Supervised Feature Subset Selection and Feature Ranking for Multivariate Time Series without Feature Extraction

Shuchu Han **Alexandru Niculescu-Mizil**
NEC Laboratories America

Abstract

We introduce supervised feature ranking and feature subset selection algorithms for multivariate time series (MTS) classification. Unlike most existing supervised/unsupervised feature selection algorithms for MTS our techniques do not require a feature extraction step to generate a one-dimensional feature vector from the time series. Instead it is based on directly computing similarity between individual time series and assessing how well the resulting cluster structure matches the labels. The techniques are amenable to heterogeneous MTS data, where the time series measurements may have different sampling resolutions, and to multimodal data.

1 Introduction

From cyber-physical systems to IoT to healthcare, multi-dimensional time series data is ubiquitous in many applications and it is collected at an increasing rate and by an increasing number of sensors. It is not uncommon for larger systems to be instrumented with thousands of sensors making multiple measurements a second. The transmission, storage, processing and analysis of such a large amount of data is often impractical, especially if the analysis must be real-time, or must be performed on low powered edge computing devices. To cope with this problem practical systems often use only data from a small subset of informative sensors, while irrelevant or redundant sensors are ignored.

In this paper we tackle the sensor selection problem in the context of multivariate time-series classification where the task is to assign one label to an entire MTS segment. The traditional approach to this problem is vectorize the data by extracting several features from each time-series in an MTS segment (e.g. mean value, standard deviation, spectrum, etc.) and concatenate the features from all time-series into an 1-D feature vector of fixed dimension. Once this vectorized representation of an MTS segment is obtained, classical feature selection can be used to obtain a small subset of informative features (see Figure 1). One downside of this approach is that it is critically reliant on feature engineering and on the user's domain knowledge to decide what kinds of features to extract. For example, one we may say that the spectrum of sound at different frequency is a deciding factor for classifying different pronunciations; or the mean value of stocks is one of the crucial factors for the value of S&P 500 index. This, however, may be difficult when the user has little knowledge or intuition on the nature of the connection between the different time-series and the label. In this case one would have to resort to simply extracting well-known time-series features using some tool such as TSFRESH [Li et al., 2016] and hope for the best.

To circumvent the feature engineering problem we propose two supervised sensor selection algorithms for multivariate time-series classification that do not require the vectorization of the MTS segments. Instead, we only require a distance measure between time-series which can be calculated directly using, for example, Dynamic Time Warping (DTW) [Berndt and Clifford, 1994]. The key intuition behind our algorithms is that, if a sensors produces similar time-series in segments with the same label, and dissimilar time-series in segments with different labels, then that sensor is likely an informative one.

Based on this idea, we propose both a sensor ranking algorithm (akin to filter based methods in classical feature selection) and a sensor subset selection algorithm. For sensor ranking, we calculate a relevance score for each sensor by first constructing a similarity graph among the time-series produced by the sensor

Preliminary work. Under review by AISTATS 2020. Do not distribute.

across all MTS segments. We then find the largest eigenvector of the normalized adjacency matrix of this graph, which reflects its cluster structure [Shi and Malik, 2000]. Finally, the relevance score of a sensor is calculated as the normalized mutual information between this eigenvector and the ground truth labels (Figure 1).

While the sensor ranking technique is simple and efficient, it will assign a similar relevance score to highly correlated sensors thus potentially leading to the selection of redundant sensors. To address this problem we also propose a sensor subset selection algorithm. We again start by calculating an adjacency matrix for each sensor, then find a linear combination these matrices that (1)approximates similarity matrix of the labels and (2)uses a small number of sensors and (3)uses minimally redundant sensors.

Since the proposed techniques are only based on distance measures between time-series produced by the same sensor do not require that all sensors sample data at the same rate. This makes them readily applicable to heterogeneous MTS data where different sensors have different sampling rates, without needing to sub-sample high frequency sensors or interpolate low frequency sensors in order to convert the MTS data to a matrix format. This is a major advantage over MTS sensor selection techniques based on inter-sensor correlation such as CLeVer [Yoon et al., 2005] or Corona [Yang et al., 2005] which require all sensors to have the same sampling rates.

The rest of paper is organized as follows. In Section (2), we provide some background and the math notations that are used in our equations. In Section (3), we present our algorithms for sensor ranking and sensor subset selection. In section (4), we discuss our connection to two-stage kernel learning algorithms. The experiments and discussion are introduced in Section (5). In Section (6), we extend our work to the application on heterogeneous data. And we summarize our work in Section (7).

2 Background and Notations

Assume we are given a labeled data set $\{(\boldsymbol{X}_i, y_i)\}_{i=1..n}$, where $\boldsymbol{X}_i$ is a MTS segment, and $y_i \in \mathcal{R}$ is the corresponding label. The each MTS segment $\boldsymbol{X}_i$ consists of m time-series or sensors $\{\boldsymbol{x_{i,j}}\}_{j=1..m}$ with $\boldsymbol{x_{i,j}} \in \mathcal{R}^{l_{i,j}}$. The length $l_{i,j}$ of each time-series $\boldsymbol{x_{i,j}}$ may vary between segments due to different segment duration (i.e. $l_{i_1,j} \neq l_{i_2,j}$) or between sensors due to different sampling rates (i.e. $l_{i,j_1} \neq l_{i,j_2}$) or both. The goal is to find a subset $j_1, ..., j_k \subset [1..m]$ of sensors that that are (1) predictive of the labels and (2) have low redundancy.

Symbol	Dimension	Meaning
$\boldsymbol{X}_i$		MTS
$\boldsymbol{x}_{i,j}$	$\mathcal{R}^{m \times l_{i,j}}$	time-series
y_i	$\mathcal{R}$	data label
$\boldsymbol{y}$	$\mathcal{R}^n$	label vector
$\boldsymbol{M}_j$	$\mathcal{R}^{n \times n}$	distance matrix of j-th row
$\boldsymbol{W}_j$	$\mathcal{R}^{n \times n}$	similarity graph of j-th row
$\boldsymbol{D}$	$\mathcal{R}^{n \times n}$	degree matrix
$\boldsymbol{v}$	$\mathcal{R}^n$	power iteration embedding
$\boldsymbol{L}$	$\mathcal{R}^n$	graph Laplacian
$\boldsymbol{G}_I$	$\mathcal{R}^{n \times n}$	redundancy constraint matrix
$\boldsymbol{Q}$	$\mathcal{R}^{n \times n}$	redundancy constraint matrix
$\boldsymbol{H}$	$\mathcal{R}^{n^2 \times m}$	flattened similarity matrix

Table 1: Math notations.

The distance between two time-series $\boldsymbol{x_{i_1,j}}$ and $\boldsymbol{x_{i_2,j}}$ can be calculated using Dynamic Time Warping (DTW) [Berndt and Clifford, 1994, Keogh and Ratanamahatana, 2005]. For example, given two time series:

$$S = s_1, s_2, \cdots, s_i, \cdots, s_n$$
$$T = t_1, t_2, \cdots, t_j, \cdots, t_m,$$

the sequences S and T can be arranged to form a n-by-m grid, where each grid point, (i, j), corresponds to an alignment between elements s_i and t_j. A warping path, P, maps the elements of S and T, such that the "distance" between them are minimized:

$$P = p_1, p_2, \cdots, p_K, \quad max(n, m) \leq K < n + m,$$

where K is the length of wrap path and the k^{th} warp path is:

$$p_k = (i, j),$$

where i is an index from time series S and j is an index from time series T. The warp path start from $p_1 = (1, 1)$ and end at $p_K = (n, m)$. The index i and j have to be monotonically increasing in the warp path.

