2016 23rd International Conference on Pattern Recognition (ICPR)
Cancún Center, Cancún, México, December 4-8, 2016

Aggregation procedure of Gaussian Mixture Models for additive features

Antonio Ridi*†, Christophe Gisler*†, Jean Hennebert*†
*University of Applied Sciences Western Switzerland,
School of Engineering and Architecture of Fribourg, iCoSys Institute
{antonio.ridi, christophe.gisler, jean.hennebert}@hefr.ch
†University of Fribourg, Department of Informatics, Fribourg, Switzerland
{antonio.ridi, christophe.gisler, jean.hennebert}@unifr.ch

***Abstract*—In this work we provide details on a new and effective approach able to generate Gaussian Mixture Models (GMMs) for the classification of aggregated time series. More specifically, our procedure can be applied to time series that are aggregated together by adding their features. The procedure takes advantage of the additive property of the Gaussians that complies with the additive property of the features. Our goal is to classify aggregated time series, i.e. we aim to identify the classes of the single time series contributing to the total. The standard approach consists in training the models using the combination of several time series coming from different classes. However, this has the drawback of being a very slow operation given the amount of data. The proposed approach, called *GMMs aggregation procedure*, addresses this problem. It consists of three steps: (i) modeling the independent classes, (ii) generation of the models for the class combinations and (iii) simplification of the generated models. We show the effectiveness of our approach by using time series in the context of electrical appliance consumption, where the time series are aggregated by adding the active and reactive power. Finally, we compare the proposed approach with the standard procedure.**

I. Introduction

A time series is usually defined as a collection of samples equally spaced in time coming from a system observed for a certain amount of time. Time series are popular in several fields, as statistics, economics and environmental science. More formally, a time series is defined as $\mathcal{X} = \{\mathbf{x}_t : t \in \tau\}$ where $\tau = \{1, 2, \ldots, T\}$ is the index set.

Sometimes time series can be aggregated together. More specifically, with the term "aggregated", we generically refer on the way of combining two or more independent time series. In this work we focus on time series that are aggregated together by adding their features. For sake of simplicity, in this paper we will refer to them as "additive time series". Several fields are concerned, as economics [1], electrical appliance recognitions [2] or urban traffic [3]. Considering a set of N time series $\{\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_N\}$, we can define the additive time series as $\mathcal{Y} = \{\mathbf{y}_t : t \in \tau\}$, where:

$$\mathbf{y}_t = \sum_{n=1}^{N} \mathbf{x}_{n,t} = \mathbf{x}_{1,t} + \mathbf{x}_{2,t} + \cdots + \mathbf{x}_{N,t} \tag{1}$$

In this work, we focus on the classification of the additive times series. Supposing that each independent time series $\mathcal{X}$ belongs to a specific class, the aim of the classification of the additive times series $\mathcal{Y}$ is to retrieve the list of classes that are contributing to the total. This task could also be seen as a multi-labeling problem, given that $\mathcal{Y}$ has to be associated with multiple target labels. The difficulty of the task is related to the number of combinations that can be generated.

Suppose to have a set of independent time series divided in M classes, being P the average number of time series per class. The aim is to build the models of all the class combinations for being able to identify additive time series. The standard approach for classifying the additive time series consists in training the models using all the combinations of time series. The number of different combinations depends on the aggregation level L, i.e. the number of time series aggregated together. We can estimate the number of combinations through the following formula:

$$c_s = \binom{M}{L} P^L = \frac{(M)!\ P^L}{(M-L)!\ L!} \tag{2}$$

where the binomial coefficient represents the number of models and the exponential represents the number of combinations per model. The number of combinations c_s exponentially grows when increasing the aggregation level L and, in few steps, it requires a high computational time.

In this work, we propose a different approach able to generate the models of the additive time series without generating all the time series combinations. In particular, we use Gaussian Mixture Models (GMMs) as machine learning technique for building these models. We take advantage of the additive property of the Gaussians that complies with the additive property of the features of the additive time series. Our approach consists in three steps: (i) modeling the independent classes, (ii) the generation of the models for the class combinations and (iii) the simplification of the generated models.

