - c_{M_2}(k)|^2},
\] (7)

in which \( c_{M_1} \), \( c_{M_2} \) are respectively the cepstral coefficients associated to the output of the models \( M_1 \) and \( M_2 \), while \( k \) is the cepstrum order. This metric has become widely popular because it is easy to calculate and reduces the computational effort with respect to other ones [19].

3. Classification algorithm

\textit{fierClass} is a multi-signal and multi-class classification algorithm. Its goal is to classify a stream of signals by means of the cepstrum, comparing the coefficients with the ones learned from a training data set. At the best of the authors’ knowledge, it is the first time that cepstrum is used to classify multivariate time series.

Thanks to the homomorphic property of cepstrum, in the proposed contribution the classification is obtained monitoring all the recorded signals at the same time, automatically extracting the informative content out of the global stream of data. With \textit{fierClass}, there is no need to design any set of handcrafted features, which could be time-consuming and could potentially bias the learning process, nor to go through the computationally heavy process of the state-of-the-art algorithms.

To explain the steps needed to make this possible, in this section the classification algorithm is illustrated in both its training and classification phases.
3.1. Training

Goal of the training phase is to create a database of known dynamics that are used for comparison during the classification. The algorithm is supposed to be trained with a data set \( \mathcal{D} \) of \( N_i \) input-output tuples

\[
\mathcal{D} = \left\{ \{y_i, x_i\}, \; i = 1, \ldots, N_i \right\},
\]

in which \( x_i \) and \( y_i \) are the \( i \)-th input matrix and output scalar, respectively. Each matrix \( x_i \in \mathbb{R}^{n \times T} \) (with \( i = 1, \ldots, N_i \)) is termed instance and is composed of \( n \) time series (also called input signals) of length \( T \), as in

\[
x_i = \begin{bmatrix} x_{i,1} \\ \vdots \\ x_{i,n} \end{bmatrix},
\]

in which \( x_{i,j} \in \mathbb{R}^{1 \times T} \), with \( j = 1, \ldots, n \). Instead, \( y_i \in \mathbb{Z} \), with \( i = 1, \ldots, N_i \), is a numeric flag indicating the mode (or class) that the training example belongs. The number of tuples \( N_i \), as well as the dimension of their elements, can be freely chosen, but they cannot be changed once the algorithm is trained for a given application. No restrictions are enforced on the number of modes nor on the number of inputs for each mode. Furthermore, all the time series are assumed to be sampled at the same sample time \( T_s \).

To capture the dynamics of training instances, their cepstrum coefficients are computed. To this end, given a generic \( x_i \), the spectrum related to its \( n \) time series is first computed and regularized on a sliding window \( w \) (\( w \leq T \)), expressed in seconds as \( \alpha = w T_s \). The first step is obtained by means of the Fast Fourier Transform (FFT)

\[
\Phi_{x_{i,j}}(t) = \frac{1}{w} \left| X_{i,j} \right|^2,
\]

in which \( X_{i,j} \) is the periodogram estimate computed with the FFT. Then, the regularized spectrum \( \Phi_{x_{i,j}} \) is computed averaging the \( T - w \) spectra evaluated slicing a given time series \( x_{i,j} \)

\[
\Phi_{x_{i,j}} = \frac{1}{T - w} \sum_{t=w}^{T} \Phi_{x_{i,j}}(t),
\]

in which \( \Phi_{x_{i,j}}(t) \) is the spectrum of the time series computed on the portion of signal \( x_{i,j}(t - w + 1, \ldots, t) \). The size of the sliding window \( w \) is a tuning parameter of the algorithm and also defines the dimension of the data buffer needed in the classification phase. Further details on how to calibrate this parameter are provided later on.

Cepstrum coefficients \( c_{x_{i,j}} \) of \( x_{i,j} \) are then evaluated according to the definition as the Inverse Fast Fourier Transform (IFFT) of the logarithm of the regularized spectrum \( \Phi_{x_{i,j}} \)

\[
c_{x_{i,j}} = \text{IFFT} \left( \log \left( \Phi_{x_{i,j}} \right) \right),
\]

with \( c_{x_{i,j}} \in \mathbb{R}^{1 \times w} \).

These operations are repeated for all the signals of all the training instances forming the training database \( \mathcal{C} \)

\[
\mathcal{C} = \left\{ \{y_i, c_i\}, \; i = 1, \ldots, N_i \right\},
\]

in which \( c_i \in \mathbb{R}^{n \times w} \) is the matrix containing the learned cepstrum \( c_{x_{i,j}} \) (with \( j = 1, \ldots, n \)) of the \( N_i \) instances. A sketch of the training algorithm is described in Algorithm 1.