If we define a distance between two elements, such as:

$$\delta(i, j) = |s_i - t_j|.$$

The DTW distance of two time series is the optimal warp path with minimum distance:

$$DTW(S, T) = \min_P [\sum_{k=1}^{q} \delta(p_k)],$$

where q is the length of optimal warping path.

3 Algorithm

3.1 Build Similarity Graph for Sensors

As mentioned, we build a similarity graph for each sensor of $\boldsymbol{X}_i$. There exists many graph construction

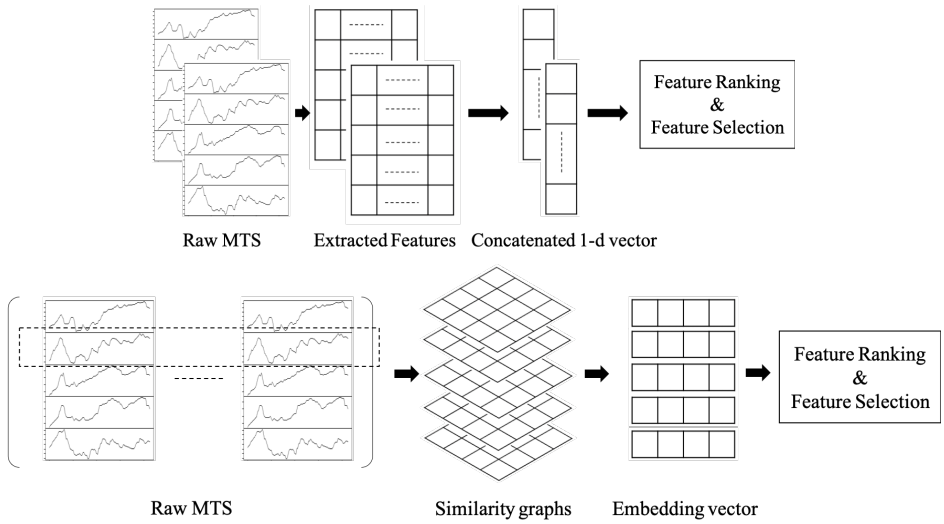

Figure 1: Top: common approach for feature selection based on feature engineering. Bottom: proposed solution in this work.

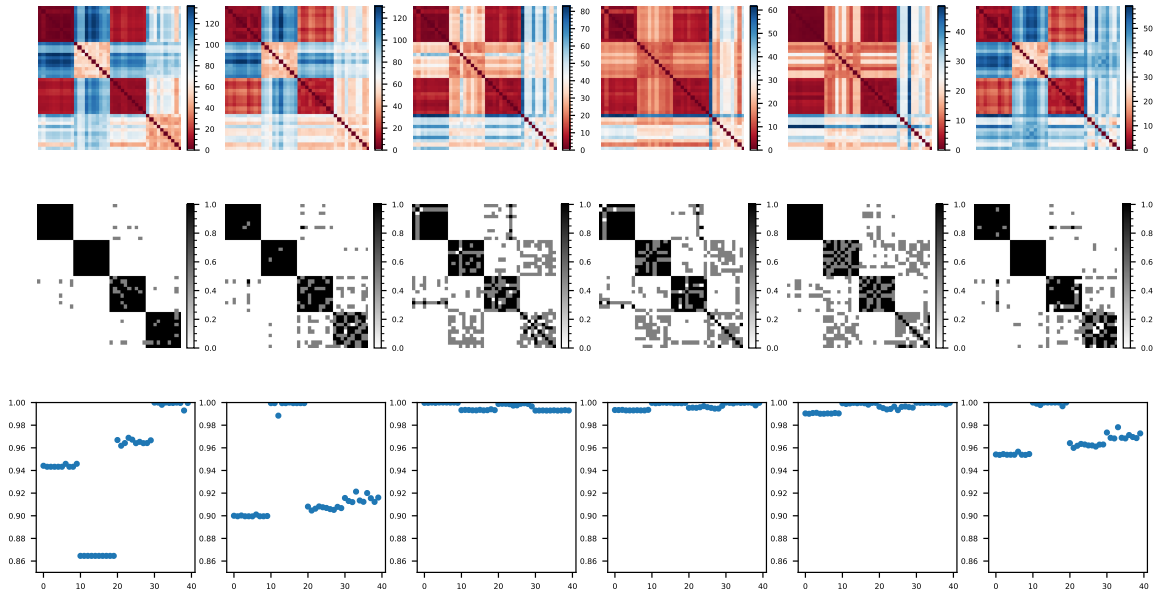

Figure 2: Illustration of our idea by using the "BasicMotions" dataset. The MTS samples are sorted according to their labels ("Standing", "running", "walking", "badminton") from left to right as in each subplots on purpose. The goal is to observe the cluster structure revealed by the DTW distance. Top: visualization of distance matrix where the distances is calculated by DTW. Center: Affinity matrices of constructed k-nearest neighbor graph. Bottom: the corresponding spectral embedding vector by PIE.

algorithms in machine learning research area, for example, k-nearest neighbor graph, sparse graph [Liu et al., 2010] and etc. In our work, we choose *k-nearest neighbor graph* as it can preserve the local geometry structure of original data's distribution. The first step (1) is to calculate the distance between two time series. We assume our time series data are real values and use the Dynamic Time Warping [Berndt and Clifford, 1994] [Keogh and Ratanamahatana, 2005] metric. (Note: Our work can easily be extended to non-real value data, as long as the user can define a distance metric. For example, for string representation, the Levenshtein distance (or Edit distance) can be used.) (2) once we have the distance matrix $\boldsymbol{M}_j$ for j-th row of $\boldsymbol{X}$, we generate the k-nearest neighbor graph $\boldsymbol{W}_j$ which is directed graph with edge weight equal to "1.0". The reason we reset the distance is for the purpose of normalization since the DTW distance bound of different sensors are quite uncertain. After that, we transfer the directed graph into an undirected one by:

$$\boldsymbol{W}_j = 0.5 * (\boldsymbol{W}_j + \boldsymbol{W}_j^T), \tag{1}$$

then we obtain a similarity graph with symmetric adjacency matrix. Following the spectral clustering work [Shi and Malik, 2000], we set the diagonal of $\boldsymbol{W}_j$ to zero.

To summary, the output of this graph construction step is a set of matrices: $\{\boldsymbol{W}_1, \boldsymbol{W}_2, \cdots, \boldsymbol{W}_m, \boldsymbol{W}_y\}$, where $\boldsymbol{W}_j \in \mathcal{R}^{n\times n}$.

3.2 The Structure Representation Vector of a Graph

In this section, we introduce a spectral embedding vector which encodes the cluster structure information of similarity graphs we constructed in last section. We believe spectral graph embedding is a right approach as the performance shown in spectral clustering [Shi and Malik, 2000] and unsupervised feature selection [Cai et al., 2010]. Among many possible choices such as Power Iteration Embedding (PIE) [Lin and Cohen, 2010], Top-k Spectral Clustering Embedding [Shi and Malik, 2000] and Heat Kernel Embedding [Peng et al., 2015], we choose PIE as our spectral embedding vector by its "1-D" dimensional vector property and its simplicity of calculation. The description of PIE is as follows.