This papers is organized as follows: in Section II we present the related works, in Section III we introduce the GMMs and in Section IV we show the proposed procedure. In Section V we analyze a case of study in the context of the electrical appliance identification and in Section VI we discuss the results. Finally in Section VII we conclude the paper.

978-1-5090-4846-5/16/$31.00 ©2016 IEEE

II. Related works

Generally speaking, among the most common approaches for the time series classification we find Dynamic Time Warping (DTW), GMMs and Hidden Markov Models (HMMs). DTW determines a measure of the similarity between the time series in the test set with those in the training set, by warping them non-linearly in the time dimension. A very different approach is used by GMMs and HMMs, that use stochastic probabilistic techniques for estimating the probability density distribution of the features. Differently from the GMMs, HMMs include the notion of state and try to capture the state sequence that can be implicitly contained in time series. The strength of these classification techniques is the ability to cope well with the intrinsic characteristics of time series, in fact (i) they are not particularly sensitive to the total amount of noise, (ii) they scale well with the length of time series and (iii) the difference in lengths between time series is not problematic. Other machine learning classification techniques, as Neural Networks, are much more sensitive to the intrinsic characteristics of time series [4].

As previously said, in this work we address the classification of additive time series, i.e. when two or more time series are aggregated by adding their features. The classification task consists in retrieving the classes of the time series combined together. This problem can also be addressed from a multi-labeling classification perspective. In fact, each aggregated time series is associated with multiple target labels. The multi-label classification problem has two approaches: (i) transformation, which transforms multi-label classification problem into single label classification problem and (ii) algorithm adaptation, which adapts an existing single label classification algorithm to handle multi-label data. While the first method is more general and can be applied to several classifiers, the other applies only to specific classifiers [5]. To best of our knowledge, the multi-label classification applied to aggregated time series has not been formalized so far.

Some works deal with a different but related problem, i.e. the "disaggregation" of the aggregated time series. Disaggregation algorithms are usually defined as those able to separate the single contributions in the whole signal. Instead of finding the labels of the single time series, all the time series are computed. This task is clearly more difficult than the multi-label classification, given that at each time step a decision on the level of the contributions is taken. The disaggregation task has two main approaches, i.e. (i) optimization and (ii) pattern recognition. The optimization method combines different time series for finding the most similar to those observed. The main drawback is that only the known time series are combined and if an unknown time series has been aggregated, important effects could derive on the identification rate. Instead, the pattern recognition approach identifies specific events able to identify specific time series [2]. In this approach the steady-states and transient parts of the time series are analyzed. The steady-state has been defined as *a difference between any two samples of a sequence that does not exceed a given tolerance value* [6]. The transient analysis searches for relevant changes in the aggregated time series when a transition is verified and analyzes the portions of the signal considered unstable.

In the appliance recognition field, the Non Intrusive Load Monitoring (NILM) measures the consumption of the whole home by using smart meters. The aim is to "disaggregate" the total energy consumption for separating the single appliance contributions. The most used steady-state technique is the Power change approach, that consists in analyzing the active and eventually the reactive power of the total power consumption [6]. As a drawback, it is hard to separate appliances having similar consumption signatures. Other approaches are based on the analysis of current, voltage and other features in time or frequency. Trajectories in the voltage-current domain or the voltage noise can also be used [2]. Other techniques use the information left by the appliances on their signature when they are turned on or off. Even if this information is usually very short in time, it can be used for identifying the appliance. The major drawback is that the transitions can be captured by using a high sampling frequency.

In this work, we propose a new approach for the additive time series classification. Our method is based on GMMs and we take advantage of the additive property of the Gaussians that complies with the additive property of the features.

