Neural Computing Applications manuscript No.
(will be inserted by the editor)

EANN2014: Unsupervised Feature Selection for Sensor Time-series in Pervasive Computing Applications

Davide Bacciu

Received: date / Accepted: date

Abstract The paper introduces an efficient feature selection approach for multivariate time-series of heterogeneous sensor data within a pervasive computing scenario. An iterative filtering procedure is devised to reduce information redundancy measured in terms of time-series cross-correlation. The algorithm is capable of identifying non-redundant sensor sources in an unsupervised fashion even in presence of a large proportion of noisy features. In particular, the proposed feature selection process does not require expert intervention to determine the number of selected features, which is a key advancement with respect to time-series filters in literature. The characteristic of the prosed algorithm allow to enrich learning systems, in pervasive computing applications, with a fully automatized feature selection mechanism which can be triggered and performed at run-time during system operation. A comparative experimental analysis on real-world data from three pervasive computing applications is provided, showing that the algorithm addresses major limitations of unsupervised filters in literature when dealing with sensor time-series. Specifically, it is presented an assessment both in terms of reduction of time-series redundancy as well as in terms of preservation of informative features with respect to associated supervised learning tasks.

Keywords Feature Selection · Multivariate time-series · Pervasive computing · Echo State Networks · Wireless Sensor Networks

D. Bacciu
Dipartimento di Informatica - Università di Pisa
Tel.: +39-050-2212749
Fax: +39-050-2212726
E-mail: bacciu@di.unipi.it

1 Introduction

Pervasive computing puts forward a vision of an environment enriched by a distributed network of devices with heterogeneous sensing and computational capabilities, that are used to realize customized services supporting everyday activities. Pervasive computing systems deploy sensors that continuously collect data concerning the user and/or the environmental status. This data comes under the form of streams, i.e. time-series, of sensor information with a considerably heterogeneous nature (e.g. temperature, presence, motion, etc.). This results in consistent amounts of information that need to be transferred and processed, typically in real time, to implement the system services, that are often realized by computational learning models (e.g. for predicting user activities based on sensed data) [18]. In this context, the availability of effective feature selection techniques for multivariate time-series becomes fundamental. Feature selection entails the identification of a subset of the original input sequences from a given dataset, targeted at removing irrelevant and/or redundant information sources. In a pervasive computing application this serves, on the one hand, to reduce the computational and communication overhead of transferring and processing large amounts of sensor information. On the other hand, it suppresses redundant/irrelevant information which might negatively affect the predictive performance of the learning models.

The key difference between feature selection and feature extraction is that the former approach identifies a subset of the original input features, preserving the original semantics of the data, while the latter approach seeks a transformation (more or less complex) of the original data to generate a more compact representation of the information [8]. This is particularly relevant in a pervasive system where, at run-time, the original data is sensor information distributed across the network. In this context, the computation of a feature extraction transform would typically require to transfer all sensor data to a single device where the transformation is computed. Additionally, the run-time application of a feature extraction transformation requires higher computational efforts with respect to a simple selection rule on a pre-determined subset of features. The former, in fact, requires at least to compute a linear combination of the original features (e.g. such as in Principal Component Analysis) to generate the transformed data representation (i.e with a cost that is at least linear with respect to the size of original data), whereas the latter can be performed with a constant time operation. Such computational aspects are a key factor to be taken into consideration if the application runs on low-power devices.

This work has been developed in the context of the RUBICON project [1], which proposes a vision of a wireless sensor and actuator network where data analysis and learning capabilities are spread in all components of the systems, depending on the capabilities of the hosting devices. To this end, it defines a pervasive learning system that consists of a network of learning modules distributed on devices characterized by limited computational and communication capabilities. Each such device hosts a learning component which is trained to perform real-time short-term predictions based on the temporal his-

tory of the input signals, gathered by the sensors onboard the mote or received from another node through its radio interface. The learning components are are realized using Echo State Networks (ESNs) [11], that are recurrent neural networks from the Reservoir Computing paradigm [13] which are characterized by a good trade-off between computational efficiency and ability to deal with dynamic systems and noisy data. By this means, it is possible to deploy a learning component even on devices with very low-computational and power capabilities, such as wireless sensor motes [2]. This allows a pervasive embedding of intelligence within the ambient where the learned knowledge is deployed, close to where the related input information is generated, e.g. the sensors. RUBICON realizes a general purpose learning system capable of addressing a large variety of tasks concerning the on-line processing of sensor-data streams; it also provides mechanisms that allow to continuously adapt the learned knowledge by incrementally adding new learning tasks, e.g. allowing the system to accommodate changes in the monitored environment.

We are interested in designing an efficient feature selection scheme for such a pervasive learning system. The optimization of the number of sensor streams feeding the learning modules is, in fact, a key issue in such a resource constrained environment, requiring effective feature selection techniques for multivariate time-series. Further, the fact that the RUBICON learning system allows to incrementally deploy new predictive tasks during system's operation, poses additional requirements on the feature selection model. The first is computational efficiency, as the selection process has to be performed during system operation whenever a request for a new predictive task is posted. The second is the automatization of the feature selection process, as this has to be performed automatically by the learning system without any form of human/expert intervention (e.g to determine the number of selected features from a ranking).

In literature, there are few feature selection approaches specifically tailored to multivariate time-series. Most take a *wrapper* approach where feature subset selection optimizes the performance of a specific learning model [9], typically by training multiple model instances with different configurations of the inputs. *Filter* approaches, instead, select features through an external optimization criterion: their use is typically limited to supervised classification problems, where the class labels can be exploited to identify those features that do not contribute substantially to class-separability. The CleVer approach [19], which will be further discussed in the next section, is among the unique unsupervised filter techniques developed primarily for multivariate time-series. Previous works have noted how such sophisticated state-of-the-art feature selection techniques, which show excellent performances on multivariate time-series benchmarks, do not provide significant results in the context of open-ended discovery in real-world scenarios comprising a sensor-rich environment [6]. Motivated by this, we propose a simple, yet effective, feature selection technique based on a cross-correlation analysis of multivariate sensor time-series, that is specifically tailored to the identification and removal of redundant sensor streams in an autonomous fashion. The proposed approach is

based on an iterative filter heuristics that incrementally removes/selects time-series based on redundancy information. The algorithm is characterized by limited computational requirements and by the ability to cope with the heterogeneous information sources that characterize a pervasive sensor system. Further, the feature selection process does not require expert intervention to determine the number of selected features and can therefore be fully automatized in the distributed learning system.

The remainder of the paper is organized as follows: Section 2 summarizes the background on feature selection techniques for multivariate time-series; Section 3 presents the proposed incremental feature selection algorithm, whose performance is assessed on real-world benchmarks from indoor pervasive computing scenarios in Section 4. Finally, Section 5 concludes the paper.

2 Feature Selection for Multivariate Time-series

The literature on feature selection provides a wide choice of algorithms developed for flat vectorial data, while very few approaches have been designed to deal specifically with multivariate time-series. Feature selection, in this context, amounts to the identification of relevant/non-redundant attributes of a multivariate sample that represents the evolution of information in time. Let us define an *univariate* time-series x^n as a the sequence of observations

$$x^n(1), \ldots, x^n(t), \ldots, x^n(T^n),$$

where $x^n(t)$ is the observation at time t of the n-th sample time-series, and T^n is the sequence length. A D-dimensional *Multivariate Time-Series* (MTS) $\mathbf{x}^n$ is a collection of D univariate time-series

$$x_i^n = x_i^n(1), \ldots, x_i^n(T^n) \text{ s.t. } i = 1, \ldots, D,$$

where $x_i^n(t)$ is the observation at time t of the i-th component of the n-th sample MTS. The term x_i^n denotes the i-th univariate time-series in $\mathbf{x}^n$. In the following, we use the terms feature and variable to refer to a component of the MTS: each feature i is then associated to a set of univariate time-series, one for each sample n. In this context, feature selection can be interpreted as the problem of identifying a subset of D' relevant/informative univariate time-series $\{x_{i_1}, \ldots, x_{i_{D'}}\}$ out of the original D time-series composing the MTS $\mathbf{x}$ (where we have dropped the n-th superscript to identify a generic variable, rather than the specific sample).

Approaches in literature can be differentiated, as for the flat case, in *wrapper* and *filter* methods. Wrappers are the prominent approach to feature selection for multivariate time-series in literature: here, the feature subset is selected to optimize the predictive and generalization abilities of a specific computational learning model. For instance, Recursive Feature Elimination (RFE) [9] iteratively trains a Support Vector Machine (SVM) classifier and ranks the input features with respect to a SVM-performance criterion, removing the attributes with the lowest rank at each iteration. RFE has been

originally proposed for vectorial data only but it has later been applied to MTS information, e.g. in [12]. Corona (Correlation as Features) [17] is another wrapper method that transforms each multivariate time-series into the corresponding correlation matrix, whose coefficients are fed to a support vector machine that is then used to apply the RFE method. The central issues of the wrapper approach are its considerable computational requirements, induced by the multiple retraining of the underlying learning model, and the fact that the selected attributes tend to be extremely specific for the given learning module and for the target learning task.