The power iteration embedding embedding vector is an early stopping approximation of the largest eigenvector of normalized affinity matrix $\boldsymbol{W}$ which equals to $\boldsymbol{D}^{-1}\boldsymbol{W}$, where $\boldsymbol{D}$ is the degree matrix of $\boldsymbol{W}$. The power iteration embedding $\boldsymbol{v}^t$ can be calculated as:

$$\boldsymbol{v}^t = c\boldsymbol{D}^{-1}\boldsymbol{W}\boldsymbol{v}^{t-1}, \tag{2}$$

where c is a normalizing constant to limit the value of $\boldsymbol{v}$, and is set as $c = \|\boldsymbol{D}^{-1}\boldsymbol{W}\boldsymbol{v}^{t-1}\|_1^{-1}$. mbedding approximates the cluster structure of graph by using an one dimensional vector. The detailed algorithm is described in Alg. (1).

Algorithm 1: PIE($\boldsymbol{W}$) [Lin and Cohen, 2010]

Input: matrix $\boldsymbol{W} \in \mathcal{R}^{n\times n}$.
Output: power iteration embedding $\boldsymbol{v}^t \in \mathcal{R}^{n\times 1}$.
Apply positive random normalization: $\boldsymbol{W} \leftarrow \boldsymbol{D}^{-1}\boldsymbol{W}$;
Initialize $\boldsymbol{v}^0 \in \mathcal{R}^{n\times 1}$;
Repeat;
$\boldsymbol{v}^{t+1} \leftarrow \frac{\boldsymbol{W}\boldsymbol{v}^t}{\|\boldsymbol{W}\boldsymbol{v}^t\|_1}$;
$\delta^{t+1} \leftarrow |\boldsymbol{v}^{t+1} - \boldsymbol{v}^t|$;
$t \leftarrow t+1$;
Until $\|\delta^t - \delta^{t+1}\|_{max} \simeq 0$;
return $\boldsymbol{v}^t$

3.3 Sensor Ranking

Our first task is to rank the importance of sensors according to the labels. We introduce a filter method named as "PIE-rank" which includes two steps: (1) calculates the PIE embedding vector of each sensor's similarity graph, and (2) evaluates the Normalized Mutual Information (NMI) score between the PIE embedding vector and ground truth label vector $\boldsymbol{y}$. The NMI score is calculated as follows.

$$r(\boldsymbol{v}^t) = \text{score}(\boldsymbol{v}^t, \boldsymbol{y}) = \text{NMI}(\boldsymbol{v}^t, \boldsymbol{y}) = \frac{I(\boldsymbol{v}^t, \boldsymbol{y})}{\sqrt{H(\boldsymbol{v}^t)H(\boldsymbol{y})}}, \tag{3}$$

where $I(\cdot,\cdot)$ denotes the mutual information , and $H(\cdot)$ denotes the entropy. When calculate the mutual information between $\boldsymbol{v}^t$ and $\boldsymbol{y}$, k-means clustering is used to put the real values of $\boldsymbol{v}^t$ into different bins. The number of bins equals to the number of different labels in $\boldsymbol{y}$. Higher ranking score means the spectral embedding vector is more close to the label's distribution.

The sensor ranking algorithm (PIE-rank) can be summarized by:

Algorithm 2: PIE-rank

Input: MTS and labels: $\{(\boldsymbol{X}_i, y_i)\}$.
Output: Ranked scores $\boldsymbol{r}$.
for $i \leftarrow 1$ **to** m **do**
  Build the distance graph $\boldsymbol{M}_i$;
  Generate the k-nearest neighbor graph $\boldsymbol{W}_i$;
  $\boldsymbol{W}_i = 0.5 * (\boldsymbol{W}_i + \boldsymbol{W}_i^T)$;
  $\boldsymbol{v}^t = \text{PIE}(\boldsymbol{W}_i)$;
  $\boldsymbol{r}(i) = \text{scores}(\boldsymbol{v}^t, y)$;
end
Sort($\boldsymbol{r}$);
return $\boldsymbol{r}$

3.4 Sensor Subset Selection

The sensor ranking algorithm introduced in previous Section 3.3 has very straightforward result and characterized by its simplicity and efficiency. However, the redundancy existing in the top selected sensors [Yu and Liu, 2004] are not handled. The redundancy [Peng et al., 2005] means there exists several sensors which are highly correlated to each other. To minimize the redundancy, we present the sensor subset selection algorithm "PIE-SS" ("SS" means "Subset Selection") in this section.

3.5 Object function

Our learning goal is to select a subset of sensors with minimum redundancy. Our intuition is to search a

sparse linear combination of each sensor's similarity graph, and let the combined new graph can approximate the distribution of labels (represented by a label graph) as much as possible.At the same time, the selected subset sensors should have minimum redundancy w.r.t the NMI value among their PIE embedding vectors.

One challenge here is that the redundancy constraint is in the spectral space while the linear approximation of similarity graph is in original data space. It is possible that two different sensor graphs will have same spectral embedding vector. For example, for two graphs have the same cluster structure but different edge connection patterns in each cluster, they spectral embedding vectors will be very similar to each other.

Base on the above learning goal and observations, we propose following object function to calculate the optimal sensor subset with minimum redundancy:

$$\min_{\alpha} \frac{1}{2}\|\boldsymbol{W}_y - \sum_{i=1}^{m}\alpha_i \boldsymbol{W}_i\|_F^2 + \lambda\sum_{i=1}^{m}|\alpha_i| + \beta\sum_{i,j}\alpha_i\alpha_j I(\boldsymbol{v}_i^t, \boldsymbol{v}_j^t), \quad \text{s.t. } \alpha_i \geq 0. \quad (4)$$

where $\lambda \geq 0, \beta \geq 0$ are Lasso penalty and redundancy control parameters. The first term represents the approximation of label graph by the linear combination of sensor graphs. The second term put the sparse constraint to the coefficients of linear combination. The third term add redundancy constraints to the selection of sensor graph in final subset.

The objective function can be rewrote in matrix form as:

$$\min_{\alpha} \frac{1}{2}\|\boldsymbol{W}_y - \boldsymbol{W}\boldsymbol{\alpha}\|_F^2 + \lambda\|\boldsymbol{\alpha}\|_1 + \beta\boldsymbol{\alpha}^T\boldsymbol{G}_I\boldsymbol{\alpha}. \quad (5)$$

The proposed object function is non-convex by nature as the matrix $\boldsymbol{G}_I$ is not a positive semi-definite one (even it is close to) [Jakobsen, 2014]. Moreover, the diagonal elements of $\boldsymbol{G}_I$ also let the object function to penalty the "self-redundancy" $\boldsymbol{G}_I(i,i)$ and create selection bias in favor of sensor graph whose embedding vector has lower entropy as pointed out in [Nguyen et al., 2014]. Following the treatment in [Nguyen et al., 2014], we introduce the matrix $\boldsymbol{Q}$ and hyperparameter λ to equation (5). The matrix $\boldsymbol{Q}$ is defined as:

$$\boldsymbol{Q}_{ij} = \begin{cases} \boldsymbol{I}(\boldsymbol{v}_i^t, \boldsymbol{y}) & \text{if } i = j \\ \frac{1}{2}(\boldsymbol{I}(\boldsymbol{v}_i^t; \boldsymbol{y}|\boldsymbol{v}_j^t) + \boldsymbol{I}(\boldsymbol{v}_j^t; \boldsymbol{y}|\boldsymbol{v}_i^t)) & \text{if } i \neq j. \end{cases} \quad (6)$$

The improved object function then becomes:

$$\mathcal{J}(\boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{W}_y - \boldsymbol{W}\boldsymbol{\alpha}\|_F^2 + \lambda\|\boldsymbol{\alpha}\|_1 + \beta\boldsymbol{\alpha}^T(\boldsymbol{Q} + \gamma\mathcal{I})\boldsymbol{\alpha}, \quad (7)$$

where $\mathcal{I}$ is the identity matrix who has the same size as $\boldsymbol{Q}$.