III. Gaussian Mixture Models

GMMs is a stochastic machine learning technique that estimates the probability densities of the features. The probability densities are represented by multivariate Normal distributions by using a parameter vector $\theta = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$, where $\boldsymbol{\mu}$ is the mean vector and $\boldsymbol{\Sigma}$ is the covariance matrix. The probability densities are computed as a weighted sum of multivariate Normal densities, called mixtures. The likelihood of a generic sample $\mathbf{x}$ is equal to:

$$p(\mathbf{x} \mid \theta) = \sum_{k=1}^{K} \frac{w_k}{\sqrt{(2\pi)^D |\boldsymbol{\Sigma}_k|}} e^{[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)]} \quad (3)$$

where K is the number of mixtures, w_k is the weight and D the dimension. Considering the mixture case, the parameter vector θ is $\{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K, \boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_K, w_1, \ldots, w_K\}$. In addition, the sum of the weights must be equal to 1.

Mixture models are typically used for representing complex density functions and the mixture parameters are estimated during the training phase. A common procedure for training GMMs consists in (i) finding clusters in the training data, usually with a K-Means algorithm, (ii) using the centroids as initial mean values of the mixture parameters and (iii) estimating the mixture parameters, usually with the Expectation-Maximization (EM) algorithm. For estimating the mixture parameters, the EM algorithm alternates two steps until convergence:

Expectation (E-step). For every mixture k the responsibility r_{nk} of each point to each mixture is computed:

$$r_{nk} = \frac{w_k p(\mathbf{x}_n \mid \theta_k)}{\sum_k w_k p(\mathbf{x}_n \mid \theta_k)} \quad (4)$$

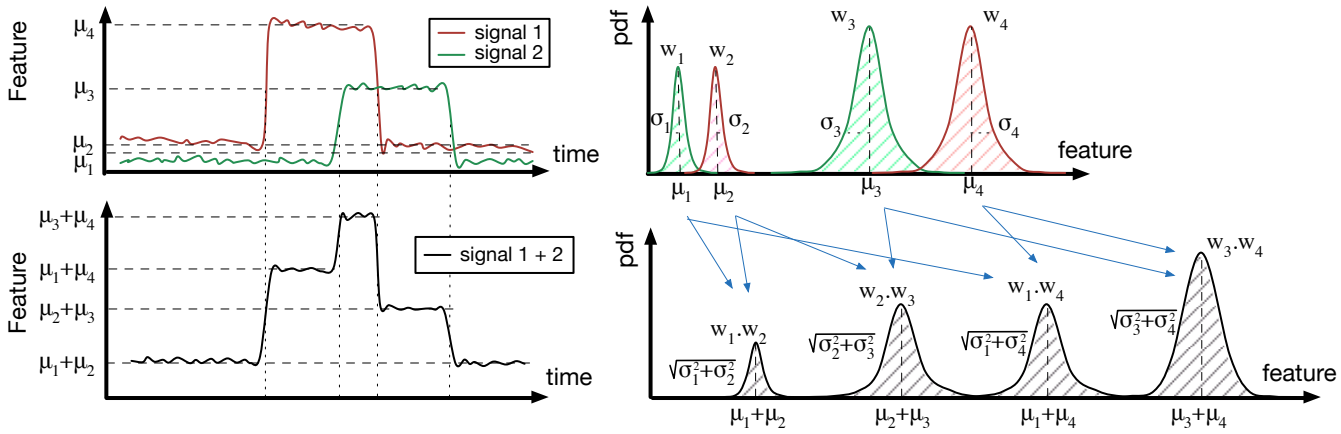

Fig. 1. Synthetic example of the model merging for the classification of additive time series.

The expectation step is divided in two steps: the computation of the cluster posterior probabilities and the evaluation of the densities.

Maximization (M-step). During the M-step, the parameters are updated. The maximum-likelihood estimation of the mean, the covariance and the mixture weights can be computed. The mean of the k-th mixture is the weighted average of all the points assigned to the mixture, while the covariance matrix is proportional to the weighted empirical scatter matrix. The weight of a mixture is the proportion of the sum of the responsibilities against the total.