**Algorithm 1 Training**

```plaintext
1: function fierClassTrain(\( \mathcal{D}, w \))
2: Initialize \( \mathcal{C} \)
3: for all \( x_i \in \mathcal{D} \) do
4: for all \( x_{i,j} \in x_i \) do
5: iterations ← 0
6: for \( t \leftarrow w, T \) do
7: \( \Phi_{x_{i,j}} \leftarrow \Phi_{x_{i,j}} + \frac{1}{T} \left| \text{FFT} \left( x_{i,j}(t - w, \ldots, t) \right) \right|^2 \)
8: iterations ← iterations + 1
9: end for
10: \( \Phi_{x_{i,j}} \leftarrow \Phi_{x_{i,j}} / \text{iterations} \)
11: \( c_{x_{i,j}} \leftarrow \text{IFFT} \left( \log \left( \Phi_{x_{i,j}} \right) \right) \)
12: end for
13: end for
14: return \( \mathcal{C} \)
15: end function
```

Although the focus of the proposed approach is also to provide a more computationally efficient algorithm, an evaluation of the computational burden is discussed. Computationally, FFT and IFFT are \( O(N \log N) \) (with \( N \) the number of samples – in this case \( N = w \)). Thus, the proposed learning algorithm has a complexity of

\[
O = nN_i(T - w + 1)w \log w.
\]

The computational burden of the training phase is not negligible, but it is only a linear function of the number of instances (\( N_i \)), the number of time series provided (\( n \)), their duration (\( T - w + 1 \)), and the computing cost of the FFT/IFFT (\( w \log w \)).

3.2. Classification

Once the learning phase is completed, fierClass can be employed for classifying a stream of data \( s \in \mathbb{R}^{n \times w} \),

\[
s(t) = \begin{bmatrix} s_1(t) \\ \vdots \\ s_n(t) \end{bmatrix},
\]

with the same number of time series \( n \) and window size \( w \) of the learning phase. To predict the output class, at time instant \( t \), the cepstrum coefficients of signal \( s_j \) (with \( j = 1, \ldots, n \)) are computed as in

\[
c_{s_j}(t) = \text{IFFT} \left( \log \left( \Phi_{s_j}(t) \right) \right),
\]

with \( \Phi_{s_j} \) being the cepstrum of the signal \( s_j \).
in which $\Phi_s (t)$ is the spectrum computed on the last $w$ buffered samples of signal $s_j$ or $s_j (t - w + 1, \ldots, t)$ as shown in the example in Fig. 1.

The computed cepstrum is then compared with that of all the instances $c_i$, with $i = 1, \ldots, N_t$, learned by means of a modified version of (7)

$$d(c_i, s) = \sqrt{\frac{1}{N_{cep}} \sum_{k=0}^{N_{cep}} |k|r(k)|^2}$$

$$= \sqrt{\frac{1}{N_{cep}} \sum_{k=0}^{N_{cep}} \sum_{j=1}^{N_t} c_{x_{i,j}}(k) - c_i(k)}^2.$$

The distance in (17) differs from the original in (7) for two main characteristics:

- first of all, $r(k) = \sum_{j=1}^{N_t} c_{x_{i,j}}(k) - c_i(k)$ is the sum of the mismatch between the learned and the current cepstrum for all the $n$ input signals;
- second, the cepstral coefficients vector is not infinitive long and, more importantly, decays to zero rapidly. This result was proved also in [48]. For this reason, computing the Martin distance on infinite terms (or even with a great number of them) is impractical or results may be unacceptable. In this paper, we propose to evaluate the metric only on the first $N_{cep}$ coefficients, in which $N_{cep}$ is chosen as a tuning parameter;

Based on the computed distances, the predicted class $\hat{y}$ is the one minimizing the distance in (17):

$$\hat{y} = \arg\min_{c_i \in C} d(c_i, s).$$

The classification phase of the algorithm is sketched in Algorithm 2 and summarized in the flowchart in Fig. 2.