3.6 Calculation of Q for Large Data

The calculation of mutual information is time consuming, and it is quite difficult to finish the calculation of matrix $\boldsymbol{Q}$ in a short time. To solve this scalability issue, we apply the Nystrom matrix approximation following the strategy in [Nguyen et al., 2014].

$$\tilde{\boldsymbol{Q}} = \begin{bmatrix} A & B \\ B^T & B^T A^{-1} B \end{bmatrix} \quad (8)$$

$\tilde{\boldsymbol{Q}}$ and Positive Semidefinite(PSD) Matrix. Our first step is to pre-processing the matrix $\tilde{\boldsymbol{Q}}$ by adjusting the parameter γ. The goal is to improve $\tilde{\boldsymbol{Q}}$ to $\hat{\boldsymbol{Q}}$ which is a PSD matrix.

3.6.1 Coordinate Descent Solver

To solve Equation (7), we apply the Coordinate Descent algorithm [Friedman et al., 2010] [Wright, 2015] by its scalability. We flatten each matrix $\boldsymbol{W} \in \mathcal{R}^{n\times n}$ into a vector representation $\boldsymbol{h} \in \mathcal{R}^{n^2\times 1}$. This will help to better understand steps of calculating derivatives. Now Equation (7) is changed to:

$$\mathcal{J}(\boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{h}_y - \boldsymbol{H}\boldsymbol{\alpha}\|_2^2 + \lambda\|\boldsymbol{\alpha}\|_1 + \beta\boldsymbol{\alpha}^T\hat{\boldsymbol{Q}}\boldsymbol{\alpha},$$

where $\boldsymbol{H} \in \mathcal{R}^{n^2\times m}$ and each column of $\boldsymbol{H}$, $\boldsymbol{H}_j$, is the flat version of $\boldsymbol{W}_j$, $\boldsymbol{h}_y \in \mathcal{R}^{n^2\times 1}$, $\boldsymbol{\alpha} \in \mathcal{R}^{m\times 1}$ and $\hat{\boldsymbol{Q}} \in \mathcal{R}^{m\times m}$.

The object function $\mathcal{J}(\boldsymbol{\alpha})$ can be split into two parts as:

$$\mathcal{J}(\boldsymbol{\alpha}) = f(\boldsymbol{\alpha}) + g(\boldsymbol{\alpha}) \quad (9)$$

$$f(\boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{h}_y - \boldsymbol{H}\boldsymbol{\alpha}\|_2^2 + \beta\boldsymbol{\alpha}^T\hat{\boldsymbol{Q}}\boldsymbol{\alpha} \quad (10)$$

$$g(\boldsymbol{\alpha}) = \lambda\|\boldsymbol{\alpha}\|_1. \quad (11)$$

We separate each term into α_k part and α_{-k} part. The notation '$-k$' means the set does not include k-th item.

$$
\begin{aligned}
f(\boldsymbol{\alpha}) = &\frac{1}{2}\|\boldsymbol{h}_y - \boldsymbol{H}_{\cdot,-\boldsymbol{k}}\boldsymbol{\alpha}_{-\boldsymbol{k}} - \boldsymbol{H}_{\cdot,k}\alpha_k\|_2^2 \\
&+ \beta\boldsymbol{\alpha}_{-k}\hat{\boldsymbol{Q}}_{-k,-k}\boldsymbol{\alpha}_{-k} \\
&+ \beta\alpha_k(\sum_j \hat{\boldsymbol{Q}}_{kj}\boldsymbol{\alpha}_j) + \beta(\sum_i \boldsymbol{\alpha}_i\hat{\boldsymbol{Q}}_{ik})\alpha_k,
\end{aligned}
$$

The derivative of f over one coordinate α_k is:

$$
\begin{aligned}
\frac{\partial f}{\partial \alpha_k} &= -\boldsymbol{H}_k^T(\boldsymbol{h}_y - \boldsymbol{H}_{\cdot,-k}\boldsymbol{\alpha}_{-k} - \boldsymbol{H}_{\cdot,k}\boldsymbol{\alpha}_k) \\
&\quad + \beta(\sum_j \hat{\boldsymbol{Q}}_{kj}\boldsymbol{\alpha}_j) + \beta(\sum_i \boldsymbol{\alpha}_i\hat{\boldsymbol{Q}}_{ik}), \\
&= -\boldsymbol{H}_k^T(\boldsymbol{h}_y - \boldsymbol{H}\boldsymbol{\alpha}) \\
&\quad + \beta\hat{\boldsymbol{Q}}_{k,\cdot}\boldsymbol{\alpha} + \beta\boldsymbol{\alpha}^T\hat{\boldsymbol{Q}}_{\cdot,k}. \\
&= \boldsymbol{H}_k^T\boldsymbol{H}\boldsymbol{\alpha} - \boldsymbol{H}_k^T\boldsymbol{h}_y + 2\beta\boldsymbol{\alpha}^T\hat{\boldsymbol{Q}}_{\cdot,k}.
\end{aligned} \tag{12}
$$

3.6.2 Proximal Operator

The componentwise proximal operator of $g(\alpha_i)$ which is in ℓ_1-norm is:

$$
\mathrm{Prox}_{\lambda\|\cdot\|_1}(\alpha_i) = \arg\min_{\hat{\alpha}} \frac{1}{2}\|\hat{\alpha}_i - \alpha_i\|_2^2 + \lambda\|\hat{\alpha}\|_1, \tag{13}
$$

and the solution is:

$$
\hat{\alpha}_i^{lasso} = \mathrm{sign}(\hat{\alpha}_i)\max(|\hat{\alpha}_i| - \lambda, 0) \tag{14}
$$

3.6.3 Scalability

For large dataset, which means the MTS has many sensors in our case, the running time of regular coordinate descent algorithm is kind of challenge. To improve the scalability, we take the advantage of multi-core architecture of modern computer system and the power of statistic optimization. To be specific, we apply the asynchronous stochastic coordinate descent algorithm proposed in [Liu and Wright, 2015].

3.6.4 PIE-SS

With the solution of object function (4), we present the "PIE-SS" as in (3).