Differently from wrappers, filter approaches use an external optimization criterion with respect to the learning model that will be using the selected data. The optimization criterion can be either supervised, typically targeting the preservation of most task-discriminative features, or unsupervised, that usually seeks to remove redundancy between the attributes to yield to a compact set of good quality features for subsequent learning tasks. Most of the filter techniques for time-series data in literature, are limited to supervised approaches for classification tasks, such as the work in [10], where the selected time-series are those that best separate multivariate samples from different classes. The Relief method, originally proposed for vectorial data, uses entropy as a measure of the ability of a feature to discriminate classes and has been extended to time-series data by [7]. The Maximum-Relevance Minimum-Redundancy (MRMR) [16] method is among the most popular filter techniques for vectorial data whose objective is the identification of a feature subset such that selected attributes are mutually as dissimilar as possible, while they are as similar as possible to the target class variable. Similarity and dissimilarity are measured by means on pairwise mutual information and the MRMR algorithm is derived from the maximization of a joint function of the feature dissimilarity and similarity. Despite its widespread diffusion in vectorial feature selection, the MRMR algorithm has not yet found application to timeseries data: this might be the result of the difficulty in estimating mutual information on time-series. Such problems are already manifest when estimating mutual information for continuous vectorial observations [16]: these are, typically, discretized to a number of categorical values on which mutual information is ultimately estimated. The MRMR technique appears to be poorly suited to feature selection in our pervasive computing scenario, for a number of reasons: first, it is a supervised technique developed specifically for classification tasks, whereas in our pervasive computing scenario we seek an unsupervised method which identifies features that can be employed in a number of different supervised tasks, of both classification and regression type. Second, it is difficult to define accurate and robust estimators of mutual information for heterogeneous timeseries data (e.g. of mixed categorical and continuous nature) which typically require either strong approximations, such as the discretization of continuous variables, or computationally intensive routines [15]. Finally, the MRMR method provides a ranking of the features but requires the user to determine the number of ultimately selected features, whereas we seek a completely data-driven process that can automatically determine such number.

The CleVer method [19] is one of the few unsupervised filter approaches specifically tailored to multivariate time-series. It exploits the properties of the principal components common to all the time-series to provide a ranking of the more informative features. Based on the assumption that there exists a common subspace across all multivariate data items, it first performs a Principal Component Analysis (PCA) considering each univariate time-series in isolation, i.e. using the set of observations for the given univariate time-series as the PCA dataset. Then, it computes the principal components common to the univariate time-series, by bisecting the angles between their principal components, and calculates a loading matrix providing information on the contribution of each of the original D features to the common principal components. Such a loading matrix provides only a ranking of the original features; hence, a selection criterion is required to extract the relevant feature subset. Three approaches are proposed by [19] for subset selection from the ranking

- a classical top-k method selecting the k features whose loading vector has the largest $\mathcal{L}_2$-norm;
- a clustering-based method using k-means to identify attributes with similar patterns of loading values which selects the k features closest to each of the k cluster prototypes;
- an hybrid approach applying, first, k-means clustering and, then, ranking the attributes with respect to their contribution to the clusters.

Clearly, all the approaches described above require expert intervention (by the user) to determine the number of selected features, i.e. the top-k elements for the ranking based method and the number of clusters k for the k-means based approach. On the positive side, the CleVer method is characterized by low computational requirements coupled with a competitive performance on MTS feature selection benchmarks [19]. Its excellent performance, together with the fact that it represents the sole example of unsupervised feature filter for MTS in literature, motivates to consider it as the reference baseline for the experimental comparison with the algorithm introduced in this paper.

The characterizing contribution of the proposed algorithm with respect to the state of the art discussed above is two-fold. On the one hand, it puts forward an approach that formulates feature selections as a completely unsupervised process, devoid of any need for human expert intervention and that does not require to associate the feature selection process to a specific supervised learning task and/or to a specific learning model. As discussed above, the majority of MTS feature selection approaches in literature are, instead, based on supervised techniques; the very few of them taking an unsupervised approach, i.e. those related to the CleVer method, require expert intervention to determine the number of selected features from a ranking. On the other hand, the proposed algorithm is (to the extent of the author knowledge) the first specifically tailored to the identification and removal of redundant sensor MTS. Previous works [6] have observed that state-of-the-art feature selection techniques with competitive performances on MTS benchmarks are poorly suited to deal with the characteristics of multivariate sensor streams. This

paper provides an experimental grounding of such intuition by thoroughly assessing the performance of the proposed model and of the state of the art CleVer method on real-world data from pervasive sensor networks.

3 Unsupervised Sensor Time-series Selection by Cross-Correlation

The Section introduces a filter algorithm for the unsupervised open-ended discovery of non-redundant feature subsets from sensor data. The algorithm has been designed to take into account the key requirements posed by the specific pervasive computing application, that are

- the ability to deal specifically with multivariate time-series (MTS) data;
- the use of unsupervised information only, such that its result are independent of a specific predictive task;
- the automatization of the feature selection process, so that it can be performed online with respect to system operation, with no human intervention;
- computational efficiency.

To this end, we introduce the *Incremental Cross-correlation Filter* (ICF) algorithm for unsupervised feature subset selection on multivariate time-series (MTS) of sensor data. The ICF algorithm targets the reduction of feature redundancy, measured in terms of their pairwise cross-correlation. The cross-correlation of two discrete time-series x^1 and x^2 is a measure of their similarity as a function of a time lag (offset) τ, calculated through the sliding dot product

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

where $\tau \in [-(T^1-1),\ldots,0,\ldots,(T^2-1)]$ and T^1, T^2 are the time-series lengths. Intuitively, the lag where the maximum of the cross-correlation is computed provides information about the displacement between the first time-series and the second.

The cross-correlation in (1) tends to return large numbers for signals whose amplitude is larger: this would prevent from comparing time-series from different sensor modalities due to the considerably different scales of the sensor readings. To this end, we introduce the *normalized cross-correlation*

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

where $\phi_{xx}(0)$ denotes the zero-lag autocorrelation, i.e. the correlation of a time-series x with itself. The normalized function $\overline{\phi}_{x^1x^2}(\tau)$ takes values in $[-1,+1]$, where a value of $\overline{\phi}_{x^1x^2}(\tau)=1$ denotes that the two time-series have the exact same shape if aligned at time τ. Similarly, a value of $\overline{\phi}_{x^1x^2}(\tau)=-1$ indicates that the time-series have the same shape but opposite signs, while $\overline{\phi}_{x^1x^2}(\tau)=0$

denotes complete signal uncorrelation. From our point of view, both negative and positive extremes denote a certain redundancy in the information captured by the two time-series. Therefore, the correlation value at the point in time where the signals of the two time-series are best aligned is

$$\overline{\phi}^{\,*}_{x^1x^2} = \max_{\tau} |\overline{\phi}_{x^1x^2}(\tau)|. \tag{3}$$

The ICF algorithm implements a forward selection-elimination procedure that filters out redundant features, where redundancy is measured by the normalized cross-correlation in (2). ICF is based on the iterative application of a set of four selection/elimination rules, with no backtracking on the inclusion/exclusion of a feature in the final subset, which allows to maintain the computational complexity of the iterative process linear with respect to the feature number. The four selection/elimination rules are backed-up by the following intuitions

- A variable that is not correlated with any of the other features, should be selected.
- A variable that is correlated with all the variables that have already been selected is a good candidate for elimination.
- If the selection/elimination rules result in a working set of mutually correlated variables, act conservatively and maintain all those features that are less correlated with the selected ones.

The ICF algorithm is articulated in three phases:

1. The first computes a score of pairwise feature redundancy using (3);
2. The second phase performs a preliminary denoising to get rid of uninformative features;
3. The last phase iteratively applies the selection rules until all features are assigned to either the selected or the deleted status.