In this work the hypothesis of uncorrelated features is made, as usually done in several implementations. When the covariance matrices are diagonal, the computational complexity is widely reduced. From the computational complexity perspective, the EM algorithm dominates during the GMMs training. Considering t iterations, k Gaussians, n points and d dimensions, the total complexity of the EM algorithm is equal to $O(t(kd^3+nkd^2))$ for full covariance matrices and $O(tnkd)$ for diagonal covariance matrices. The main difference is during the E-step, where we need to invert $\mathbf{\Sigma}_k$ and compute its determinant [7].

IV. GMMs aggregation procedure

We propose an approach able to generate the models of the additive time series without computing all the time series combinations. By using the GMMs, we take advantage of the additive property of the Gaussians that complies with the additive property of the features of the additive time series.

When using GMMs, every category is represented by a given number of Gaussians. For computing the combination of two or more models, we can use an interesting property of the Gaussians: if X and Y are independent random variables normally distributed, then their sum is also normally distributed [8]. The sum of two independent normally distributed random variables is normal, with its mean being the sum of the two means, and its variance being the sum of the two variances:

$$\begin{cases} X \sim \mathcal{N}(\mu_X, \sigma_X^2) \\ Y \sim \mathcal{N}(\mu_Y, \sigma_Y^2) \\ Z = X + Y \end{cases} \tag{5}$$

then:

$$Z \sim \mathcal{N}(\mu_X + \mu_Y, \sigma_X^2 + \sigma_Y^2) \tag{6}$$

The main algorithm is presented in Algorithm 1. For creating the models for all the combinations of L classes, we firstly have to compute the models of the independent classes (A), and later we can iteratively merge (B) and simplify (C) the models.

Algorithm 1 Aggregating schema

Input: data x_i, label c_i, aggregation level L
$\lambda^{new}(\boldsymbol{\mu}, \mathbf{\Sigma}, \pi) \leftarrow$ GMMs(x_i,c_i)
$\lambda^{old} \leftarrow \lambda^{new}$
for $l = 1$ **to** $L - 1$ **do**
  $\lambda^{new} \leftarrow$ merge_models(λ^{new}, λ^{old})
  $\lambda^{new} \leftarrow$ simplify_models(λ^{new})
end for

Algorithm 2 Merging models

Input: GMMs$'(\lambda')$, GMMs$''(\lambda'')$
$M \leftarrow$ total number of classes
for $m \leftarrow 1$ **to** M **do**
  $nGauss_1 \leftarrow$ total number of Gaussians for GMMs$'$
  $nGauss_2 \leftarrow$ total number of Gaussians for GMMs$''$
  $k \leftarrow 0$
  for $k_1 \leftarrow 1$ **to** $nGauss_1$ **do**
    for $k_2 \leftarrow 1$ **to** $nGauss_2$ **do**
      $\boldsymbol{\mu}_{m,k} \leftarrow \boldsymbol{\mu}'_{m,k_1} + \boldsymbol{\mu}''_{m,k_2}$
      $\mathbf{\Sigma}_{m,k} \leftarrow \mathbf{\Sigma}'_{m,k_1} + \mathbf{\Sigma}''_{m,k_2}$
      $w_{m,k} \leftarrow \pi'_{m,k_1} \cdot w''_{m,k_2}$
      $k \leftarrow k + 1$
    end for
  end for
end for
Output: new models $\lambda(\boldsymbol{\mu}, \mathbf{\Sigma}, w)$

Algorithm 3 Simplifying models - Optimal solution

Input: GMMs (λ)
$dist_{thr} \leftarrow$ set a distance threshold
$M \leftarrow$ total number of classes
for $m \leftarrow 1$ **to** M **do**
  $reduce \leftarrow true$
  while $reduce$ **do**
    $comb \leftarrow$ all the Gaussians combinations
    $C \leftarrow$ number of combinations
    for $c \leftarrow 1$ **to** C **do**
      $(k_1, k_2) \leftarrow comb(c)$
      $dist_{vect}(c) \leftarrow$ distance computation
    end for
    $dist_{min} \leftarrow$ minimum distance
    $(\hat{k}_1, \hat{k}_2) \leftarrow$ Gaussian combination corresponding to the minimum distance
    if $dist_{min} < dist_{thr}$ **then**
      $\lambda_{m,k} \leftarrow$ recompute the new parameters
      $\lambda_{m,\hat{k}_1} \leftarrow \lambda_{m,k}$
      remove $\lambda_{m,\hat{k}_2}$
    else
      $reduce \leftarrow false$
    end if
  end while
end for
Output: reduced models $\lambda(\boldsymbol{\mu}, \boldsymbol{\Sigma}, w)$