In this case, at each new sample, the computational complexity is $O = nw \log(w)$. Assuming to use fierClass for classifying a set of time series of length $T_{validation}$ (instead of a continuous stream of data), the overall computational complexity becomes

$$O = n(T_{validation} - w + 1)w \log(w).$$

**Algorithm 2** Classifying

1: function fierclassifying($C, s, w, N_{cep}$)
2: Initialize $\hat{y}$, $r$, $d$, $d_{min}$
3: for all $s_j \in s$ do ▷ Compute the cepstrum of stream $S$
4: $\Phi_s \leftarrow \frac{1}{n} \cdot |FFT(s_j(t-w+1, \ldots, t))|^2$
5: $c_s \leftarrow \text{IFFT} (\log (\Phi_s))$
6: end for
7: for all $c_i \in C$ do ▷ Compute the distance
8: $d \leftarrow 0$
9: $r \leftarrow 0$
10: for $k \leftarrow 0, N_{cep}$ do
11: for all $x_{i,j} \in x_i$ do
12: $r \leftarrow r + (c_{x_{i,j}}(k) - c_i(k))$
13: end for
14: $d \leftarrow d + k \cdot |r|^2$
15: end for
16: $d \leftarrow \sqrt{d}$
17: if $d < d_{min}$ then ▷ Update predicted class
18: $d_{min} = d$
19: $\hat{y} = y_i$
20: end if
21: end for
22: return $\hat{y}$
23: end function

**Remark 3.1.** Finally, a remark on the tuning parameters is addressed aiming to provide useful guidelines for the manual tuning of the algorithm (Table 1). The rule of thumb for the algorithm calibration can be summarized as follows:

- the size of the window $w$ affects the spectrum representation. In fact for large values of $w$, the spectral representation is fine, but due to the significant amount of past data stored, transient duration increases; on the contrary, short windows provide a prompter reaction when the class changes, but the spectrum is described with fewer points, which affect the information accuracy in the frequency domain;
- the value of $N_{cep}$ represents the maximum cepstrum order that is accounted in the modified Martin distance. Theoretically, this value should be infinity to perfectly characterize the system dynamics. However, as widely known for the autocovariance function, high-order cepstrum coefficients are poorly estimated due to the limited amount of data available for large temporal lags. Thus, the modified Martin distance is truncated at the $N_{cep}$-th order, reducing the effect of the noise affecting the estimated coefficients on the classification process.
According to these guidelines, in the application example we proceeded as follows: given an initial guess for \( N_{\text{cep}} \), the window \( \alpha \) is tuned trading-off system responsiveness and buffer size, trying to obtain a reactive classification and stable output at steady state. Once the window is set to its optimal value \( \alpha^o \), a fine tuning of \( N_{\text{cep}} \) is performed, leading to the performance maximization value of \( N_{\text{cep}}^o \). As illustrated in Fig. 11 and Fig. 12, classification performance shows a significant trend for different values of \( w \), which reaches its peak for \( \alpha^o = 2 \) seconds. Instead, given a certain value of \( \alpha \), performance settle once the optimal frequency window is found (in our application \( N_{\text{cep}} \geq 4 \), in which the peak is reached for \( N_{\text{cep}}^o = 5 \)).

It is also worth to remark that \( N_{\text{cep}} \) does not show any significant trend when the number of input signals is sufficiently large (in the proposed example, when at least five inputs are used). In fact, when only few signals are involved, the effect of noise on the performance is significant for different values of \( N_{\text{cep}} \) (up to 20 percentage points in terms of accuracy), making the overall calibration process more sensitive to the parameters tuning, increasing the required human effort and vanishing some of the benefits of the proposed approach.

4. Experimental results

To validate the proposed approach, fierClass is tested in order to detect whether a driver is using the phone while driving based only on the sensed motion, a classification problem already analyzed in a previous work of the authors [49]. The algorithm is automatically activated and deactivated when the driver is inside the vehicle, thanks to a Bluetooth beacon placed under the steering wheel (as shown in Fig. 3). Movements are sensed through the smartphone’s tri-axial inertial measurement unit (IMU) measuring both the accelerations and the angular rate (thus, \( n = 6 \)); signals are recorded at the sampling frequency of 120 Hz. The classification process is run at the same frequency, obtaining one predicted output at each new sample.

The smartphone is considered in-use when the driver performs one of the most common activities (e.g., handling, texting, scrolling, browsing, calling etc.) and not in-use otherwise, as illustrated in Fig. 4. fierClass is trained and tested against data collected during an experimental campaign in the Milan area, driving the vehicle in both a mixed urban and high-way environment.