The first ICF phase builds a matrix of feature redundancy $R \in \{0,1\}^{D\times D}$, such that $R_{ij} = 1$ if features i and j are pairwise redundant and $R_{ij} = 0$ otherwise. Given a MTS dataset, the redundancy matrix is computed as follows

1. For each sample $\mathbf{x}^n$, use (3) to compute the maximum cross-correlation between all univariate sequences x_i^n, x_j^n in $\mathbf{x}^n$. If $\overline{\phi}^{\,*}_{x_i^n x_j^n} \approx 1$ for the pair i, j, assume features i and j to be correlated on the n-th sample, and increment the partial correlation counts in the frequency matrix (assuming $S_{ij} = 0$ initially)
$$S_{ij} = S_{ij} + 1.$$
2. Compute the percentage of samples in which each pair i, j is correlated, i.e. $S_{ij} = S_{ij}/N$.
3. Set $R_{ij} = 1$, if the corresponding feature pair i, j, with $i \neq j$, is correlated on more than $\theta_P\%$ samples, i.e. $S_{ij} > \theta_P\%$.
4. Set the diagonal of R to zero, i.e. $R_{ii} = 0$ for all features i, to discount trivial correlations.

The redundancy matrix R provides a unified picture of which variables are mutually correlated on a sufficiently large share of input samples S. Experimentally, we have determined that a value of $\theta_P\% = 20\%$ is already sufficient to detect redundancies in a variety of experimental scenarios (nevertheless the value can also be determined on a per-task basis through cross-validation). Note that numerical issues discourage from using the exact $\overline{\phi}^*_{x_i^n x_j^n} = 1$ match in item 2 above: here, we suggest to consider a pair i, j to be correlated if $\overline{\phi}^*_{x_i^n x_j^n} > 0.99$.

The second phase preprocesses the initial feature set to delete those features comprising mostly noise. To this end, it uses feature autocorrelation, that is the cross-correlation of a univariate time-series to itself. In particular, autocorrelation is computed using the un-normalized sliding dot product in (1): this measure is characterized by the fact that a time-series constituted primarily by noise has a peak value at lag $\tau = 0$ and a mean autocorrelation approximately equal to 0 off time 0 (i.e. for all other lags). More formally, given a feature i, we compute the average off time 0 absolute autocorrelation as

$$\psi_i^* = \frac{\sum_{n=1}^{N} \frac{1}{T^n - 1} \sum_{\tau \neq 0} |\phi_{x_i^n x_i^n}(\tau)|}{N} \tag{4}$$

where $\phi_{x_i^n x_i^n}(\tau)$ is defined in (1) and T^n is the length of the n-th time-series. The i-th feature is deleted if its average absolute autocorrelation approaches 0: the ψ_i^* value will not be exactly 0, in general, so the deletion test is softened by pruning those features i having $\psi_i^* < 0.1$. Deletion of a feature, in this phase, resizes the redundancy matrix R by removing the row $R_{i\cdot}$ and column $R_{\cdot i}$ associated with the feature.

The third phase applies the feature selection/elimination rules exploiting the information in the redundancy matrix R generated by the previous two steps. It defines a set of unassigned features $\mathcal{F}$, that initially contains all the variables that have not been deleted by the second phase. The rules are applied iteratively to $\mathcal{F}$ following a priority order, until all features are assigned to either the set of selected variables $\mathcal{SF}$ or to the set of the deleted ones $\mathcal{DF}$. The details of the ICF rules and their priority pattern are described by the following procedure

1. RULE 1 - If a row $R_{i\cdot}$ is completely uncorrelated with the others in R (i.e. $R_{i\cdot}$ contains only zeros)
   (a) Add i to the selected subset: $\mathcal{SF} = \mathcal{SF} \cup \{i\}$;
   (b) Remove i from $\mathcal{F}$ and remove the corresponding entries in R;
   (c) If an uncorrelated feature j is generated as a result of the previous step, move j from $\mathcal{F}$ to $\mathcal{DF}$ and remove the corresponding entries in R.
2. RULE 2 - If a row $R_{i\cdot}$ is correlated with all the others and there is at least 1 non completely correlated feature (i.e. R does not contain only ones off-diagonal)
   (a) Add i to the deleted subset: $\mathcal{DF} = \mathcal{DF} \cup \{i\}$;
   (b) Remove i from $\mathcal{F}$ and remove the corresponding entries in R.

3. RULE 3 - If all features in $\mathcal{F}$ are mutually correlated with each other, i.e. R contains only ones off-diagonal,
   (a) Select the feature i that is less correlated with those currently in $\mathcal{SF}$;
   (b) Add i to the selected subset: $\mathcal{SF} = \mathcal{SF} \cup \{i\}$;
   (c) Remove i from $\mathcal{F}$;
   (d) Move the remaining features $\mathcal{F}$ to the deleted subset ($\mathcal{DF} = \mathcal{DF} \cup \mathcal{F}$) and terminate.
4. RULE 4 - If neither RULE 2 nor RULE 3 apply,
   (a) Extract feature $i \in \mathcal{F}$ that is correlated with the minimum number of features still in $\mathcal{F}$;
   (b) Define $\mathcal{S}(i) \subset \mathcal{F}$ as the subset of features correlated with i and select $j \in \mathcal{S}(i)$ as the maximally correlated feature with those currently in $\mathcal{SF}$;
   (c) Add i to $\mathcal{SF}$ and j to $\mathcal{DF}$;
   (d) Remove i, j from $\mathcal{F}$ and remove the corresponding entries in R.

The elimination-selection rules are tested sequentially, in the order in which they are presented above, and their test conditions are such that at least one of them *fires* at each iteration of the algorithm; hence, the cost of computing the third phase is at most linear in the number of the original features.

The rationale of step 1(c) above is that a feature j encoding the same information of already selected variables has to be deleted to avoid to be selected by future steps (otherwise RULE 1 is likely to be applied to j at the following iteration). Note that, in step 3(a), we determine the feature i that is minimally correlated with those in $\mathcal{SF}$ by measuring the pairwise cross-correlation between i and all $j \in \mathcal{SF}$, averaged across all samples, i.e.

$$\mu_{ij} = \sum_{n=1}^{N} \frac{\overline{\phi}^{*}_{x_i^n x_j^n}}{N}, \tag{5}$$

where N is the number of multivariate time-series in the dataset. The value of μ_{ij} is then used to determine the minimally correlated feature i (in average) as

$$i = \arg\min_{i' \in \mathcal{F}} \left\{ \frac{\sum_{j \in \mathcal{SF}} \mu_{i'j}}{|\mathcal{SF}|} \right\}, \tag{6}$$

where $|\mathcal{SF}|$ is the cardinality of $\mathcal{SF}$. Step 4(a) uses a similar strategy to determine which feature j, from the set of i-correlated features $\mathcal{S}(i)$, has to be deleted, i.e.

$$j = \arg\max_{j' \in \mathcal{S}(i)} \left\{ \frac{\sum_{k \in \mathcal{SF}} \mu_{j'k}}{|\mathcal{SF}|} \right\}. \tag{7}$$

The use of cross-correlation to assess the pairwise information redundancy among timeseries is not novel. The key contribution of ICF is to use such a measure within a novel forward selection/elimination scheme based on the intuitions discussed early in this section. By this means, ICF allows to perform a multivariate feature selection process using only pairwise dependency information, thus maintaining a limited computational complexity (as discussed in

the forthcoming paragraph). Such an approach resembles the characteristics of constraint-based algorithm for Bayesian Network structure search [4]: here, such intuition is used, for the first time, in the context of multivariate time-series. Despite the ICF algorithm being defined in terms of cross-correlation, this is general enough to be seamlessly extended to work with any pairwise or multivariate measure of timeseries rendundancy/dependency.

A pseudo-algorithmic description of the complete ICF procedure is provided in Algorithm 1. The computational complexity of this algorithm is, in general, dominated by the computation of the redundancy matrix in the first phase which mainly depends on the cost of computing the pairwise cross-correlation on the sample MTS. The asymptotic complexity of redundancy matrix computation is

$$O(N \cdot (D^2 \cdot T^{max})),$$

where N is the dataset length and the second term results from the computation of pairwise cross-correlations between D univariate time-series with a maximum length T^{max}. Computation of the autocorrelation parameter for noisy suppression in the second ICF can be embedded in the redundancy matrix calculation loop at no additional costs, as shown in Algorithm 1. The third ICF phase is very efficient, i.e. linear in the number of features D, as the forward selection scheme processes each variable, at most, once with constant time operations. Therefore, the final complexity of the ICF algorithm is $O(N \cdot (D^2 \cdot T^{max}) + D)$. The asymptotic complexity of the CleVer method cannot be straightforwardly derived as it depends on the number of iterations required by the k-means algorithm to identify the feature clusters. Nevertheless, it can be approximated from below as

$$\Omega(N \cdot (D^2 \cdot T^{max}) + N^2 \cdot p),$$

where the first summation term depends on the singular value decomposition performed on the N timeseries, while the second term is due to the identification of the p common principal components among the timeseries [19]. Hence, the ICF asymptotic complexity can be considered not worse that that of the CleVer method.