Algorithm 4 Simplifying models - Suboptimal solution

Input: GMMs (λ)
$dist_{thr} \leftarrow$ set a distance threshold
$M \leftarrow$ total number of classes
for $m \leftarrow 1$ **to** M **do**
  $\hat{k}_1 \leftarrow 1; \hat{k}_2 \leftarrow 1$
  $reduce \leftarrow true$
  $nGauss \leftarrow$ compute the number of Gaussians
  $nGauss_{old} \leftarrow nGauss$
  while $reduce$ **do**
    $search \leftarrow true$
    $k_1 \leftarrow \hat{k}_1; k_2 \leftarrow \hat{k}_2$
    while $k_1 < nGauss$ **and** $search$ **do**
      while $k_2 < nGauss$ **and** $search$ **do**
        $dist_{val} \leftarrow$ computation of the distance
        if $dist_{val} < dist_{thr}$ **then**
          $search \leftarrow false$
          $\hat{k}_1 \leftarrow k1; \hat{k}_2 \leftarrow k_2$
        end if
        $k_2 \leftarrow k_2 + 1$
      end while
      $k_1 \leftarrow k_1 + 1$
      $k_2 \leftarrow 1$
    end while
    if $search$ **then**
      if $nGauss_{old} = nGauss$ **then**
        $reduce \leftarrow false$
      else
        $\hat{k}_1 \leftarrow 1; \hat{k}_2 \leftarrow 1$
        $nGauss_{old} \leftarrow nGauss$
      end if
    else
      $\lambda_{m,k} \leftarrow$ recompute the new parameters
      $\lambda_{m,\hat{k}_1} \leftarrow \lambda_{m,k}$
      remove $\lambda_{m,\hat{k}_2}$
      $nGauss \leftarrow$ compute the number of Gaussians
    end if
  end while
end for
Output: reduced models $\lambda(\boldsymbol{\mu}, \boldsymbol{\Sigma}, w)$

A. Independent classes modeling

During this step we simply train our models by using the independent classes. The set of parameters derived from this step will be used in the next steps.

B. Model merging

The main algorithm is presented in Algorithm 2. We illustrate this step by using a synthetic example in Figure 1. We assume two different signals, one represented in green and the other in red, belonging to different classes. In the upper-right side of the figure we represent the probability density functions when using the GMMs algorithm with two Gaussians for each model. The first model, in green, has two mixtures (μ_1, σ_1^2, w_1) and (μ_3, σ_3^2, w_3). The second model, in red, has also two mixtures (μ_2, σ_2^2, w_2) and (μ_4, σ_4^2, w_4). Our aim is to find the model representing the combination of the two. The model merging consists in using the property of the sum of two independent normally distributed random variables. Starting with the two models, we can combine them in order to obtain the model representing the combination of the two. Every mixture of every model has to be added to every mixture of the other model. The weight of the resulting mixture is the product of the weights of the original mixtures. The combined model therefore has four mixtures ($\mu_1 + \mu_2$, $\sigma_1^2 + \sigma_2^2$, $w_1 \cdot w_2$), ($\mu_2 + \mu_3$, $\sigma_2^2 + \sigma_3^2$, $w_2 \cdot w_3$), ($\mu_1 + \mu_4$, $\sigma_1^2 + \sigma_4^2$, $w_1 \cdot w_4$) and ($\mu_3 + \mu_4$, $\sigma_3^2 + \sigma_4^2$, $w_3 \cdot w_4$).