Algorithm 3: PIE-SS

Input: MTS and labels: $\{(\boldsymbol{X}_i, y_i)\}$.
Output: Subset of features.
for $i \leftarrow 1$ **to** m **do**
  build the distance graph $\boldsymbol{M}_i$;
  generate the k-nearest neighbor graph $\boldsymbol{W}_i$;
  $\boldsymbol{W}_i = 0.5 * (\boldsymbol{W}_i + \boldsymbol{W}_i^T)$;
  $\boldsymbol{v}^t = \mathrm{PIE}(\boldsymbol{W}_i)$;
  Solve the object function (4) and obtain results $\boldsymbol{\alpha}$;
  Select features that their coefficients are larger than zero: $\boldsymbol{\alpha}_i > 0$;
end
return *Selected features.*

4 Connection to Two-Stage Multiple Kernel Learning

Our proposed feature subset selection algorithm has strong connection to two stage multiple kernel learning [Kumar et al., 2012] [Cortes et al., 2010] [Kandola et al., 2002] and kernel-target alignment [Cristianini et al., 2002] in theory. The target alignment problem is defined as follows.

$$
\max_{\mu\geq 0} \frac{\langle\sum_{l=1}^p \mu_l\boldsymbol{K}_l, \boldsymbol{K}^{(t)}\rangle}{\|\sum_{l=1}^p \mu_l\boldsymbol{K}_l\|_F}, \text{s.t.} \quad \|\mu\|_2 = 1, \tag{15}
$$

where A is the Gram matrix of kernel A on the training samples, $\langle A, B\rangle = tr(AB^T)$ and $\|A\|_F^2 = tr(AA^T)$. The $\boldsymbol{K}^{(t)}$ is the target kernel, and $\boldsymbol{K}_l$ is predefined kernel from a set which has size equals to p.

The problem of feature subset algorithm in this works is similar to two-stage kernel learning problem. Both of them learn a non-negative linear combination of base kernels that maximizes the alignment with one target kernel. The similarity graph $\boldsymbol{W}_y$ in object function (4) can be treated as a target kernel $\boldsymbol{K}^{(t)}(\boldsymbol{X}_i, \boldsymbol{X}_j) = y_iy_j$ in binary classification case. For multiple label classification cases, it is natural to use "one-vs-all" strategy to redefine the problem.

The most obvious difference is that two-stage kernel learning problem define base kernels at instance side while our feature subset selection algorithm define base kernels at feature side. The two-stage kernel learning problem tries to search the optimal linear combination of a set of predefined kernel (size is specified by the user) on instances, and hope the combined kernels can show a data distribution as similar as target kernels which represent the distribution of labels. Contrarily, the feature subset selection problem in this work defines kernels (or similarity graphs in current setting) for each single feature, and the number of predefined kernels is equal to the number of features. The k-nearest neighbor graphs used in this work can be extended to other popular kernels such as Gaussian kernel, Epanechnikov Kernel and so on.

Another difference is that the feature subset selection problem does not include the learning of classifier/regressor which is the second stage of two-stage kernel learning problem. Since the focus of this work

is feature selection.

5 Experiments

5.1 Datasets

Our experiment results includes seven public available MTS datasets. One of them, named as "BasicMotions", is used for illustration purpose only and be excluded from comparisons. The remained six datasets are used for evaluation. Two for them, "DuckDuckGeese" and "PEMS-SF", are from the UEA & UCR time series classification repository [Bagnall et al., 2017]. The "EGG" (large) dataset is from UCI [Dua and Graff, 2017]. The "CMU-MOCAP-S16" and "KickvsPunch" is from [Baydagon, 2015]. The "HumanGait" dataset is from [Tanawongsuwan and Bobick, 2003]. A summary of them is presented in Table (2).

Name	Train	Test	#Sensors	Length	Class
BasicMotions	40	N/A	6	100	4
DuckDuckGeese	60	40	1345	270	5
EEG (UCI)	468	480	64	256	2
CMU-MOCAP-S16	29	29	62	127-580	2
KickvsPunch	16	10	62	274-841	2
HumanGait	270	270	66	133	15
PEMS-SF	267	173	963	114	7

Table 2: Summary of selected MTS datasets. The "BasicMotions" dataset is used for illustration in Fig. (2). For "CMU-MOCAP-S16" and "KickvsPunch", the length of different sensors are not the same.

5.2 Baseline Algorithms

To evaluate the performance of our proposed algorithms, we select two famous algorithms CLeVer [Yoon et al., 2005] and Corona [Yang et al., 2005] as our baseline. We also aware that there exists one latest work CSFS [Han and Liu, 2013] which has the same idea as Corona but use the Mutual Information instead of Correlation to vectorize each MTS. However, the calculation of Mutual Information for MTS with large number of sensors is a problem as we discussed in Section 3.6, let alone the CSFS need to repeat the calculation for each data sample. By this reason, we drop our comparison with it.

- CLeVer: the CLeVer algorithm uses the loadings of common principal component analysis to measure the importance of each sensor. First, a correlation coefficient matrix is built among different sensors for each MTS segment. Secondly, principal components of each coefficient matrix are calculated. Thirdly, all these principal components are aggregated together and the descriptive common principal components are calculated. Finally, the $\ell^2 - norm$ of loading vectors are used to rank the importance of each sensor.

- Corona: the Corona algorithm uses the flattened correlation coefficient matrix as feature vector and recursive feature elimination [Guyon et al., 2002] to rank sensors. The coefficients of trained support vector machine are used to indicate the importance of each sensors during the iteration.

5.3 Evaluation Classifier and Metrics

We use one nearest neighbor (NN-1) classifier as our benchmark algorithm following the UEA multivariate time series website [Bagnall et al., 2018]. Moreover, to measure the performance of PIE-SS, we use two different settings to aggregate the selected features (sensors). One is a unweighted version which can be found in most feature selection research works, another is a weighted version since we learn a linear combination though Equation (4). The details are as follows.

1. Unweighted aggregation of sensors:

$$dist_{unweighted} = \sum_{i=1}^{k} \boldsymbol{W}_i.$$

2. Weighted aggregation of sensors:

$$dist_{weighted} = \sum_{i=1}^{k} \alpha_i \boldsymbol{W}_i,$$

where $dist_*$ is the aggregated distance matrix and will be used for nearest neighbor classifier, k is the number of selected sensors.

For the evaluation metric, we use Accuracy(ACC).

5.4 Parameter Settings

We list the configuration of parameters (if any) for each algorithm as below.

- CLeVer: we set the $\delta = 0.7$ as in Alg. 1 in [Yoon et al., 2005].

- PIE-rank: when calculating the k-nearest neighbor graph of each sensor, we set k equal to 10.

- PIE-SS: the β is set to 1.0. The λ parameter is used to control the sparsity of results.