4 Experimental Evaluation

4.1 Experimental Setup and Scenario

The experimental evaluation is intended to assess the capability of the ICF algorithm in detecting and removing redundant MTS features in indoor pervasive computing scenarios. In particular, we compare the performance of ICF with respect to the CleVer method, a state of the art unsupervised feature filter for time-series, for a varying proportion of irrelevant features in the original MTS[1]. To this end, we have employed real-world data collected in two

[1] Matlab code for ICF and CleVer available at `www.di.unipi.it/~bacciu/icf`

Algorithm 1 Incremental Cross-correlation Filter

Require: A dataset of N multivariate time-series $\mathbf{x}^n$ composed of D features.

// Redundancy matrix computation (phase 1)
for $n = 1$ **to** N **do**
  for $i, j = 1$ **to** D **do**
    if $i == j$ **then**
      $\psi_i = \psi_i + (\sum_{\tau \neq 0} |\phi_{x_i^n x_i^n}(\tau)|)/(T^n - 1)$
    else if $\overline{\phi}^*_{x_i^n x_j^n} > 0.99$ **then**
      $S_{ij} = S_{ij} + 1$
    end if
  end for
end for
for $i = 1$ **to** D **do**
  for $j = 1$ **to** D **do**
    if $S_{ij}/N > 0.2$ and $i \neq j$ **then**
      $R_{ij} = 1$
    end if
  end for
  // Noise reduction (phase 2)
  if $(\psi_i/N) < 0.1$ **then**
    $\mathcal{DF} = \mathcal{DF} \cup i$
  end if
end for
Remove i-th rows and columns from R for all $i \in \mathcal{DF}$
// Forward Selection/Elimination (phase 3)
Set $\mathcal{F} = \{1, \ldots, D\} \setminus \mathcal{DF}$, $\mathcal{SF} = \{\}$
while $\mathcal{F}! = \{\}$ **do**
  // Rule 1
  if $\exists i \in \mathcal{F}$ s.t. $(R_{ij} == 0)$ $\forall j \in \mathcal{F} \setminus i$ **then**
    $\mathcal{F} = \mathcal{F} \setminus i$, $\mathcal{SF} = \mathcal{SF} \cup i$
    Remove i-th rows and columns from R
    if $\exists k \in \mathcal{F}$ s.t. $(R_{kj} == 0)$ $\forall j \in \mathcal{F} \setminus k$ **then**
      $\mathcal{F} = \mathcal{F} \setminus k$, $\mathcal{DF} = \mathcal{DF} \cup k$
      Remove k-th rows and columns from R
    end if
  end if
  // Rule 2
  if $\exists i \in \mathcal{F}$ s.t. $(R_{ij} == 1)$ $\forall j \in \mathcal{F} \setminus i$ and R contains at least one 0 off-diagonal **then**
    $\mathcal{F} = \mathcal{F} \setminus i$, $\mathcal{DF} = \mathcal{DF} \cup i$
    Remove i-th rows and columns from R
  else if R is a matrix of all 1's off-diagonal **then**
    // Rule 3
    Select feature i using (6) and set $\mathcal{F} = \mathcal{F} \setminus i$, $\mathcal{SF} = \mathcal{SF} \cup i$
    $\mathcal{DF} = \mathcal{DF} \cup \mathcal{F}$, $\mathcal{F} = \{\}$
  else
    // Rule 4
    Find $i = \arg\min_{i' \in \mathcal{F}} \sum_{j \in \mathcal{F}} R_{i'k}$
    Find $j \in S(i)$ using (7)
    $\mathcal{F} = \mathcal{F} \setminus \{i, j\}$, $\mathcal{SF} = \mathcal{SF} \cup i$, $\mathcal{DF} = \mathcal{DF} \cup j$
    Remove i-th and j-th rows and columns from R
  end if
end while
return $\mathcal{SF}$,$\mathcal{DF}$

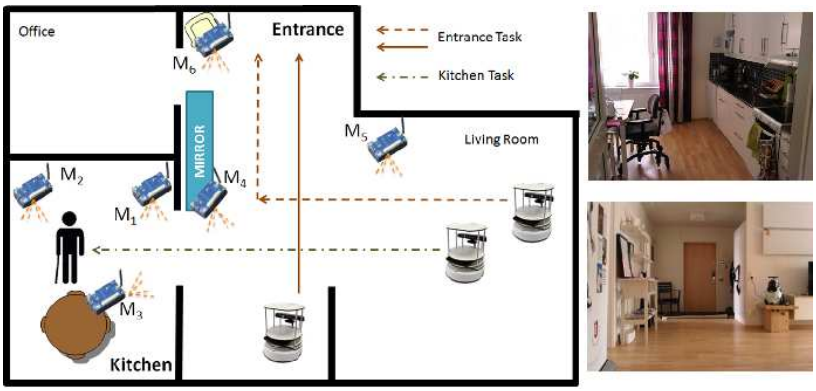


Fig. 1 Experimental scenario for the Entrance and Kitchen tasks in the Ängen facilities: M_i denotes the i-th WSN mote (Telosb platform running TinyOS). The two photos on the right show snapshots of the kitchen (top) and entrance-living (bottom) areas of the flat.

experimental deployments, one associated to mobile robot navigation and one related to Human Activity Recognition (HAR).

The former scenario comprises two different regression tasks involving the prediction of robot navigation preferences in a sensorized home-environment. The idea underlying these tasks is to learn to predict which navigation system is best to use to perform a certain trajectory based on environment characteristics and on user preferences. The preference weight to be learned is a value in $[0, 1]$, where 1 is interpreted as maximum confidence on the navigation system and 0 denotes the lowest preference (i.e. the navigation system should not be used). The resulting computational learning task is, basically, a regression problem between the multivariate input time-series and the corresponding univariate sequence of preference weights. The second scenario comprises the classification of which activity is being performed by a user based on sensor readings from wireless devices deployed in the environment as well as worn by the user. Note that, for the purpose of feature selection, we only consider the input information (i.e. the sensor readings and the robot trajectory information) but we discard the target data (i.e. the preference weight and the activity classification) as we are interested in assessing unsupervised selection methods. An in-depth analysis of the supervised learning tasks associated to the mobile robot experiment is provided by [5].

The mobile robot scenario has been designed and put into operation in the Ängen senior residence facilities in Örebro Universitet. The scenario, depicted in Fig. 1, comprises a real-world flat sensorized by an RFID floor, a mobile robot with range-finder localization and a Wireless Sensor Network (WSN) with six mote-class devices, where the term M_i is used to denote the i-th mote. Each device is equipped with light (L), temperature (T), humidity (H) and passive infrared (P) presence sensors. The input information sources include all sensors from the six motes, plus robot trajectory information under the form of its (x, y) position and orientation θ, for a total of 24 features.

As shown in Fig. 1, the experimental assessment involves two tasks. The Entrance task is intended to predict a weight evaluating the performance of the localization system on two different trajectory types, represented as dashed and continuous lines in Fig. 1. Performance on the dashed trajectory is expected to be low due to the effect of mirror disturbances which, conversely, should not affect trajectories on the continuous line. For the purpose of feature selection, the only relevant information is robot position and orientation (x, y and θ) as well as the P sensors onboard motes M_3 and M_6 (referred to as P_3 and P_6, respectively), that are the only presence sensors triggered by robot motion. The remainder of the sensors collect data that is poorly informative as it does not undergo significant changes across the timespan of data collection. The Kitchen task concerns a single trajectory type (dash-dotted arrows in Fig. 1) heading to the kitchen, where a user might be present or not. Since the robot range-finder localization is based on camera, the user is willing to switch it off every time he/she is in the room with the robot (the corresponding example trajectories are then marked with minimal preference, i.e. 0). The target of this task is to learn this user preference based on robot trajectory information and on the user presence pattern captured by the P sensors. The relevant information for this task is robot x-position (orientation and y coordinates do not change for this trajectory type) as well as the P sensors onboard motes M_1 to M_5 (i.e. P_1 to P_5), that are the only presence sensors that are triggered by robot or human motion. A total of 87 and 104 sequences have been collected for the two tasks sampling at 2Hz with an average length of 127 and 197 elements.

The HAR scenario involves a WSN comprising 4 stationary devices, called *anchors*, that are deployed on the walls of a bedroom as depicted in Fig 2. These anchors exchange radio packets with 3 sensor motes, that measure the signal strength of the received radio packets and are also equipped with temperature sensors (T) and 2D accelerometers on the x (Ax) and y (Ay) axes. Two of such motes, i.e. M_1 and M_2, are worn by the user as shown in Fig. 3; the latter mote, i.e. M_3, is deployed on a small table as shown in Fig. 2. The *Received Signal Strength* (RSS) measured from the packets received by the motes provide some very noisy form of distance information from the anchors on the walls, allowing to localize the motes in the 2D space (e.g. and also the user wearing them, see for instance the application in [2]). Therefore, the input information sources in this scenario include the 3 sensors from the 3 motes, plus the 4 RSS measurements from each anchor to each mote, for a total of 21 features.