4.1. Validation

In this subsection, a sensitivity analysis of the algorithm performance with respect to the tuning parameters \( w \) and \( N_{\text{cep}} \) is addressed. Besides, to validate the effectiveness of the proposed approach, the analysis is conducted for all the possible signals combinations, leading to more than 2500 tests. First of all, the classification behavior is analyzed computing the accuracy for all the possible combinations of

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Unit</th>
<th>Tuning Guideline</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \alpha )</td>
<td>Sliding window length</td>
<td>[s]</td>
<td>[1, 10]</td>
</tr>
<tr>
<td>( N_{\text{cep}} )</td>
<td>Maximum cepstrum order</td>
<td>[-]</td>
<td>[2, 20]</td>
</tr>
</tbody>
</table>

Table 1: Classification algorithm tuning parameters.
acceleration and angular rate components (2) (Fig. 5), ordering the obtained results based on their mean value, highlighting the maximum and minimum values not considered outliers. As shown, a trend is evident:

- almost half of the combinations (the left hand side of Fig. 5 until $\alpha_i\omega_{xyz}$) lead to poor performance ($\approx 75\%$), with minimum differences in the mean value. Besides, their distributions are confined in a narrow range of their mean value, meaning that the algorithm is insensitive to the calibration, even for a significant amount of signals used. This is because the information content of these combinations is minimal, vanishing the benefits of combining more inputs;

- instead, the average accuracy of the remaining combinations increases more significantly (up to 94.93%). The center portion of the listed combinations is more affected by the tuning parameters, producing a more spread range. Alternately, with more signals convoluted, the ranges shrink, obtaining comparable performance for different calibrations. Thus, with more signals, the algorithm achieves the highest return, minimizing the bias introduced by the algorithm manual tuning.

To better analyze these results, starting from the most performing classifier trained with a single signal ($a_{y}$), an evolving combination of signals that improve the overall classification accuracy is found, as shown in Fig. 6. In this case, the benefits of the convolution with more signals proposed in fierClass become evident: the average accuracy increases, especially for more than four input signals; the maximum and minimum values increases, improving the performance in both the best and worst case scenario; consequently, the ranges shrink, making the algorithm less sensitive to its tuning.

Accuracy is an important performance index, but it is not sufficient to describe the classification behavior entirely. For this reason, fierClass is also analyzed in terms of sensitivity and specificity, through a scatter version of the Receiver Operating Characteristic (ROC). The goal of this visualization is to jointly analyze the true positive rate and the false positive rate, showing how quickly the number of false positives (i.e., in this application a false positive is obtained when the algorithm predicts the class Using, though the phone is actually not in use) grows for an increasing number of correct claims [50]. As illustrated in Fig. 7 for an increasing number of signals, the classification qualitatively improves, according to the location of the main clusters. The variability remarks the drawbacks of an improper calibration.

To quantify the variability of the different ROC curves, the so-called Area Under Curve (AUC), which is the area below the ROC [51], is employed. In Fig. 8, the AUC is computed, limiting the analysis to the only combination
Figure 5: Sensitivity analysis of classification accuracy with respect to the tuning parameters for different signals combinations, ordered by the average accuracy (highlighted by means of the markers). As shown, the performance increase for more signals used, reaching the highest using all the available signals. Furthermore, tuning influences the classification less significantly when all the information obtained combining the signals is retained.

Figure 6: An overview of the sensitivity analysis, starting from the most performing single signal and an evolving combination of it. fierClass outperforms with more signals used, in terms of maximum, minimum, and average accuracy.

Figure 7: ROC curve of all the trained algorithms (top) and a zoom in of the most performing one (bottom). The highest true positive rate is achieved for the minimum false positive rate (i.e., top left corner) for an increasing number of inputs (6, then 5, and so on).
of input signals analyzed in Fig. 6. As already discussed, up to three inputs, the performance of the classifiers are very sensitive to the tuning parameters; contrarily, for four signals or more, the algorithm outperforms and then settles in a narrow range of its best. Once more, fiertClass benefits combining more signals together.

![Figure 8: Distribution of AUC for an increasing number of feature combinations. The proposed approach reaches the highest performance when more inputs are used.](image)

The analysis of accuracy and AUC might be misleading when the size of the validation classes is unbalanced [52]. Since tests were conducted reproducing a realistic driving experience, Fig. 9 shows that there is a mismatch between the classes Using and Not using. For this reason, two more performing indexes are analyzed: F1-score, obtained as harmonic average of precision and recall [50], and the Cohen’s Kappa, which compares the observed accuracy with the expected one [53]. As shown in both Fig. 10 and Fig. 11 the performance of the classifier improves significantly when more inputs are used (more than three, also from this analysis). Although the classes are unbalanced, the two indexes confirm what previously state about the overall classification performance.

![Figure 9: During tests, the phone was used only 27% of the time, leading to unbalanced classes.](image)

To conclude the analysis, the influence of each tuning parameter on the classification performance is addressed.