5.5 Discussion

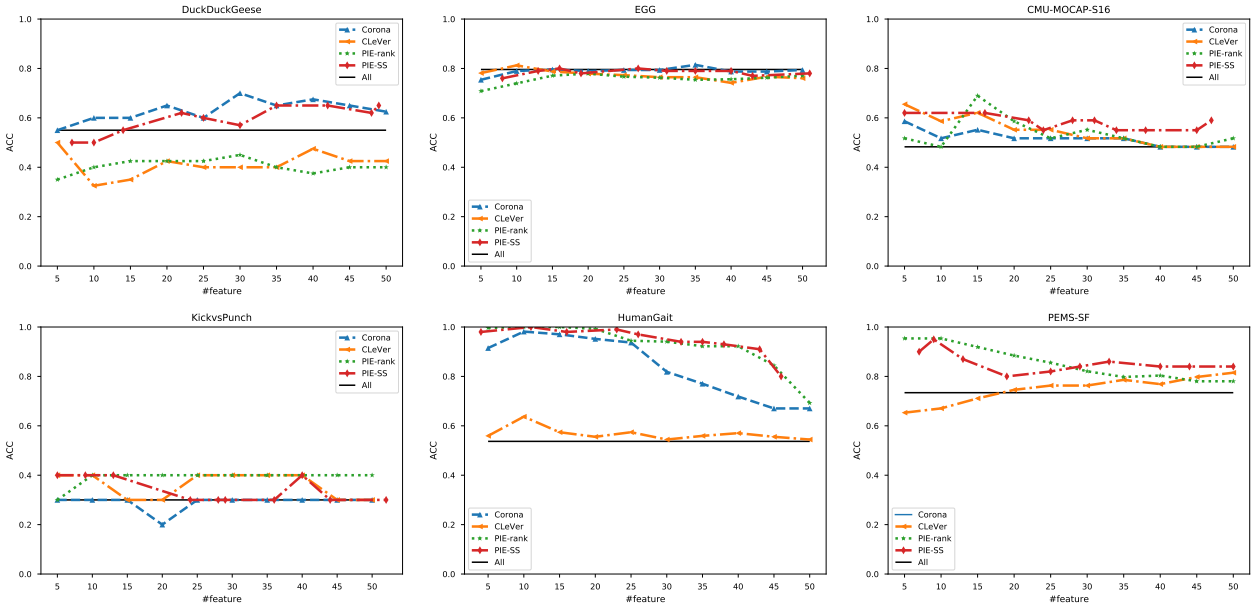

Figure 3: Classification results of different algorithms on selected MTS data. The "unweighted" version results of "PIE-SS" is reported here. Specially, for the PEMS-SF dataset, the results of "Corona" algorithm is missing as the time of calculating it is more than 24 hours (with our implementation).

Name/ACC	*dists*	Size 1	Size 2	Size 3	Size 4	Size 5	Size 6	Size 7	Size 8	Size 9	Size 10
CMU-MOCAP-S16	W	**0.62**(5)	**0.62**(16)	0.55(22)	0.55 (24)	0.55 (28)	0.55(31)	0.59(34)	0.55(38)	0.55(45)	0.59(47)
CMU-MOCAP-S16	UW	**0.62**(5)	**0.62**(16)	0.59(22)	0.55 (24)	0.59 (28)	0.59(31)	0.55(34)	0.55(38)	0.55(45)	0.59(47)
DuckDuckGeese	W	0.53(8)	0.50(10)	0.55(14)	0.57(22)	0.53(25)	0.57(30)	0.62(35)	0.62(42)	0.60(48)	**0.65**(49)
DuckDuckGeese	UW	0.50(8)	0.50(10)	0.55(14)	0.62(22)	0.60(25)	0.57(30)	**0.65**(35)	**0.65**(42)	0.62(48)	0.65(49)
EGG	W	0.77(8)	0.80(13)	0.82(16)	0.81(19)	0.81(27)	0.80(31)	0.80(35)	0.79(40)	**0.83**(43)	0.79(51)
EGG	UW	0.76(8)	0.79(13)	**0.80**(16)	0.78(19)	**0.80**(27)	0.79(31)	0.79(35)	0.79(40)	0.77(43)	0.79(51)
KickvsPunch	W	**0.60**(5)	0.40(9)	0.50(13)	0.50(24)	0.3(28)	0.40(29)	0.30(36)	0.4(40)	0.40(44)	0.40(52)
KickvsPunch	UW	**0.40**(5)	**0.40**(9)	**0.40**(13)	0.30(24)	0.3(28)	0.30(29)	0.30(36)	0.4(40)	0.30(44)	0.30(52)
HumanGait	W	0.98(4)	**1.00**(11)	**1.00**(16)	**1.00**(23)	**1.00**(26)	**1.00**(32)	**1.00**(35)	**1.00**(38)	0.99(43)	0.97(46)
HumanGait	UW	0.98(4)	**1.00**(11)	0.98(16)	0.99(23)	0.97(26)	0.94(32)	0.94(35)	0.93(38)	0.91(43)	0.80(46)
PEMS-SF	W	**0.95**(7)	**0.95**(9)	0.89(13)	0.88(19)	0.87(25)	0.87(29)	0.88(33)	0.88(40)	0.87(44)	0.87(50)
PEMS-SF	UW	0.90(7)	**0.95**(9)	0.87(13)	0.80(19)	0.82(25)	0.84(29)	0.86(33)	0.84(40)	0.84(44)	0.84(50)

Table 3: Classification performance by PIE-SS with different size of selected sensors. The size of selected sensors are put inside the parenthesis. Results include two different settings of aggregating distance matrix. They are marked as "UW" for '$dist_{unweighted}$" and "W" for '$dist_{unweighted}$". The integer value in the parenthesis after each accuracy value means the size of selected sensors by the PIE-SS algorithm. The highest value of each row is marked as bold.

Comparison among algorithms We compare the performance among PIE-rank, PIE-SS (unweighted aggregation), CLeVer and Corona. First, we obtain the sensor selection results of them by using the training samples. Secondly, we calculate the distance matrix for each sensor crossing different samples (include both training and testing) by using DTW. Lastly, the final distance matrix $dist_{unweighted}$ is calculated, and the classification performance is evaluated by NN-1 classifier. The accuracy results are reported in Fig. (3). The number of selected sensors is from 5 to 50 with step equals to 5.

Overall, there is no single algorithm shows dominated higher accuracy. The performance of Corona and PIE-SS are quite close to each other and they show better performance than using all sensors (that means no sensor selection) generally. For CLeVer, its performance is not stable as it almost show worse classification accuracy than without doing any sensor selections in dataset "DuckDuckGeese" and "EGG". The PIE-rank algorithm show same unstable performance as PIE-SS. It shows lower performance in dataset "DuckDuckGeese" and "EGG", but has very good performance in other datasets. It even beats Corona and PIE-SS in dataset "EGG" and "PEMS-SF" at small number of selected sensors.

Meanwhile, we need to mention that "CLever" is an unsupervised sensor selection algorithm and the label information is not used for improving the classification performance.

Compare the performance of without sensor selection Another important evaluation is to measure the success of sensor selection. We draw a horizon line in each figure to show the performance of NN-1 classifier without any sensor selection. If one algorithm's curve is above the line, it means the sensor selection is success. Otherwise, it means fail. For example, in plot of "DuckDuckGeese", the PIE-rank and CLeVer algorithms are fail.

However, it worth to mention that PIE-rank show strong performance in dataset "CMU-MOCAP-16", "KickvsPunch", "HumanGait" and "PEMS-SF". Considering the advantage of its calculation time, it is quite an effective algorithm.

Comparison between weighted and unweighted aggregation The result of PIE-SS by using different aggregation strategy is reported in Table (2). The weighted version show better classification performance than unweighted one except for dataset "DuckDuckGeese". The results match our expectation since we learn a linear combination of similarity graph to approximate the label matrix.

6 Application to Heterogeneous Data

In this section, we present an application of our PIE-rank algorithms on heterogeneous data. The motivation is to show the unique capability of our approach comparing to existing MTS related feature selection algorithms. Especially, (1) we avoid the feature extraction for time series, (2) we do not need the distance between two heterogeneous features.