The HAR experimental campaign comprised the collection of 90 sequences corresponding to 3 classes (of 30 sequences each) of user activities, that are *Exercising*, *Relaxing* and *Cleaning*. The sequences have been collected with the user performing the three activities as naturally as possible and the collected sequences have been hand-labeled with the corresponding activity class. The exercising activity involves a number of push-ups, squats and sit-ups and it is performed in the middle of the room; relaxation involves laying still on the sofa, while cleaning is associated to dusting furniture located in different

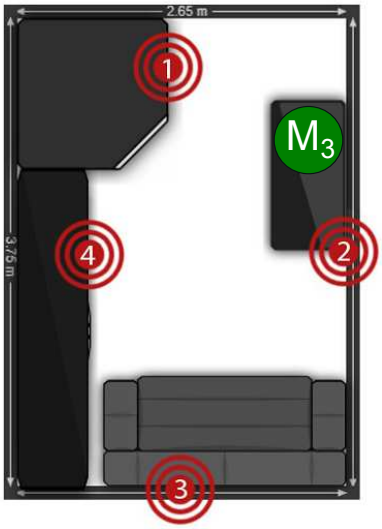


Fig. 2 Room layout for the HAR scenario: anchor devices are located at an height of 1.80m from the floor and their position is represented by the red circles enclosing their identifier number. Mote M_3 is located on a small table (dark rounded rectangle on the right hand side). The *Relaxation* activity is performed on the sofa (depicted in light gray at the bottom of the picture), *Exercising* is carried on in the center of the room, while *Cleaning* involves moving around the room. The black rectangles on the left-side denote generic room furniture (not relevant for the task).

Fig. 3 Motes worn by the user are located on the right wrist, i.e. mote M_1, and on the right ankle, i.e. mote M_2.

positions of the room. Due to the nature of the task, it is hard to define ground-truth knowledge on which information source is significative and non-redundant: for instance, we expect the RSS sources to be relevant (due to the positioning information they convey) but also redundant (it is likely that 2 or 3 anchors provide sufficient information given the relatively small room size). Nevertheless, we can identify with reasonable certainty which sensors are gathering poorly informative measurements, that are all the temperature

sensors, as well as the accelerometers on mote M_3 and its associated RSS (given that is permanently located on a table).

4.2 Experimental Results and Discussion

The performance of the CleVer and ICF algorithms can be evaluated in terms of what information sources are selected for varying input configurations comprising different initial sets of features and characterized by increasing proportions of redundant features. Among the three CleVer subset selection approaches discussed in 2, we have implemented the k-means selection approach as it has shown the best experimental performance in the original CleVer paper [19]. Table 1 shows the features selected by CleVer and ICF on the Entrance and Kitchen task of the mobile robot scenario, where the selected relevant features are highlighted in bold. In Table 1, we use the term M_i to indicate that the input configuration includes all the transducers in the i-th mote, while x, y and θ denote the robot position and orientation. Since the CleVer algorithm requires the user to determine the number of selected features, we provide two set of results: one (CleVer-OPT) using the (known) optimal number of relevant features; the second (CleVer-ID) using the number of features found by ICF on the same configuration.

The prevalence of bold-highlighted terms in Table 1 shows that the ICF algorithm always manages to identify a larger number of significative features, with respect to both CleVer algorithm versions. In particular, the ICF is capable of consistently reducing the number of input features by maintaining the majority (if not all, as in the Kitchen task) of the relevant features even when a large number of uninformative features is included. Conversely, the performance of both the CleVer methods deteriorates consistently as the proportion of redundant features increases. Additionally, the results on last four configurations of the Entrance task in Table 1 highlight a key critical point of the CleVer algorithm, which yields different features subsets for different repetitions of the feature selection process (note that the number of selected features in CleVer-OPT and CleVer-ID is the same for these configurations). This behavior originates from the well-known sensitivity to initialization of the k-means algorithm integrated in the CleVer selection process. ICF, on the other hand, has a stable behavior yielding to the selection of the same feature subset for multiple algorithm repetitions. A stable behavior is fundamental for operating feature selection in our pervasive computing scenario, as we are seeking a reliable and compact set of features which will serve as inputs of a learning module that will be automatically trained and deployed during system operation, with no expert intervention to counteract randomizing effect of the feature selection process.

A quantitative evaluation of the performance of ICF and CleVer methods can be provided in terms of precision and recall analysis of the selected features. In this context, precision is the proportion of selected features that are truly significative (i.e. the true positives) with respect to the total size of the

Table 1 Feature selection result on the Entrance and Kitchen data for varying input configurations: M_i denotes all the transducers in the i-th mote while x, y and θ are the robot position and orientation. The relevant features (based on expert knowledge) are in bold.

Entrance Task			
Configuration	CleVer-OPT	CleVer-IT	ICF
(M_3,x,y,θ)	L$_3$,$\mathbf{P}_3$,T_3,$\mathbf{y}$	L_3,$\mathbf{P}_3$,$\mathbf{y}$	$\mathbf{P}_3$,$\mathbf{x}$,$\mathbf{y}$
(M_3,M_6,x,y,θ)	L$_3$,$\mathbf{P}_3$,$\mathbf{P}_6$,T$_6$,$\boldsymbol{\theta}$	L$_3$,$\mathbf{P}_6$,$\mathbf{x}$,$\boldsymbol{\theta}$	$\mathbf{P}_6$,$\mathbf{P}_3$,$\mathbf{x}$,$\mathbf{y}$
(M_4-M_6,x,y,θ)	L_4,P_4,L$_6$,T$_6$,$\boldsymbol{\theta}$	P_4,L_6,T_6,$\boldsymbol{\theta}$	L$_4$,P_4,$\mathbf{P}_6$,$\mathbf{y}$
(M_3-M_6,x,y,θ)	L$_3$,T$_3$,P_4,$\mathbf{P}_6$,$\boldsymbol{\theta}$	L$_3$,$\mathbf{P}_3$,T$_5$,$\boldsymbol{\theta}$	$\mathbf{P}_3$,$\mathbf{P}_6$,$\mathbf{x}$,$\mathbf{y}$
(M_1-M_6,x,y,θ)	P_2,L$_3$,P_4,L$_5$,$\boldsymbol{\theta}$	L$_1$,P_2,T$_2$,L$_3$,L$_6$,$\boldsymbol{\theta}$	P_1,P_2,$\mathbf{P}_3$,$\mathbf{P}_6$,$\mathbf{x}$,$\mathbf{y}$
Kitchen Task			
Configuration	CleVer-OPT	CleVer-IT	ICF
(M_3,x,y,θ)	$\mathbf{x}$,y	L_3,$\mathbf{x}$,y	L_3,$\mathbf{P}_3$,$\mathbf{x}$
(M_1,M_3,x,y,θ)	L_3,$\mathbf{x}$,y	L_3,T_3,y	L_1,$\mathbf{P}_3$,$\mathbf{x}$
(M_1-M_3,x,y,θ)	L_1,L_2,H_2,$\mathbf{x}$	L_1,L_2,H_2,$\mathbf{x}$	$\mathbf{P}_1$,$\mathbf{P}_2$,$\mathbf{P}_3$,$\mathbf{x}$
(M_1-M_5,x,y,θ)	L_1,$\mathbf{P}_2$,L_3,T_3,L_4,H_5	L_1,L_2,H_2,L_3,T_3,L_4	$\mathbf{P}_1$,$\mathbf{P}_2$,$\mathbf{P}_3$,$\mathbf{P}_4$,$\mathbf{P}_5$,$\mathbf{x}$
(M_1-M_6,x,y,θ)	T_1,L_2,H_2,L_3,$\mathbf{P}_4$,P_6	T_1,L_2,$\mathbf{P}_2$,H_2,L_3,$\mathbf{P}_4$	$\mathbf{P}_1$,$\mathbf{P}_2$,$\mathbf{P}_3$,$\mathbf{P}_4$,$\mathbf{P}_5$,$\mathbf{x}$

feature subset, which therefore includes both True Positives (TP) and False Positives (FP). Recall, on the other hand, provides information of whether the algorithms are identifying most of the relevant features for the task, i.e. the proportion of TP with respect to the total number of significative features. In other words, precision and recall are

$$prec = \frac{TP}{TP + FP} \quad \text{and} \quad rec = \frac{TP}{TP + FN},$$

where FN are the False Negatives, i.e. the relevant features not included in the final feature subset.