C. Model simplification

One of the main problem of the proposed method is that the number of Gaussians quadratically grows when increasing the agglomeration level L. However, very often Gaussians close together are created. In this step, our aim is to reduce the number of Gaussians by merging the closest ones. We separate two phases: (i) the computation of the distances between Gaussians and (ii) the re-computation of the new parameters. The computation of the distance is performed by using the Kullback-Leibler (KL) distance, through the following formula:

$$dist = log(\frac{|\boldsymbol{\Sigma}_{m,k_2}|}{|\boldsymbol{\Sigma}_{m,k_1}|}) + \boldsymbol{\Sigma}_{m,k_1}\boldsymbol{\Sigma}_{m,k_2}^{-1} - D + (\boldsymbol{\mu}_{m,k_2} - \boldsymbol{\mu}_{m,k1})^T \boldsymbol{\Sigma}_{m,k_1}^{-1}(\boldsymbol{\mu}_{m,k_2} - \boldsymbol{\mu}_{m,k_1}) \quad (7)$$

We compare this distances with a threshold value ($dist_{thr}$). If the computed distance is below the threshold, than the Gaussians are considered "similar" and can be merged together. The

re-computation is computed through the following formula:

$$\begin{cases} w_{m,k} = w_{m,k_1} + w_{m,k_2} \\ \boldsymbol{\mu}_{m,k} = f_1 \boldsymbol{\mu}_{m,k_1} + f_2 \boldsymbol{\mu}_{m,k_2} \\ \boldsymbol{\Sigma}_{m,k} = f_1 \boldsymbol{\Sigma}_{m,k_1} + f_2 \boldsymbol{\Sigma}_{m,k_2} + f_1 f_2 \cdot \\ \quad (\boldsymbol{\mu}_{m,k_1} - \boldsymbol{\mu}_{m,k_2})(\boldsymbol{\mu}_{m,k_1} - \boldsymbol{\mu}_{m,k_2})^T \end{cases} \tag{8}$$

where:

$$\begin{cases} f_1 = \dfrac{w_{m,k_1}}{w_{m,k_1} + w_{m,k_2}} \\ f_2 = \dfrac{w_{m,k_2}}{w_{m,k_1} + w_{m,k_2}} \end{cases} \tag{9}$$

We develop two different methods for the Gaussian merging, as respectively shown in Algorithm 3 and 4:

- *Optimal solution*. It consists in computing all the combinations of all the Gaussians and merging the two Gaussians with the minimum distance if below the threshold. In practice in this case we compute all the distances between the Gaussians and we select the best one.
- *Suboptimal solution*. In some cases the previous method could be quite long, especially when having several Gaussians. This approach consists in analyzing all the combinations but greedily selecting the first occurrence that is below the threshold. When two Gaussians are merged, the search continues from the point in which it has been stopped.

In both cases, we repeat the procedure until the distance between the two closest Gaussians is over the threshold.

V. Case of study: electrical appliance identification

As previously said, our method can be used when additive time series are present. We present here a case of study in the appliance recognition field, where the electrical consumption of appliances is usually measured in term of active and reactive power. Given that the appliances are plugged in parallel on the network, the time series have the additive property, meaning that the electrical consumption of the total is equal to the sum of the single contributions. One of the principal applications is the Non-Intrusive Load Monitoring (NILM) or multi-signal detection, i.e. when the measuring system, called smart meter, is placed outside the house. By the nature of the problem, the smart meter acquires the whole consumption of the house, where all the appliance consumptions are added together. The principal goal of the analysis is to detect which appliances contribute to the total.

A. Database

Given that our method requires to build the models of the independent classes (the appliances in this case), we use a database where the appliances are recorded separately. In a second step we artificially add together the independent time series for generating the additive time series. We use ACS-F2 database [9] as reference database for our analyses. This database contains the electrical appliance consumption of 225 different appliances uniformly spread among 15 categories. The appliances are acquired two times, in the so-called *sessions*, by using a low sampling frequency ($10^{-}1$ Hz). We use the first session as training set and the second session as test set. We compare our method with a standard GMMs approach, where the models of the combinations are trained from the combinations of the time series. The training procedure is very different for the two approaches:

- *Standard GMMs*. We compute all the combinations of all the time series and we train the models using these combinations.
- *GMMs aggregation procedure*. We compute the models for the independent classes and we merge them using the proposed approach.