We choose the public available data "The PhysioNet Computing in Cardiology Challenge 2012" [Silva et al., 2012] [Goldberger et al., 2000] for our experiment. The original goal of the challenge is to predict the mortality of ICU patients. Our goal is to rank the importance of features and compare the result with several published works [Severeyn et al., 2012] [Krajnak et al., 2012] [Di Marco et al., 2012] [McMillan et al., 2012] in the Cardiology research domain. (We want to mention here that all the authors of this work have no domain knowledge about the Cardiology related research other than average common sense.) The introduced data is heterogeneous based on following observations:

- Different type of feature values: real values, category values and time series,
- The time series have different length not only among different sensor, also across different instances.
- The time series of different sensors have different sampling rate.

Name	Set-A	Set-B
#Samples	2099	2064
#Label of "1"	259	282
#Real value features	3	3
#Category features	2	2
#MTS	37	37

Table 4: Summary of cleaned datasets of Physionet Challenge 2012. Label "1" means not survival.

Except the heterogeneous characteristic, the data also include a lot of missing values. To prepossess the data, we follow part of the data cleaning procedures as introduced in work [Silva et al., 2012] [Johnson, 2018]. To be specific, we only apply the steps in the author's Notebook before the section of feature extraction for time series. After prepossessing, we obtain two cleaned datasets "Set-A" and "Set-B" as in Table (4). When we build graph for each feature, we use Euclidean distance for feature: "Age", "Height" and "Weight (static one)", and 1/0 distance for category feature: "Gender" and "ICUType". For time series features, we apply DTW distance.

We calculate the ranking scores of "Set-A" and "Set-B" by using "PIE-rank" and obtain two versions of ranking results. To obtain the final ranking scores, we use the average of them and report the result in Figure (4).

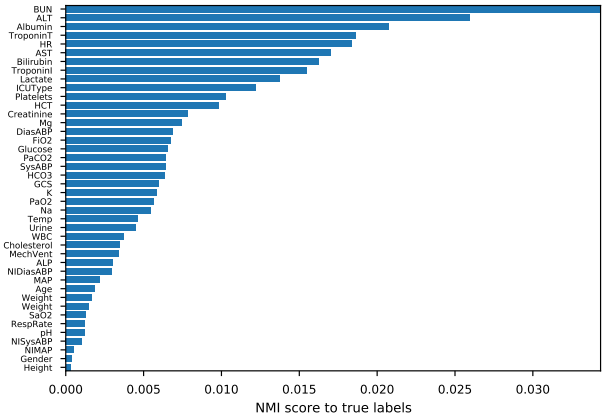

Figure 4: Ranking result of Physionet Challenge 2012 by PIE-rank.

Comparison to domain knowledge as in [Krajnak et al., 2012] In the work [Krajnak et al., 2012], the author provides 19 selected features by the clinicians (as shown in Table 2 of the original paper). Among, them 15 unique raw features are used. The other four repeat features with different extraction methods. We check the rank of these 15 raw features in our ranking result and see how many of them are ranked high. From the result of Table (5), we found

Feature	Rank
Age	33
Bilirubin	7
BUN	1
Creatinine	13
Glasgow Coma Score (GCS)	21
HCO3	20
Heart Rate (HR)	5
PaO2	23
pH	38
Platelets	11
Potassium (K)	22
Systolic ABP (SysABP)	19
Temp	25
Urine	26
White Blood Cell count (WBC)	27

Table 5: Ranking result of 19 selected features by the clinicians in work [Krajnak et al., 2012].

that 8 out of 15 raw features are ranked in the top half of original 42 raw features by our PIE-rank algorithm.

Comparison to simple correspondence analysis as in [Severeyn et al., 2012] The work [Severeyn et al., 2012] uses the simple correspondence analysis technique to analyze the importance of features that related to the survive or not. In the conclusion part (Section 4 of the original paper), the author list eight important features. The ranking result of them by PIE-rank are shown in table (6) As we can see, 4 out

Feature	Rank
Creatinine	13
Urine	26
BUN	1
Bilirubin	7
GCS	21
MechVent	29
SOFA score	NA
SAPS score	NA

Table 6: Ranking result of eight selected features as in [Severeyn et al., 2012]. The SOFA and SAPS scores are not included in our preprocessed data.

of 6 selected features are ranked in the top half by our PIE-rank algorithm.

Comparison to feature selection result I as in [Di Marco et al., 2012] In work [Di Marco et al., 2012], the authors used forward sequential selection algorithm and cross-validation (CV) to select 32 most important features reported the occurrences of them in the repeated evaluations by CV.

No	Features (rank by PIE-rank)
10	Age(33), GCS(21), Temp(25), BUN(1), Glucose(17)
9	Weight(34,35), Sodium("Na"")(24), WBC(27), Bilirubin(7), Cholesterol(28)
8	Height(42), ICU Type(10), TroponinI(8), TroponinT(4)
7	PaCO2(18), RespRate(37), HR(5), HCT(12), Albumin(3), ALP(30)
6	SaO2(36), NISysABP(39), Mg(14), Platelets(11), Lactate(9)
5	PaO2(23)
4	SysABP(19)
3	pH(38), MAP(32), NIDiasABP(31), ALT(2), AST(6)

Table 7: Ranking result of 32 selected features as in [Di Marco et al., 2012]. The "Occurrence" (the "No" column) values are provided in Table 1 of the cited work.

Comparison to feature selection result II as in [McMillan et al., 2012] In work [McMillan et al., 2012], the authors present three list of ranked top-10 features by their proposed algorithms. We compare their ranking results with us as in Table (**??**).

Rank	Result-I	Result-2	Result-3
1	K(22)	K(22)	K(22)
2	Age(33)	Albumin(3)	Temp(25)
3	ALP(30)	Age(33)	HR(5)
4	Platelets(11)	Glucose(17)	ALP(30)
5	Temp(25)	Urine(26)	Albumin(3)
6	Urine(26)	ALP(30)	PaO2(23)
7	Glucose(17)	GCS(21)	Age(33)
8	BUN(1)	pH(38)	Temp(25)
9	pH(38)	Albumin(3)	RespRate(37)
10	ALT(2)	Urine(26)	ALP(30)

Table 8: Ranking results of selected features by [Di Marco et al., 2012]. The rank results are provided in Table 1 of the cited work. The value in the parenthesis are the ranking result of our PIE-rank algorithm.

Summary From Table (5) to (8), we can see that different algorithms have different top-ranked features. Here we present a simple counting analysis of feature occurrences crossing four tables. We only list the features that appear at least three times out of four tables. The analysis results are shown in Table (9).

There are ten features that appear at least three times among all four tables. Three of them are ranked in top-10 by our "PIE-rank" algorithm. The "BUN" and "GCS" features seems to be the most important features if we consider all results here (include our ranking result). The very contradictory features are "Age"

No	Features	PIE-rank
4	BUN	1
4	GCS	21
3	Age	33
3	Urine	26
3	HR	5
3	PaO2	23
3	pH	38
3	Platelets	11
3	Temp	25
3	Bilirubin	7

Table 9: Summary of ten features that are counted most among four tables. "No" means the occurrences.

and "pH"that are considered important in Cardiology research field but show less impact by our algorithm regarding the survival or not for patients at ICU.