Figure 4 provides the precision and recall plots corresponding to the configurations in Table 1, as a function of the initial number of features. The behavior of the precision-recall curves for ICF confirms the intuition that the number of FP and FN does not grow with the size of the search space (and the number of potentially irrelevant features). In particular, ICF shows a markedly higher recall than the CleVer methods, exhibiting a conservative redundancy reduction process which prevents from discarding consistent shares of features that will become relevant for the successive training of the supervised learning tasks. Conversely, both CleVer methods experience a marked precision and recall deterioration when the number of initial features and the proportion of irrelevant ones grows, which is due to an increase in both FP and FN in the identified features subsets.

To further assess the performance of the filter methods with respect to noisy inputs, we have modified the original Entrance and Kitchen data by introducing 10 artificial features whose observations have been generated by random sampling from a uniform distribution in $[0, 1]$ and from a mean-zero

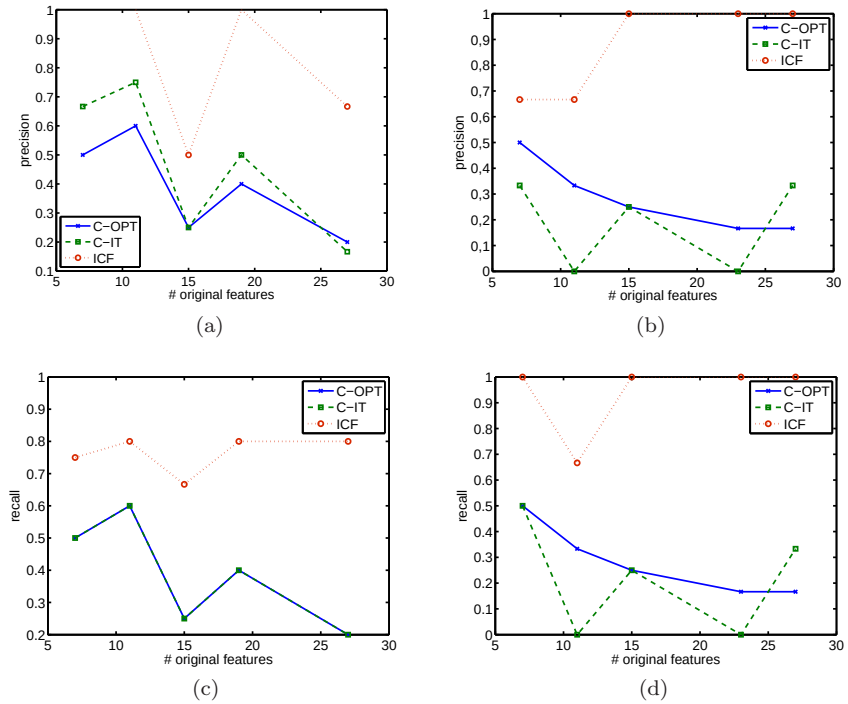


Fig. 4 Quantitative feature selection performance, where Clever algorithms are identified as C-OPT and C-IT: 4(a) and 4(b) show the precision of the selected features as a function of the original input space size for the Entrance and Kitchen tasks, respectively. Figures 4(c) and 4(d) show the associated recall: note that the C-OPT recall curve is completely overlapping with that by C-ID in 4(c).

unit-variance Gaussian. Table 2 shows the features identified by the 3 methods: the original input configuration includes all the sensor and trajectory information (i.e. correspond to the last lines of the Entrance and Kitchen task in Table 1); the *10-Unif* and *10-Gauss* denote the noise-injected datasets corresponding to uniform and Gaussian sampling, respectively. ICF results are clearly not influenced by the presence of noisy components, yielding to the identification of the same feature subsets found on the original data, on both tasks. Such noise robustness is due to the autocorrelation filtering, as the 10 noisy features are removed completely at second phase of ICF and never enter the forward selection-elimination part of the algorithm. Conversely, the CleVer method seems to be considerably affected by both types of noisy features. In particular, the presence of noisy features confuses the Clever selection process to the point that it cannot identify any significant feature in the majority of the cases listed in Table 2.

Representation entropy [14] provides an additional means to quantitatively assess the effectiveness of the algorithms in terms of amount of redundancy present in the selected feature subsets. Let X be the $K \times K$ covariance matrix

Table 2 Effect of noise addition on feature selection performance: *Original* denotes the configuration using all inputs from mote M_1 to M_6 and trajectory information; *10-Unif* and *10-Gauss* denote the *Original* configuration extended with additional 10 input features generated by uniform noise in $[0, 1]$ and zero-mean/unary-variance Gaussian noise, respectively. The relevant features (based on expert knowledge) are in bold.

Entrance Task			
Configuration	CleVer-OPT	CleVer-IT	ICF
Original	P_2,L_3,P_4,L_5,$\boldsymbol{\theta}$	L_1,P_2,T_2,L_3,L_6,$\boldsymbol{\theta}$	P_1,P_2,$\mathbf{P}_3$,$\mathbf{P}_6$,$\mathbf{x}$,$\mathbf{y}$
10-Unif	T_2,H_2,$\mathbf{x}$,N_2,N_4	L_2,H_2,T_5,N_2,N_3,N_4	P_1,P_2,$\mathbf{P}_3$,$\mathbf{P}_6$,$\mathbf{x}$,$\mathbf{y}$
10-Gauss	T_5,$\mathbf{x}$,N_1,N_5,N_6	H_2,T_5,N_1,N_4,N_6,N_9	P_1,P_2,$\mathbf{P}_3$,$\mathbf{P}_6$,$\mathbf{x}$,$\mathbf{y}$
Kitchen Task			
Configuration	CleVer-OPT	CleVer-IT	ICF
Original	T_1,L_2,H_2,L_3,$\mathbf{P}_4$,P_6	T_1,L_2,$\mathbf{P}_2$,H_2,L_3,$\mathbf{P}_4$	$\mathbf{P}_1$,$\mathbf{P}_2$,$\mathbf{P}_3$,$\mathbf{P}_4$,$\mathbf{P}_5$,$\mathbf{x}$
10-Unif	L_6,T_6,N_2,N_3,N_5,N_6	H_2,L_5,N_6,N_7,N_8,N_{10}	$\mathbf{P}_1$,$\mathbf{P}_2$,$\mathbf{P}_3$,$\mathbf{P}_4$,$\mathbf{P}_5$,$\mathbf{x}$
10-Gauss	$\mathbf{P}_2$,L_6,N_1,N_4,N_5,N_9	H_2,L_5,N_2,N_3,N_7,N_8	$\mathbf{P}_1$,$\mathbf{P}_2$,$\mathbf{P}_3$,$\mathbf{P}_4$,$\mathbf{P}_5$,$\mathbf{x}$

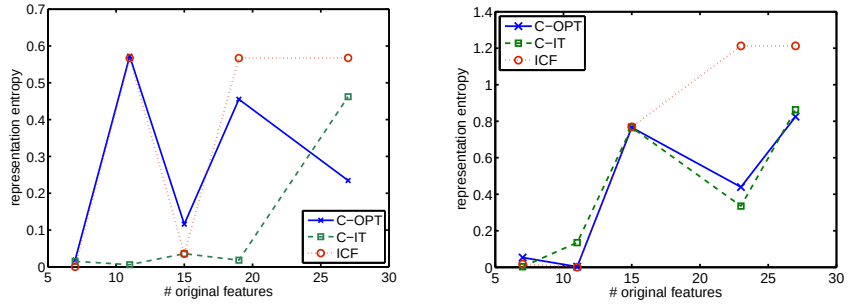
of the K selected features and λ_i be the eigenvalue of X associated to the i-th feature, we define the normalized eigenvalue

$$\overline{\lambda}_i = \lambda_i / \sum_{j=1}^{K} \lambda_j.$$

Then, the representation entropy can be written as

$$E_R = -\sum_{i=1}^{K} \overline{\lambda}_i \log \overline{\lambda}_i \tag{8}$$

and it is such that E_R attains its minimum when all the information is concentrated along a single feature, making the rest redundant, while it is maximum when the information is equally distributed among all the features. In other words, the representation entropy of the selected subset provides a measure of how much redundant is the final set of features. Figure 5(a) and 5(b) show the E_R value for the Entropy and Kitchen tasks as a function of the input configuration. Overall, ICF confirms its ability to identify and filter-out redundant information, by selecting features that encode different information, yielding to consistently better performances with respect to CleVer when the proportion of redundant features is higher. Coherently with the precision-recall analysis, the advantages of ICF over CleVer are particularly marked on the Kitchen task. The representation entropy on the Entrance task in Figure 5(a) has a less neat behavior: ICF has the best entropy performance when the proportion of redundant features is higher, though one can observe a drop in the representation entropy corresponding to the third configuration in Table 1. This is consistent with the drop in precision that can be observed in Figure 4(a) for the same configuration. The low representation entropy for ICF in this particular configuration might be due to the presence of the P_4 and



(a) (b)

Fig. 5 Effect of feature selection on redundancy reduction, in terms of representation entropy of the ICF and CleVer methods on the Entrance (Fig. 5(a)) and Kitchen (Fig. 5(b)) tasks.