The test set is the same for both the methods, consisting in all the combinations of all the time series in the ACS-F2 test set. For a matter of computational time of the standard approach, we limit the aggregation level to 2, having 105 different models to train.

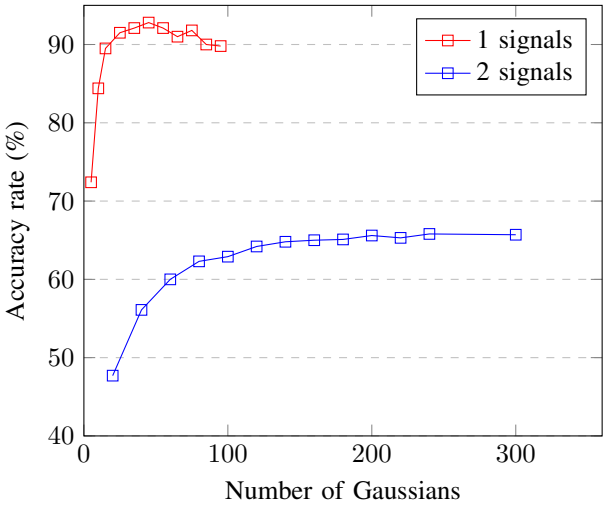

Fig. 2. Trend of the accuracy rate when increasing the number of Gaussians. In red the independent classes, while in blue the combination of 2 signals.

B. GMMs aggregation procedure

The first step of our procedure consists in computing the models of the 15 original categories. As illustrated in Figure 2, the red line shows the trend of the accuracy when increasing the number of Gaussians. The best case is obtained when using 30 Gaussians, yielding an accuracy rate of 92.8%. Then, we use the independent models for computing the models of all the possible class combinations, as described by the Algorithms 1, 2, 3 and 4.

In Figure 3, we show the trend of the accuracy rate when varying the threshold for both cases. When the distance is equal to 0, then the model simplification is not applied. As expected, the optimal solution performs better than the sub-optimal solution. However, this difference can be observed only from a certain threshold. The best case is attained without the model simplification, yielding an accuracy rate of 69.0%.

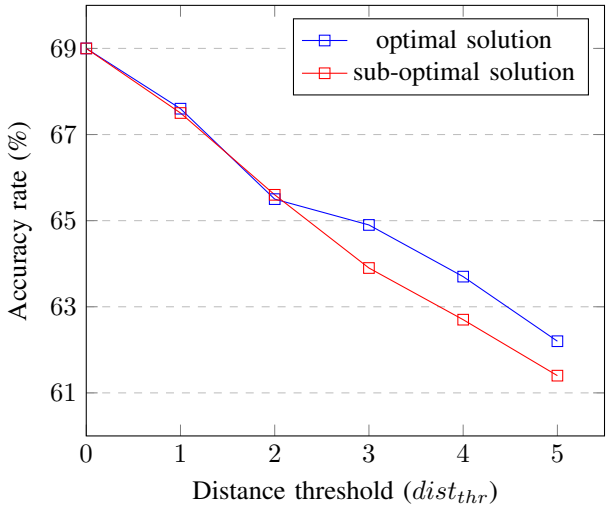

Fig. 3. Trend of the accuracy rate when increasing distance threshold. The optimal and sub-optimal solutions are represented in blue and red line.

C. Standard GMMs approach

As previously said, the standard approach consists in computing all the combinations of the time series for training the models of the class combinations. By using the Formula 1, we are able to compute the total number of combinations, equal to 23625. By using a standard GMMs, each model is trained using 225 time series combinations. As illustrated in Figure 2, the blue line shows how the accuracy rate varies when increasing the number of Gaussians. The results saturate around 200 Gaussians, yielding an accuracy rate of 65.7%.

Even if using an aggregation level L equal to 2, the computation takes a long time. Unfortunately, increasing the agglomeration level generates a large number of combinations and a large number of models. Because of the high computational time, we have to limit our comparison to an aggregation level equal to 2.