The cepstrum order, \( N_{cep} \), does not heavily influence the classification outcome when all the input signals are used, as shown in Fig. 12. However, reducing the number of signals, fiertClass becomes more sensitive to the cepstrum order, showing a peak at \( N_{cep} = 4 \). When less information is available, the classifier tends to overfit more and be more influenced to the noise corrupting the estimate of higher order cepstrum coefficients.

![Figure 10: Analysis of the F1-score for the different input signals combinations. The index grows significantly and its distribution contracts for more than three inputs.](image)

The sliding window’s dimension also influences the final outcome (Fig. 13): for a short window (\( \alpha < 2 \) seconds), the algorithm has a worse representation of low-frequency components, while for longer windows (\( \alpha \geq 5 \) seconds) the buffer takes more time to empty, influencing the classification for several seconds after a transition. This trend is even more stressed when the number of signals grows, since the information combined from sources with different dynamics is either partially captured or the promptness...
Figure 12: Accuracy vs cepstrum order. Top: the classification performance shows a trend, reaching the peak for $N_{cep} = 4$. Bottom: with more input signals, the classification performance grows and then settles for higher cepstrum order.

Figure 13: Accuracy vs window length. The effect of a different bufferization influences both the granularity in the frequency domain, as well as the time needed to fill/empty the buffer after a transition. The tuning parameter shows a maximum in the classification accuracy for $\alpha = 2$ seconds, especially when the number of input signals increases.

Given these analysis, it is possible to state that the optimal performance is achieved for $\alpha^o = 2$ s and $N_{cep}^o = 5$.

4.2. Testing

After learning the algorithm hyperparameters, the algorithm is tested against a new set of data. The testing datasets are composed of two new test drives of 5 minutes each, which are collected for testing and are not used in the validation phase. Table 2 reports the results of the analyzed indexes (i.e., accuracy, area under curve, F1-score and Cohen’s Kappa) for the most performing classifier and, using the same tuning, for the two tests separately and their average results.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Accuracy [%]</th>
<th>AUC [-]</th>
<th>F1 Score [-]</th>
<th>Cohen’s Kappa [-]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validation</td>
<td>96.94</td>
<td>0.958</td>
<td>0.944</td>
<td>0.923</td>
</tr>
<tr>
<td>Testing 1</td>
<td>95.34</td>
<td>0.919</td>
<td>0.869</td>
<td>0.84</td>
</tr>
<tr>
<td>Testing 2</td>
<td>94.64</td>
<td>0.926</td>
<td>0.873</td>
<td>0.839</td>
</tr>
<tr>
<td>Testing average</td>
<td>94.99</td>
<td>0.922</td>
<td>0.872</td>
<td>0.84</td>
</tr>
</tbody>
</table>

Table 2: Classification performance for tuning $\alpha^o = 2$ s and $N_{cep}^o = 5$, using all the six input signals available.

Notably, the performance obtained in the testing phase align with those obtained in validation. In this case, as illustrated, the performance computed on the test set showed to lose two percentage points in terms of accuracy and three in the area under curve. The mismatch is more noticeable for the F1-score and Cohen’s Kappa, but the obtained performance can still be considered acceptable.

5. Concluding Remarks

This work presented a novel frequency-based classification approach for time series which can handle a multisignal input, as in practice is the case of measurements coming from different sensors. Specifically, our proposal leverages the computation of the cepstrum coefficients of each input signal, combining them in the frequency domain in a way that corresponds to signal convolution in the time domain. This operation merges the information content of the different inputs while reducing the input dimension. The classification is then computed in the cepstrum domain, convolving a multivariate stream of data and comparing it with respect to the previously learned instances.

The proposed approach, which yields the distinctive advantage of removing the subjective feature engineering step inherent in many time-based classification methods, proved to offer very good performance when tested in an experimental setting of classifying whether a car driver is using the phone while driving. On the current setting, the proposed approach can yield very robust and effective responses. The favorable results were obtained with significantly less computational effort than current state of the art algorithms, making the proposed approach feasible for its use on devices with limited resources (e.g., in our case, a smartphone). Besides, the proposed approach makes the analysis of the output easily accessible and interpretable.

Unfortunately, the benefits of this approach come with some limitations: the algorithm is calibrated over two hyperparameters, which have proved to significantly empha-
size the information content in the frequency domain when properly tuned, but they require some manual calibration; besides, in further developments the problem of managing measurements collected with different level of accuracy and reliability, a problem of extreme importance in many applications and not yet solved by the time series classification research community, will be tackled.

References