P_6 sensors, which are both selected only in this case, and that might encode partially redundant information.

Representation entropy does not provide information on how significant is the portion of information left out of the final subset, nor it provides an indication on how adequate are the identified features for the final supervised learning task. To understand whether the redundancy reduction process implemented by ICF has an impact on such final learning tasks, we have considered a simple supervised learning scenario, comprising training of an ESN on the Kitchen task, using different numbers of reservoir neurons from the set $[10, 50, 100, 300, 500]$ (see [11] for further details on the model and [5] for a detailed account on model selection and on the supervised learning task). Training has been performed using a cross validation setting to identify the model hyperparameters and performance has been measured in terms of the Mean Absolute Error (MAE) on a test set comprising hold-out sequences not seen in training and validation. Figure 6 shows the performance when using all the WSN inputs without feature filtering, when using oracular ground truth knowledge on the relevant features and when using the CleVer-OPT,CleVer-IT and ICF filtered inputs reported in the last row of Table 1. Clearly, there is no difference between the performance when using oracular knowledge and the ICF-selected configuration, confirming that the ICF algorithm is capable of identifying a compact set of non-redundant features, without discarding task-significant features that may affect the final supervised learning performance. Conversely, the features identified by both CleVer methods do not provide sufficient discriminative information yielding to considerably poorer supervised learning performances. Note how the presence of redundant features in both CleVer-selected and non-feature selected configurations negatively impacts the performance as the size of the ESN increases, due to the fact that the noise introduced by the redundant features tends to be incorporated in the larger parameter space.

Table 3 shows the feature selection results on the HAR task comprising sensor data related to user exercising, relaxing and cleaning activities. Due

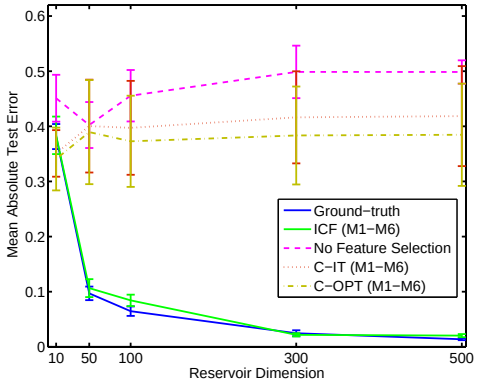


Fig. 6 Mean Squared Error of the supervised learning model on the Kitchen task using different input configurations resulting from the use of ICF and CleVer feature selection (ICF, C-IT and C-OPT), ground-truth oracular knowledge and when using all available inputs (No Feature Selection).

Table 3 Feature selection result on the HAR data for varying input configurations: M_i denotes all the transducers in the i-th mote, Ax_i and Ay_i are the 2D accelerometers of the i-th mote and R_i^j denotes the RSS value of the i-th mote with respect to the j-th anchor. The best representation entropy E_R for each configuration is highlighted in bold.

	CleVer-OPT		CleVer-IT		ICF	
Conf	Selected	E_R	Selected	E_R	Selected	E_R
(M_1)	Ax_1,Ay_1,R_1^1,R_1^2,R_1^3,R_1^4	0.624	Ax_1,R_1^1,R_1^3	0.022	R_1^2,R_1^3,R_1^4	**1.063**
(M_2)	T_2,Ay_2,R_2^1,R_2^2,R_2^3,R_2^4	0.058	Ay_2,R_2^4	0.016	R_2^1,R_2^3	**0.665**
(M_1, M_3)	Ax_1,$R_1^1R_1^2R_1^4$,R_3^2,R_3^4	0.043	Ax_1,R_1^3,R_3^4	0.019	R_1^2,R_1^3,R_1^4	**1.063**
(M_2, M_3)	T_2,R_2^1,R_2^2,R_2^3,R_2^4,R_3^2	**1.517**	R_2^3,R_3^1	0.465	R_2^1,R_2^3	0.665
(M_1, M_2)	Ax_1,Ay_1,$R_1^1R_1^2R_1^3$,R_1^4, T_2,Ay_2,R_2^1,R_2^2,R_2^3,R_2^4	0.851	Ax_1,$R_1^1R_2^3$	0.785	$R_1^2R_1^3$,R_1^4	**1.063**
$(M_1 - M_3)$	T_1,$R_1^1R_1^2R_1^3$,R_1^4,R_2^1,R_2^2, R_2^3,R_2^4,Ax_3,R_3^3,R_3^4	0.097	T_1,$R_1^1R_1^2$	0.020	$R_1^2R_1^3$,R_1^4	**1.063**

to the nature of the task, it is difficult to determine a-priori which features are truly significant and non-redundant; hence, in Table 3, we measure feature selection performance in terms of the purely unsupervised representation entropy score in (8). Nevertheless, as noted in Section 4.1, there are certain sensors sources that are certainly capturing little information, given the user activity setup. This is the case of all temperature sensors T_i as well as of the RSS information associated to mote M_3, i.e. R_3^j for the RSS received from the j-th anchor. The ICF algorithm always selects a very compact set of features, all involving RSS information; nevertheless, none of them involves the non-significant table mote M_3. Conversely, both versions of the CleVer algorithm select at least one RSS value in mote M_3 whenever this is provided in the input configuration. Additionally, CleVer has also the tendency to select the non-significant temperature sensors from motes T_1 and T_2. This is particularly true

for the CleVer-OPT version, due to the larger number of features selected (in this case, due to the lack of ground truth, we have approximated the optimal number of features to all those features that were not clearly non-significant). The values of the representation entropy confirm the ability of ICF in isolating a compact number of non-redundant features, yielding to the highest entropy on almost all input configurations. The CleVer-IT method, despite using the same number of features as ICF, yields considerably poorer results, with as little as 2% of the representation entropy for the same input configuration.

To evaluate the impact of feature selection on the final HAR tasks, we have performed an analysis of the supervised learning performance of ESN models trained on the three classification tasks, that are the recognition of the Exercising, Relaxing and Cleaning activities. We have considered the same model selection and cross-validation setup discussed for the Kitchen task in Figure 6, while varying the number of reservoir neurons in $[10, 50, 100, 300]$; the performance has again been measured in terms of mean absolute error on hold-out test sequences (amounting to roughly 30% of the total sequences). Figure 7 shows the performance when using all inputs without feature filtering as well as when using the features selected by the three methods (ICF, Clever-IT and Clever-OPT) corresponding to the last row of Table 3. These results suggest that the three HAR tasks are characterized by fewer irrelevant features with respect to the Kitchen task and which do not seem to have a negative impact on the predictive performance when using all available inputs. This, in turn, results in the fact that the CleVer-OPT configuration has generally the second best performance, as it is the feature-selected configuration using the largest set of original inputs, i.e. 57% of the all input features. The ICF configuration, on the other hand, uses only 3 features yielding predictive performances that are comparable to the CleVer-OPT and non feature selected cases in the Exercising task. On the Relaxing and Cleaning tasks, ICF performance is closer to CleVer-OPT, which uses thrice the number of input features, than to CleVer-IT, which uses the same number of features: for instance, on the Relaxing task ICF yields to test errors 2% higher than CleVer-OPT, whereas CleVer-IT makes 10% more errors. This confirms ICF ability in identifying compact subsets of non-redundant features, that encode relevant information for the supervised learning tasks, in a completely unsupervised manner. CleVer, on the other hand, requires expert supervision to determine the number of selected features and the relative quality of its features with respect to supervised performance seems to be lower than that of ICF. The trade-off between reducing the number of input features and achieving good supervised performances is central for our pervasive computing application, where the supervised ESN models are intended to be deployed on computationally constrained devices. Here, reducing the number of ESNs inputs preserves memory and computational resources (by reducing the number of learning model parameters) and may prolong battery duration in WSN devices (due to the lower transmission costs associated with fewer inputs). In this sense, the ICF results in Figure 7 seem promising as they show a good trade-off between predictive accuracy and the number of input features needed to achieve it.