VI. Discussion

One of the strong point of the proposed procedure for generating the models for the additive time series is the capability to scale well when increasing the number of classes, the number of time series per class and the aggregation level, especially when compared to the standard approach. However, our approach is sensitive to the number of Gaussians, given that their number quadratically increases when increasing the aggregation level. For that reason it is particularly important the model simplification step, that permits to sensibility reduce the number of Gaussians. On the other side, the model simplification step yields a decrease of the accuracy rate, as shown in Figure 3.

The comparison with the standard approach in term of accuracy rate is particularly interesting. In our case of study, we analyzed the performances of our system compared to the standard approach with an aggregation level equal to 2. Our approach performs better than the standard approach when using a distance threshold smaller than 2.

VII. Conclusions

In this work we reported a new approach, called *GMMs aggregation procedure*, for the classification of additive time series. By using GMMs, we take advantage of the additive property of the Gaussians that complies with the additive property of the features of the additive time series. While the standard approach consists in training the models from additive time series, we propose a different procedure divided in three steps: (i) the modeling of the independent classes, (ii) the generation of the models for the class combinations and (iii) the simplification of the generated models. In practice, the models of the combined classes are generated by merging the Gaussians belonging to the single models. However, given the high number of Gaussian generated, we reduce their number by applying two algorithms: (i) optimal and (ii) suboptimal.

We reported about a case of study in the field of the electrical appliance recognition. In particular we compared the GMMs aggregation procedure with the standard approach, showing the benefits in terms of accuracy rate. In addition, we show the trend of the two algorithms for reducing the number of Gaussians when increasing a threshold value. A limitation of our analysis was in the number of aggregation level, that we had to limit to 2, because of the high computational time of the standard GMMs approach.

The proposed approach is completely new, and it could be potentially applied in several domains, as for instance the financial time series. To best of our knowledge, we are the firsts to propose a system for creating artificial models for aggregated time series with additive features. Probably several improvements are possible, in particular we suspect that other methods could be more effective for merging the Gaussians. Finally, our approach could be easily extended to other machine learning technique, based on the estimation of probability densities as mixture of Gaussians, as the HMMs.

References

[1] R. J. Rossana and J. J. Seaterb, "Temporal aggregation and economic time series," *Journal of Business & Economic Statistics*, vol. 13, pp. 441–451, 1995.

[2] A. Zoha, A. Gluhak, M. Imran, and S. Rajasegarar, "Non-Intrusive Load Monitoring Approaches for Disaggregated Energy Sensing: A Survey," *Sensors*, vol. 12, no. 12, pp. 16 838–16 866, 2012.

[3] M. Gaudry, "An aggregate time-series analysis of urban transit demand: The montreal case," *Transportation Research*, vol. 9, no. 4, pp. 249 – 258, 1975.

[4] A. Nanopoulos, R. Alcock, and Y. Manolopoulos, "Feature-based classification of time-series data," *Information processing and technology*, 2001.

[5] G. Tsoumakas and I. Katakis, "Multi Label Classification: An Overview," *International Journal of Data Warehousing and Mining*, vol. 3, no. 3, pp. 1–13, 2007.

[6] M. Figueiredo, A. De Almeida, and B. Ribeiro, "Home electrical signal disaggregation for non-intrusive load monitoring (NILM) systems," *Neurocomputing*, vol. 96, pp. 66–73, 2012.

[7] M. Zaki and W. Meira, Eds., *Data Mining and Analysis, Fundamental Concepts and Algorithms*. Cambridge University Press, 2014.

[8] S. Roweis and Z. Ghahramani, "Neural Computation," *Lecture Notes in Computer Science*, vol. 11, no. 2, 1998.

[9] A. Ridi, C. Gisler, and J. Hennebert, "ACS-F2 - A New Database of Appliance Consumption Analysis," in *Proceedings of the 6th International Conference on Soft Computing and Pattern Recognition (SOCPAR)*, 2014, pp. 145–150.