7 Conclusion and Future Works

In this work, we introduce two algorithms to rank and select subset of features (or sensors) for multivariate time series data. The unique part of our algorithms is that we avoid the feature extraction for each single time series which usually requires the domain knowledge regarding the input data.

Our algorithms have substantial connection to multiple kernel learning. The adjacency matrix of our graph representation of each feature has the same form and meaning as a kernel matrix. It is natural to extend our work by connect it with kernel-based learning algorithms. Nevertheless, we also observe the computation issues when we build graph representation for each feature. In the future, we may try approximation algorithms such as Matrix sketching to reduce the computation time or apply deep learning algorithms to learn an embedding vector regarding the graph cluster structure directly.

References

[Bagnall et al., 2018] Bagnall, A., Dau, H. A., Lines, J., Flynn, M., Large, J., Bostrom, A., Southam, P., and Keogh, E. (2018). The uea multivariate time series classification archive, 2018. *arXiv preprint arXiv:1811.00075*.

[Bagnall et al., 2017] Bagnall, A., Lines, J., Bostrom, A., Large, J., and Keogh, E. (2017). The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 31:606–660.

[Baydagon, 2015] Baydagon, M. (2015). Multivariate time series classification data sets (in matlab format).

[Berndt and Clifford, 1994] Berndt, D. J. and Clifford, J. (1994). Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA.

[Cai et al., 2010] Cai, D., Zhang, C., and He, X. (2010). Unsupervised feature selection for multi-cluster data. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 333–342. ACM.

[Cortes et al., 2010] Cortes, C., Mohri, M., and Rostamizadeh, A. (2010). Two-stage learning kernel algorithms. In *27th International Conference on Machine Learning, ICML 2010*, pages 239–246.

[Cristianini et al., 2002] Cristianini, N., Shawe-Taylor, J., Elisseeff, A., and Kandola, J. S. (2002). On kernel-target alignment. In *Advances in neural information processing systems*, pages 367–373.

[Di Marco et al., 2012] Di Marco, L. Y., Bojarnejad, M., King, S. T., Duan, W., Di Maria, C., Zheng, D., Murray, A., and Langley, P. (2012). Robust prediction of patient mortality from 48 hour intensive care unit data. In *2012 Computing in Cardiology*, pages 477–480. IEEE.

[Dua and Graff, 2017] Dua, D. and Graff, C. (2017). UCI machine learning repository.

[Friedman et al., 2010] Friedman, J., Hastie, T., and Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1.

[Goldberger et al., 2000] Goldberger, A. L., Amaral, L. A., Glass, L., Hausdorff, J. M., Ivanov, P. C., Mark, R. G., Mietus, J. E., Moody, G. B., Peng, C.-K., and Stanley, H. E. (2000). Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215–e220.

[Guyon et al., 2002] Guyon, I., Weston, J., Barnhill, S., and Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1-3):389–422.

[Han and Liu, 2013] Han, M. and Liu, X. (2013). Feature selection techniques with class separability for multivariate time series. *Neurocomputing*, 110:29–34.

[Jakobsen, 2014] Jakobsen, S. K. (2014). Mutual information matrices are not always positive semidefinite. *IEEE Transactions on information theory*, 60(5):2694–2696.

[Johnson, 2018] Johnson, A. (2018). Python code parsing data from physionet challenge 2012.

[Kandola et al., 2002] Kandola, J., Shawe-Taylor, J., and Cristianini, N. (2002). Optimizing kernel alignment over combinations of kernel.

[Keogh and Ratanamahatana, 2005] Keogh, E. and Ratanamahatana, C. A. (2005). Exact indexing of dynamic time warping. *Knowledge and information systems*, 7(3):358–386.

[Krajnak et al., 2012] Krajnak, M., Xue, J., Kaiser, W., and Balloni, W. (2012). Combining machine learning and clinical rules to build an algorithm for predicting icu mortality risk. In *2012 Computing in Cardiology*, pages 401–404. IEEE.

[Kumar et al., 2012] Kumar, A., Niculescu-Mizil, A., Kavukcuoglu, K., and Daume III, H. (2012). A binary classification framework for two-stage multiple kernel learning. *arXiv preprint arXiv:1206.6428*.

[Li et al., 2016] Li, J., Cheng, K., Wang, S., Morstatter, F., Robert, T., Tang, J., and Liu, H. (2016). Feature selection: A data perspective. *arXiv:1601.07996*.

[Lin and Cohen, 2010] Lin, F. and Cohen, W. W. (2010). Power iteration clustering. In *ICML*, volume 10, pages 655–662. Citeseer.

[Liu and Wright, 2015] Liu, J. and Wright, S. J. (2015). Asynchronous stochastic coordinate descent: Parallelism and convergence properties. *SIAM Journal on Optimization*, 25(1):351–376.

[Liu et al., 2010] Liu, W., He, J., and Chang, S.-F. (2010). Large graph construction for scalable semi-supervised learning. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 679–686.

[McMillan et al., 2012] McMillan, S., Chia, C.-C., Van Esbroeck, A., Rubinfeld, I., and Syed, Z. (2012). Icu mortality prediction using time series motifs. In *2012 Computing in Cardiology*, pages 265–268. IEEE.

[Nguyen et al., 2014] Nguyen, X. V., Chan, J., Romano, S., and Bailey, J. (2014). Effective global approaches for mutual information based feature selection. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 512–521. ACM.

[Peng et al., 2005] Peng, H., Long, F., and Ding, C. (2005). Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on pattern analysis and machine intelligence*, 27(8):1226–1238.

[Peng et al., 2015] Peng, R., Sun, H., and Zanetti, L. (2015). Partitioning well-clustered graphs: Spectral clustering works! In *Conference on Learning Theory*, pages 1423–1455.

[Severeyn et al., 2012] Severeyn, E., Altuve, M., Ng, F., Lollett, C., and Wong, S. (2012). Towards the prediction of mortality in intensive care units patients: A simple correspondence analysis approach. In *2012 Computing in Cardiology*, pages 469–472. IEEE.

[Shi and Malik, 2000] Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905.

[Silva et al., 2012] Silva, I., Moody, G., Scott, D. J., Celi, L. A., and Mark, R. G. (2012). Predicting in-hospital mortality of icu patients: The physionet/computing in cardiology challenge 2012. In *2012 Computing in Cardiology*, pages 245–248. IEEE.

[Tanawongsuwan and Bobick, 2003] Tanawongsuwan, R. and Bobick, A. (2003). Performance analysis of time-distance gait parameters under different speeds. In *International Conference on Audio-and Video-Based Biometric Person Authentication*, pages 715–724. Springer.

[Wright, 2015] Wright, S. J. (2015). Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34.

[Yang et al., 2005] Yang, K., Yoon, H., and Shahabi, C. (2005). A supervised feature subset selection technique for multivariate time series. In *Proceedings of the workshop on feature selection for data mining: Interfacing machine learning with statistics*, pages 92–101.

[Yoon et al., 2005] Yoon, H., Yang, K., and Shahabi, C. (2005). Feature subset selection and feature ranking for multivariate time series. *IEEE transactions on knowledge and data engineering*, 17(9):1186–1198.

[Yu and Liu, 2004] Yu, L. and Liu, H. (2004). Efficient feature selection via analysis of relevance and redundancy. *Journal of machine learning research*, 5(Oct):1205–1224.