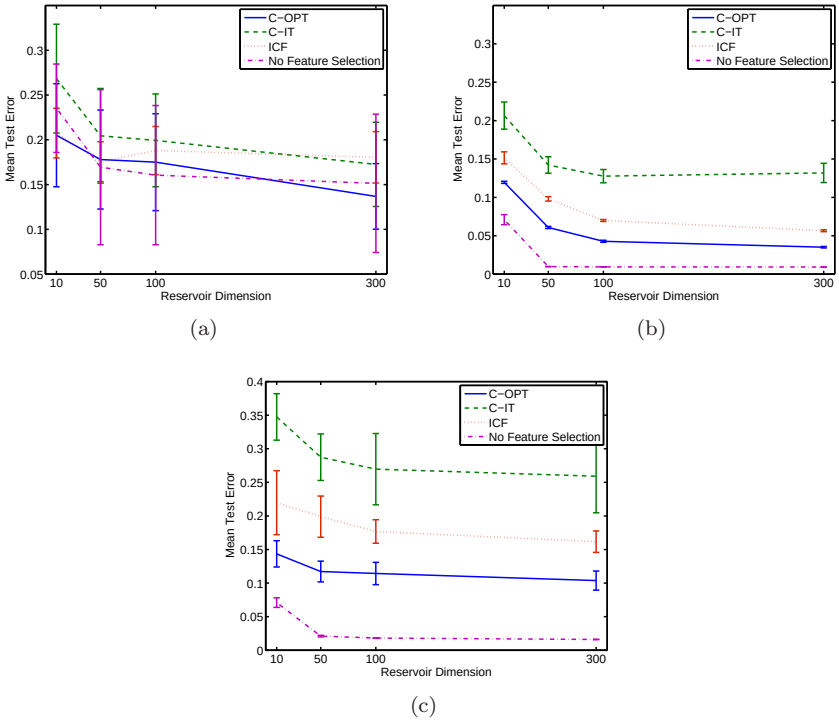


Fig. 7 Mean Test Error achieved by the supervised learning model on the three HAR classification task, comprising recognition of Exercising (7(a)), Relaxing (7(b)) and Cleaning (7(c)) activities. Results have been obtained for different input configurations resulting from the use of CleVer (C-OPT and C-IT) and ICF feature selection, as well as when using all available inputs (No Feature Selection).

5 Conclusion

Multivariate sensor time-series comprise large shares of noisy, highly redundant information which can hamper the deployment of effective predictive models in pervasive computing applications. As noted in [6] and experimentally confirmed in this paper, state-of-the-art feature filtering algorithms with competitive performances on MTS benchmarks are poorly suited to deal with the characteristics of such noisy, slowly changing, yet heterogeneous in nature, sensor streams. To address this fundamental limitation, we have introduced an efficient feature filter algorithm tailored to real-time pervasive computing applications.

The ICF algorithm has been shown to be capable of identifying non-redundant sensor information in a completely unsupervised fashion and to outperform the state-of-the-art CleVer filter method on different pervasive computing scenarios. Differently from CleVer, ICF does not require expert intervention to determine the number of selected features and provides stable feature subsets that do not change with algorithm initialization. ICF effec-

tiveness is not obtained at the cost of its computational efficiency, with an asymptotic complexity that is at most quadratic with respect to the feature set size, resulting in running times that are comparable with that of the efficient CleVer algorithm. For instance, the average time required to complete feature selection for the most complex configuration of the Entrance task (i.e. the fifth in Table 1) is of 2153msec, obtained by Java code running in an Eclipse box on an Intel I5 Quad-core at 2.7 GHz CPU equipped with 4GBytes of RAM. Note that the majority of the running time is spent on redundancy mask computation, while feature filtering effort is negligible, i.e. 1msec. When considering much more resource constrained environments such time to complete is expected to remain acceptable for non real-time applications. For instance, the same Entrance task can be expected to complete in roughly 914 seconds (time estimated based on floating-point benchmarks execution costs) on a ARM Cortex M3 CPU, which is a widely adopted architecture on smart watch systems.

The robustness and limited computational complexity of ICF makes it an excellent candidate to implement an automatized feature selection mechanism within an autonomous learning system, such as that developed as part of the RUBICON project [2]. In particular, we are planning to exploit ICF as a preliminary filtering step to reduce the complexity of a relevance-guided supervised wrapper specifically targeted at optimizing the predictive performance of the ESNs implementing the distributed learning system [3]. In this sense, the fact that ICF is limited to the detection of linear time-series correlation assumes less relevance, as we expect the supervised wrapper algorithm to identify possible nonlinear dependencies, e.g. through the nonlinearity of the ESN reservoir neurons, and to determine if the associated features can be deleted with significative advantages for the predictive performance of the supervised task. Nevertheless, we would like to study whether the ICF iterative policy can be successfully applied also to non-linear time-series correlation measures to extend the range of its applications.

Acknowledgements

This work is supported by the FP7 RUBICON project (contract n. 269914). The author would like to thank Claudio Gallicchio for providing part of the results on the Echo State Network experiment, as well as Filippo Barontini for the collection of the HAR dataset.

References

1. Amato, G., Bacciu, D., Broxvall, M., Chessa, S., Coleman, S., Di Rocco, M., Dragone, M., Gallicchio, C., Gennaro, C., Lozano, H., McGinnity, T., Micheli, A., Ray, A., Renteria, A., Saffiotti, A., Swords, D., Vairo, C., Vance, P.: Robotic ubiquitous cognitive ecology for smart homes. Journal of Intelligent and Robotic Systems pp. 1–25 (2015)

2. Bacciu, D., Barsocchi, P., Chessa, S., Gallicchio, C., Micheli, A.: An experimental characterization of reservoir computing in ambient assisted living applications. Neural Computing and Applications **24**(6), 1451–146 (2014)
3. Bacciu, D., Benedetti, F., Micheli, A.: ESNigma: efficient feature selection for Echo State Networks. In: Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN'15), pp. 189–194 (2015)
4. Bacciu, D., Etchells, T.A., Lisboa, P.J., Whittaker, J.: Efficient identification of independence networks using mutual information. Computational Statistics **28**(2), 621–646 (2013)
5. Bacciu, D., Gallicchio, C., Micheli, A., Di Rocco, M., Saffiotti, A.: Learning context-aware mobile robot navigation in home environments. In: Information, Intelligence, Systems and Applications, IISA 2014, The 5th International Conference on, pp. 57–62. IEEE (2014)
6. Cheema, S., Henne, T., Koeckemann, U., Prassler, E.: Applicability of feature selection on multivariate time series data for robotic discovery. In: Proc. of ICACTE'10, vol. 2, pp. 592–597 (2010)
7. García-Pajares, R., Benítez, J.M., Sainz-Palmero, G.: FRASel: a consensus of feature ranking methods for time series modelling. Soft Computing **17**(8), 1489–1510 (2013)
8. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. The Journal of Machine Learning Research **3**, 1157–1182 (2003)
9. Guyon, I., Weston, J., Barnhill, S., Vapnik, V.: Gene selection for cancer classification using support vector machines. Mach. Learn. **46**(1-3), 389–422 (2002)
10. Han, M., Liu, X.: Feature selection techniques with class separability for multivariate time series. Neurocomput. **110**, 29–34 (2013)
11. Jaeger, H., Haas, H.: Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. Science **304**(5667), 78–80 (2004)
12. Lal, T.N., Schroder, M., Hinterberger, T., Weston, J., Bogdan, M., Birbaumer, N., Scholkopf, B.: Support vector channel selection in BCI. Biomedical Engineering, IEEE Transactions on **51**(6), 1003–1010 (2004)
13. Lukoševičius, M., Jaeger, H.: Reservoir computing approaches to recurrent neural network training. Computer Science Review **3**(3), 127 – 149 (2009)
14. Mitra, P., Murthy, C.A., Pal, S.K.: Unsupervised feature selection using feature similarity. IEEE Trans. Pattern Anal. Mach. Intell. **24**(3), 301–312 (2002)
15. Papana, A., Kugiumtzis, D.: Evaluation of mutual information estimators for time series. International Journal of Bifurcation and Chaos **19**(12), 4197–4215 (2009)
16. Peng, H., Long, F., Ding, C.: Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. Pattern Analysis and Machine Intelligence, IEEE Transactions on **27**(8), 1226–1238 (2005)
17. Yang, K., Yoon, H., Shahabi, C.: A supervised feature subset selection technique for multivariate time series. In: Proc. of FSDM'05, pp. 92–101 (2005)
18. Ye, J., Dobson, S., McKeever, S.: Review: Situation identification techniques in pervasive computing: A review. Pervasive Mob. Comput. **8**(1), 36–66 (2012)
19. Yoon, H., Yang, K., Shahabi, C.: Feature subset selection and feature ranking for multivariate time series. IEEE Trans Knowl. Data Eng. **17**(9), 1186–1198 (2005)